Contents

Finance and Operations application documentation

2/18/2021 • 2 minutes to read • Edit Online

Learn how to make Finance and Operations applications work for your business, using the resources in this topic to find great content for end users, developers, and IT professionals.

Much of this content also applies to the related products: Dynamics 365 Commerce and Dynamics 365 Human Resources.

Budgeting

Cash and bank management

Compliance

Cost accounting

Fixed assets

General ledger

Project management

Public sector

Regional regulatory features

Human resources

Benefits

Employee development and training

Questionnaires

Recruiting

US payroll

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Development and administration for Finance and Operations apps

2/18/2021 • 3 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

Development and administration for Finance and Operations apps includes:

- Administrator experience and Lifecycle Services
- Developer experience
- Intelligence
- Mobile apps
- Data management and data entities
- Office integration

Developer experience

The developer experience is based on modern tooling using Visual Studio and .NET components.

- The development tools are decoupled from any running environment, which means that you develop against local, XML-based files, not the online database.
- Microsoft Visual Studio is the development environment. Finance and Operations customizes the Visual Studio environment to provide you with a smooth and familiar experience.
- The X++ compiler generates Common Intermediate Language (CIL) for all features. CIL is the same intermediate language used by other .NET-based (managed) languages, such as the C# programming language.
- You can leverage the browser-based client and design patterns for forms to provide an improved end-user experience.
- The Application Lifecycle Management (ALM) system supports build automation, test automation, and deployment of models to the cloud.

For more information, see Develop and customize home page.

Administrator experience and Lifecycle Services

Finance, Supply Chain Management, and Commerce are cloud-hosted. As an IT professional, you can use Dynamics Lifecycle Services (LCS) to monitor and tune your environments, deploy features, and stay up to date with recent hotfixes. Within your deployment, you can configure security, and manage when processes run. You can also use LCS when you are called on to support business intelligence and reporting, mobile apps, Office, and other integrations.

BI & reporting

Finance and Operations provides five distinct reporting experiences. Specialized tools are provided to meet the complex and diverse reporting needs of various functions throughout the organization.

- Operational views – Designed to address the specific needs of a given business persona.
- Business documents – Static documents used to capture and exchange processed business data.
- Analytical tools and visualizations – Personalized presentations of logical calculations that allow the user to explore their data.
- Electronic reporting – Tool used to configure formats for electronic documents.
- Financial reporting – Designed to provide in-depth accounting management tools based on standard views of financial activities across legal entities.

Mobile apps

The Finance and Operations mobile app empowers your organization to mobilize its business processes. After your IT admin enables the mobile workspaces feature for your organization, users can sign in to the app and immediately begin to run business processes from their mobile devices. The Dynamics 365 for Finance and Operations mobile app includes the following features that can help increase productivity:

- Users can view, edit, and act on business data, even if they have intermittent network connectivity or their mobile devices are offline. When a device reestablishes a network connection, offline data operations are automatically synchronized with Finance and Operations.
- IT admins or developers can build and publish mobile workspaces that have been tailored to their organization. The app uses your existing code assets, so you don't have to re-implement your validation procedures, business logic, or security configuration.
- IT admins or developers can easily design mobile workspaces by using the point-and-click workspace designer that is included with the Finance and Operations web client.
- IT admins or developers can optionally optimize the offline capabilities of workspaces by using the Business logic extensibility framework. Because data continues to be processed while a device is offline, your mobile scenarios remain rich and fluid, even if devices don't have constant network connectivity.

Data management and data entities

Data from Finance and Operations can easily be integrated with Microsoft and non-Microsoft data sources using Dataverse, Power Apps, and Power BI. For more information, see Data entities overview.

Office integration

The Microsoft Office integration capabilities provide users with a productive environment that helps them get the job done by using Office products. For more information, see Office integration overview.

eLearning courses

For online courses and training, check out Dynamics 365 Finance and Operations on Microsoft Learn.

What's new or changed in Finance and Operations apps home page

2/18/2021 • 2 minutes to read • Edit Online

Application releases

To see what's new or changed in each release of a Finance and Operations app, see the following topics:

- **Finance:** What's new or changed in Dynamics 365 Finance
- **Supply Chain Management:** What's new or changed in Dynamics 365 Supply Chain Management
- **Commerce:** What's new or changed in Dynamics 365 Commerce
- **Human Resources:** What's new or changed in Dynamics 365 Human Resources

Platform updates

To see what's new or changed in the Platform updates for Finance and Operations apps, see the following topic:

- What's new or changed in Platform updates

Lifecycle Services releases

To see what's new or changed in Lifecycle Services, see the following topic:

- What's new or changed in Lifecycle Services (LCS)

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

What's new and changed in Platform updates

2/18/2021 • 3 minutes to read • Edit Online

To see what's new or changed in the Platform updates for Finance and Operations apps, see the following topics.

VERSION	BUILD NUMBER	AUTO-UPDATE AVAILABILITY	LEARN MORE
Platform updates for 10.0.17	7.0.5934	April 2021	Platform updates for version 10.0.17 of Finance and Operations apps
Platform updates for 10.0.16	7.0.5860	February 2021	Platform updates for version 10.0.16 of Finance and Operations apps
Platform updates for 10.0.15	7.0.5816	January 2021	Platform updates for version 10.0.15 of Finance and Operations apps
Platform updates for 10.0.14	7.0.5778	November 2020	Platform updates for version 10.0.14 of Finance and Operations apps
Platform updates for 10.0.13	7.0.5746	October 2020	Platform updates for version 10.0.13 of Finance and Operations apps
Platform updates for 10.0.12	7.0.5688	August 2020	Platform updates for version 10.0.12 of Finance and Operations apps
Platform updates for 10.0.11	7.0.5644	July 2020	Platform updates for version 10.0.11 of Finance and Operations apps
Platform updates for 10.0.10	7.0.5600	May 2020	Platform updates for version 10.0.10 of Finance and Operations apps
Platform update 33	7.0.5559	April 2020	What's new or changed in Platform update 33 for Finance and Operations apps
Platform update 32	7.0.5493	February 2020	What's new or changed in Platform update 32 for Finance and Operations apps
Platform update 31	7.0.5457	January 2020	What's new or changed in Platform update 31 for Finance and Operations apps

VERSION	BUILD NUMBER	AUTO-UPDATE AVAILABILITY	LEARN MORE
Platform update 30	7.0.5407	November 2019	What's new or changed in Platform update 30 for Finance and Operations apps
Platform update 29	7.0.5372	October 2019	What's new or changed in Platform update 29 for Finance and Operations apps
Platform update 28	7.0.5314	July 2019	What's new or changed in Dynamics 365 for Finance and Operations platform update 28
Platform update 27	7.0.5286	June 2019	What's new or changed in Dynamics 365 for Finance and Operations platform update 27
Platform update 26	7.0.5257	May 2019	What's new or changed in Dynamics 365 for Finance and Operations platform update 26
Platform update 25	7.0.5222	April 2019	What's new or changed in Dynamics 365 for Finance and Operations platform update 25
Platform update 24	7.0.5179	March 2019	What's new or changed in Dynamics 365 for Finance and Operations platform update 24
Platform update 23	7.0.5126	January 2019	What's new or changed in Dynamics 365 for Finance and Operations platform update 23
Platform update 22	7.0.5095	December 2018	What's new or changed in Dynamics 365 for Finance and Operations platform update 22
Platform update 21	7.0.5073	November 2018	What's new or changed in Dynamics 365 for Finance and Operations platform update 21
Platform update 20	7.0.5030	September 2018	What's new or changed in Dynamics 365 for Finance and Operations platform update 20

VERSION	BUILD NUMBER	AUTO-UPDATE AVAILABILITY	LEARN MORE
Platform update 15	7.0.4841	March 2018	What's new or changed in Dynamics 365 for Finance and Operations, Enterprise edition platform update 15
Platform update 12	7.0.4709	November 2017	What's new or changed in Dynamics 365 for Finance and Operations, Enterprise edition platform update 12
Platform update 11	7.0.4679.35176	October 2017	What's new or changed in Dynamics 365 for Finance and Operations, Enterprise edition platform update 11
Platform update 10	7.0.4641.16233	August 2017	What's new or changed in Dynamics 365 for Finance and Operations, Enterprise edition platform update 10
Platform update 9	7.0.4612.35162	July 2017	What's new or changed in Dynamics 365 for Finance and Operations, Enterprise edition platform update 9
Platform update 8	7.0.4565.16212	June 2017	What's new or changed in Dynamics 365 for Finance and Operations, Enterprise edition platform update 8
Platform update 7	7.0.4542.16189	May 2017	What's new or changed in Dynamics 365 for Operations platform update 7
Platform update 6	7.0.4509.16180	April 2017	What's new or changed in Dynamics 365 for Operations platform update 6
Platform update 5	7.0.4475.16165	March 2017	What's new or changed in Dynamics 365 for Operations platform update 5
Platform update 4	7.0.4425.16161	February 2017	What's new or changed in Dynamics 365 for Operations platform update 4
Platform update 3	7.0.4307.16141	November 2016	What's new or changed in Dynamics 365 for Operations platform update 3

VERSION	BUILD NUMBER	AUTO-UPDATE AVAILABILITY	LEARN MORE
Platform update 2	7.0.4230.16130	August 2016	What's new or changed in Dynamics AX platform update 2
Platform update 1	7.0.4127.16103	May 2016	What's new or changed in Dynamics AX platform update 1
Dynamics AX 7.0	7.0.4030.16079	February 2016	What's new or changed in Dynamics AX 7.0

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Platform updates for version 10.0.17 of Finance and Operations apps (April 2021)

2/18/2021 • 2 minutes to read • Edit Online

IMPORTANT

Some or all of the functionality noted in this topic is available as part of a preview release. The content and the functionality are subject to change. For more information about preview releases, see Service update availability.

This topic lists the features that are included in the platform updates for version 10.0.17 of Finance and Operations apps. This version has a build number of 7.0.5934 and is available on the following schedule:

- **Preview of release:** February 2021
- **General availability of release (self-update):** March 2021
- **General availability of release (auto-update):** April 2021

Features included in this release

The following features are included in this release. Some of the listed features are still in preview, while others may already be generally available. See the release plan for official release dates for each feature.

- Visual Studio 2017 is now the primary supported version for X++ tools. Visual Studio 2015 is deprecated.
  - For more information, see Removed or deprecated platform features.
- (Preview) Translation support for organization views
  - For more information, see Saved views.
- Allow configuration of the publish batch size for the Excel add-in
  - For more information, see View and update entity data with Excel.
- Upgrade to React 17
- Freeze columns in grids
  - For more information, see Grid capabilities.
- Updates to client feature states
  - See the release plan for details about the client features that were turned on by default or made mandatory with this release.
- View independent software vendor (ISV) license status
  - It is now possible to view status and expiration dates for an ISV license. For more information, see View independent software vendor license status.
- executeQueryWithParameters method
  - This method runs SQL queries by using statement parameters that can help mitigate SQL injection attacks.

Most of these features must be enabled using Feature management before you can use them.

Additional resources

Bug fixes

For information about the bug fixes that are included in this update, sign in to Microsoft Dynamics Lifecycle Services (LCS), and view the KB article.

Dynamics 365: 2021 release wave 1 plan

Wondering about upcoming and recently released capabilities in any of our business apps or platform?

Check out the Dynamics 365: 2021 release wave 1 plan. We've captured all the details, end to end, top to bottom, in a single document that you can use for planning.

Removed and deprecated platform features

The Removed or deprecated platform features topic describes features that have been removed, or that are planned for removal in platform updates of Finance and Operations apps.

- A *removed* feature is no longer available in the product.
- A *deprecated* feature isn't in active development and might be removed in a future update.

A deprecation notice will be added in the Removed or deprecated platform features topic 12 months before the removal of any feature from the product.

For breaking changes that affect only compilation time, but that are binary-compatible with sandbox and production environments, the deprecation time will be less than 12 months. Typically, these changes are functional updates that must be made to the compiler.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Platform updates for version 10.0.16 of Finance and Operations apps (February 2021)

2/18/2021 • 3 minutes to read • Edit Online

This topic lists the features that are included in the platform updates for version 10.0.16 of Finance and Operations apps. This version has a build number of 7.0.5860 and is available on the following schedule:

- **Preview of release:** November 2020
- **General availability of release (self-update):** January 2021
- **General availability of release (auto-update):** February 2021

Features included in this release

The following features are included in this release. Some of the listed features are still in preview, while others may already be generally available. See the release plan for official release dates for each feature.

- User session management
  - Administrators have the ability to set a maximum session timeout for individual users. The range of the session can be from 1 hour to 2160 hours. When this session length expires, the user is required to sign in with their credentials. For more information, see User session management.

- Renamed 'Session idle timeout' to 'Session inactivity timeout'
  - To differentiate the types of session timeouts between maximum user session and inactivity user session, a change in the name was implemented. For more information, see Set the session inactivity timeout.

- Removing support for Visual Studio 2015
  - Version 10.0.16 is the last version to support Visual Studio 2015. For more information, see Removing support for Visual Studio 2015.

- Email throttling
  - This feature allows non-interactive email providers to adhere to a per-minute email sending limit, which prevents errors that are currently triggered when the system attempts to send more emails than the provider can handle. When email throttling is enabled, sending limits for Microsoft 365 email providers will be set automatically; manual configuration is required for all other email providers. For more information, see Configure and send email.

- Document (attachment) history
  - This document management feature creates a history mechanism for record attachments. This allows your organization to maintain an audit of actions related to individual attachments. For example, you can see when an attachment was created, marked for pending deletion, restored, deleted, or moved and who performed the action. The default history retention period is 180 days, however this is configurable on the **Document management parameters** page. For more information, see Configure document management.

- General availability of the Grouping with grids feature
  - For more information, see Grid capabilities.

- New HTML editor control

- Enhanced the Message::AddAction API
  - The Message::AddAction API now surfaces the action in notifications when messages are routed to the

Action center or Message details pane. For more information, see Messaging APIs.

- Mitigate a SQL injection attack
  - Use the new **Statement.executeQueryWithParameters** API to mitigate SQL injection attacks.

Most of these features must be enabled using Feature management before you can use them.

Additional resources

Bug fixes

For information about the bug fixes that are included in this update, sign in to Microsoft Dynamics Lifecycle Services (LCS), and view the KB article.

Dynamics 365: 2020 release wave 2 plan

Wondering about upcoming and recently released capabilities in any of our business apps or platform?

Check out the Dynamics 365: 2020 release wave 2 plan. We've captured all the details, end to end, top to bottom, in a single document that you can use for planning.

Removed and deprecated platform features

The Removed or deprecated platform features topic describes features that have been removed, or that are planned for removal in platform updates of Finance and Operations apps.

- A *removed* feature is no longer available in the product.
- A *deprecated* feature isn't in active development and might be removed in a future update.

A deprecation notice will be added in the Removed or deprecated platform features topic 12 months before the removal of any feature from the product.

For breaking changes that affect only compilation time, but that are binary-compatible with sandbox and production environments, the deprecation time will be less than 12 months. Typically, these changes are functional updates that must be made to the compiler.

Platform updates for version 10.0.15 of Finance and Operations apps (January 2021)

2/18/2021 • 2 minutes to read • Edit Online

This topic lists the features that are included in the platform updates for version 10.0.15 of Finance and Operations apps. This version has a build number of 7.0.5816 and is available on the following schedule:

- **Preview release:** October 2020
- **General availability (self-update):** November 2020
- **Auto-update:** January 2021

Features included in this release

- Azure Pipelines task for asset deployment now supports self-service environments
  - For more information to enable this support, see Deploy assets by using Azure Pipelines.
- Specific full user license information is now available when configuring security and assigning roles to users.
  - For more information, see Stay compliant with user licensing requirements.
- Regression suite automation tool (RSAT) 2.0
  - For RSAT users, version 10.0.15 requires the latest version of RSAT to address a known issue. RSAT 2.0.7031.98 or newer will be available for download at https://www.microsoft.com/en-us/download/details.aspx?id=57357 on Monday, October 12, 2020.

Additional resources

Bug fixes

For information about the bug fixes that are included in this update, sign in to Microsoft Dynamics Lifecycle Services (LCS), and view the KB article.

Dynamics 365: 2020 release wave 2 plan

Wondering about upcoming and recently released capabilities in any of our business apps or platform?

Check out the Dynamics 365: 2020 release wave 2 plan. We've captured all the details, end to end, top to bottom, in a single document that you can use for planning.

Removed and deprecated platform features

The Removed or deprecated platform features topic describes features that have been removed, or that are planned for removal in platform updates of Finance and Operations apps.

- A *removed* feature is no longer available in the product.
- A *deprecated* feature isn't in active development and might be removed in a future update.

A deprecation notice will be added in the Removed or deprecated platform features topic 12 months before the removal of any feature from the product.

For breaking changes that affect only compilation time, but that are binary-compatible with sandbox and production environments, the deprecation time will be less than 12 months. Typically, these changes are functional updates that must be made to the compiler.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Removed or deprecated platform features

2/18/2021 • 10 minutes to read • Edit Online

This topic describes features that have been removed, or that are planned for removal in platform updates of Finance and Operations apps.

- A *removed* feature is no longer available in the product.
- A *deprecated* feature is not in active development and may be removed in a future update.

This list is intended to help you consider these removals and deprecations for your own planning.

Detailed information about objects in Finance and Operations apps can be found in the Technical reference reports. You can compare the different versions of these reports to learn about objects that have changed or been removed in each version of Finance and Operations apps.

Feature removed effective January 28, 2021

Batch job to handle SQL index defragmentation

Reason for deprecation/removal	In order to reduce the overhead of operating, monitoring, and maintaining the index management by customers, this feature has been removed.
Replaced by another feature?	Going forward, the index maintenance will be performed by Microsoft services. This will happen continuously without affecting the user workloads.
Product areas affected	Finance and Operations apps
Deployment option	Cloud deployment - affects Microsoft-managed production environments and Tier 2 through Tier 5 sandbox environments.
Status	This feature is removed.

Platform updates for version 10.0.17 of Finance and Operations apps

> **IMPORTANT**
> Version 10.0.17 is available as part of a preview release. The content and the functionality are subject to change. For more information about preview releases, see One version service updates FAQ.

Visual Studio 2015

Reason for deprecation/removal	To support the latest versions of Visual Studio, some changes have to be made to the X++ extensions for Visual Studio. These changes are incompatible with Visual Studio 2015.

Replaced by another feature?	Visual Studio 2017 will replace Visual Studio 2015 as the deployed and required version.
Product areas affected	Visual Studio development tools
Deployment option	All
Status	Deprecated. Upon updating, the previous X++ tools will be removed from Visual Studio 2015, and the updated tools will not install on Visual Studio 2015. There is no impact on hosted builds. For build virtual machines, the build pipeline (build definition) needs to be manually updated to change the dependency from MSBuild 14.0 (Visual Studio 2015) to MSBuild 15.0 (Visual Studio 2017) as described in Update a legacy pipeline in Azure Pipelines.

User avatar

Reason for deprecation/removal	The user avatar that displays on the right side of the navigation bar was retrieved using an API from the Dynamics 365 header control, which has been deprecated.
Replaced by another feature?	Users will see their initials in a circle in the navigation bar instead. This is the same visual currently used on development machines.
Product areas affected	Web client
Deployment option	All
Status	Removed as of version 10.0.17

Enterprise Portal (EP) deprecation

Reason for deprecation/removal	The metadata artifacts associated with Dynamics AX 2012 Enterprise Portal (EP) have been deprecated, as EP was never supported in the Finance and Operations apps.
Replaced by another feature?	No
Product areas affected	Web client
Deployment option	All
Status	Deprecated. All EP code is scheduled to be removed in the October 2021 release.

Platform updates for version 10.0.15 of Finance and Operations apps

Internet Explorer 11 support for Dynamics 365 is deprecated

Reason for deprecation/removal	Effective December 2020, Microsoft Internet Explorer 11 support for all Dynamics 365 products is deprecated, and Internet Explorer 11 won't be supported after August 2021. This will impact customers who use Dynamics 365 products that are designed to be used through an Internet Explorer 11 interface. After August 2021, Internet Explorer 11 won't be supported for such Dynamics 365 products.
Replaced by another feature?	We recommend that customers transition to Microsoft Edge.
Product areas affected	All Dynamics 365 products
Deployment option	All
Status	Deprecated. Internet Explorer 11 won't be supported after August 2021.

Visual Studio add-in to apply metadata hotfixes

Reason for deprecation/removal	Metadata hotfixes are no longer supported with the One Version service updates that were introduced in July 2018 with version 8.1.
Replaced by another feature?	Individual metadata hotfixes are not available for supported versions. Cumulative quality updates are applied instead.
Product areas affected	Visual Studio add-ins
Deployment option	Development virtual machines
Status	With version 10.0.15, the add-in is no longer included in the Visual Studio tools.

Platform updates for version 10.0.14 of Finance and Operations apps

Online users page

Reason for deprecation/removal	This is a legacy page that was built for previous client/server architecture. The information on this page is not always accurate, which can be confusing and misleading.
Replaced by another feature?	We will provide a new page in a future update.
Product areas affected	System Administration
Deployment option	All
Status	By October 2021 this form will be removed.

Platform updates for version 10.0.13 of Finance and Operations apps

Custom code defined in SSRS report properties

Reason for deprecation/removal	In general, custom code offers limited benefits while at the same time, requires significant resourcing and compute to support. Custom code is primarily used by report authors to call public methods from a custom code assembly. However, the cloud-hosted service does not support references to custom assemblies for SSRS reports.
Replaced by another feature?	Report authors may choose to continue referencing public .NET APIs for Math, Conversion, and Format operations from any textbox expression. For more information, see Add Code to a Report (SSRS).
Product areas affected	Subset of application report designs defined in RDL that contain custom code.
Deployment option	All
Status	With version 10.0.13, the compiler will begin issuing a warning for instances where custom code is detected in a SSRS report definition. To fix the issue, open the report design definition and remove all custom code artifacts. This warning will be replaced with a compiler error in a future update.

Upgrade of three jQuery component libraries

Reason for deprecation/removal	Three jQuery component libraries are being updated for security fixes and to maintain currency.
Replaced by another feature?	The following libraries are being affected: jQuery (to version 3.5.0 from version 2.1.4), jQuery UI (to version 1.12.1 from version 1.11.4), jQuery qTip (to version 3.0.3 from 2.2.1). Migration guidance has been provided online by jQuery.
Product areas affected	Extensible controls, specifically custom JavaScript code utilizing deprecated or removed APIs
Deployment option	All
Status	With version 10.0.13/Platform update 37, customers can optionally move to the latest libraries by enabling the "Upgrade three jQuery component libraries" feature. Moving to the new libraries will be mandatory with the April 2021 release to allow time for migration of affected APIs.

Existing grid control/forceLegacyGrid() API

Reason for deprecation/removal	The existing grid control is being replaced by the new grid control.

Replaced by another feature?	The new grid control
Product areas affected	Web client
Deployment option	All
Status	In version 10.0.13, the new grid control is generally available, and customers can optionally turn on this feature. The new grid control will become mandatory in the October 2021 release. When the new grid control becomes mandatory, the **forceLegacyGrid()** API will no longer be honored.

Personalization without saved views

Reason for deprecation/removal	The personalization subsystem has been overhauled with the saved views feature, so that it has better performance and offers additional capabilities.
Replaced by another feature?	Saved views
Product areas affected	Web client
Deployment option	All
Status	In version 10.0.13/Platform update 37, the saved views feature is generally available, and customers can optionally turn on this feature. The saved views feature will become mandatory in the October 2021 release.

Platform updates for version 10.0.12 of Finance and Operations apps

Grid or group control form extensions containing invalid field references

Reason for deprecation/removal	The data group property on grid or group controls is used to automatically show all the fields of a field group. A grid or group control added by extension could contain fields that are no longer defined on the field group, or it might be missing fields that are defined on the field group. This can cause inconsistent behavior at runtime. Platform updates for version 10.0.12 of Finance and Operations apps now categorize this issue as a compiler *warning*. To fix this issue, open the form extension and save it.
Replaced by another feature?	This compiler warning will be replaced with a compiler error in a future update.
Product areas affected	Visual Studio development tools
Deployment option	All
Status	A compiler warning is introduced in platform updates for version 10.0.12 of Finance and Operations apps.

Platform updates for version 10.0.11 of Finance and Operations apps

Explicit safe lists for self-service environments

Reason for deprecation/removal	The process for moving IP to safe lists has changed. Self-service no longer supports IP safe lists.
Replaced by another feature?	For more information, see Configuring Azure Active Directory Conditional Access.
Product areas affected	Security
Deployment option	Cloud
Status	**Deprecated:** This feature is fully-deprecated for self-service deployments.

Visual Studio 2015

Reason for deprecation/removal	To support the latest versions of Visual Studio, some changes have to be made to the X++ extensions for Visual Studio. These changes are incompatible with Visual Studio 2015.
Replaced by another feature?	Visual Studio 2017 will replace Visual Studio 2015 as the deployed and required version.
Product areas affected	Visual Studio development tools
Deployment option	All
Status	Virtual machines deployed on version 10.0.13 (Platform update 37) or later contain Visual Studio 2017. Version 10.0.16 (Platform update 40) is the final release with support for Visual Studio 2015. Virtual machines with only Visual Studio 2015 will not be able to update to version 10.0.17 (Platform update 41).

Field groups containing invalid field references

Reason for deprecation/removal	Field groups in table metadata definitions can contain field references that aren't valid. If these field groups are deployed, they can cause runtime failures in Financial Reporting and Microsoft SQL Server Reporting Services (SSRS). Platform update 23 introduced a compiler *warning* that enabled this metadata issue to be addressed. Platform updates for version 10.0.11 of Finance and Operations apps categorize this issue as a compiler *error*. To fix this issue, follow these steps. 1. Remove the invalid field reference from the table field group definition. 2. Recompile. 3. Make sure that any errors are addressed.
Replaced by another feature?	This compiler error permanently replaces the compiler warning.
Product areas affected	Visual Studio development tools
Deployment option	All
Status	**Deprecated:** The compiler warning is a compiler error in platform updates for version 10.0.11 of Finance and Operations apps.

ISV licenses created by using the SHA1 hashing algorithm

Reason for deprecation/removal	The process for creating independent software vendor (ISV) licenses has changed. For more information, see Independent software vendor (ISV) licensing.
Replaced by another feature?	Yes. Use Windows PowerShell to create licenses.
Product areas affected	Visual Studio development tools
Deployment option	All
Status	**Deprecated:** ISV licenses that were created by using the SHA1 hashing algorithm. This algorithm depended on certificates that were created by using the MakeCert utility, and this utility has been deprecated. **Deprecated:** The use of SHA1 for security or hashing purposes. SHA1 will cease to function in early 2021. Therefore, it should no longer be used. **Removed:** Support for Transport Layer Security (TLS) 1.0 and TLS 1.1 incoming or outgoing requests.

Platform update 32

Workflow request change dialog box no longer includes user selection drop-down list

Reason for deprecation/removal	This was a security issue because the request for change could be sent to an unintended user. This was also a usability issue because it forced the user to determine who the workflow originator was and manually select them.
Replaced by another feature?	No
Product areas affected	Workflow
Deployment option	All
Status	The user selection drop-down list was removed from the request change dialog box in Platform update 32. Request change requests will be automatically sent to the originator as intended. For more information about this functionality, see Actions in workflow approval processes.

Embedded drill-through links are no longer supported in paginated documents rendered by the cloud-hosted service

Reason for deprecation/removal	Navigation URLs embedded in documents rendered by the service may contain sensitive business data. We are removing support for embedded drill-through links in documents as a security precaution to further protect customer's data. Users will also benefit from improved performance while interactively producing documents as a result of this change.
Replaced by another feature?	No
Product areas affected	Reporting
Deployment option	All
Status	This feature is actively being removed from the service. The modern client offers numerous options for producing views that include auto-generated links to assist in navigating the application. Paginated documents rendered by the service are recommended for external communications that are emailed, archived, and printed for recipients. We have improved the experience for previewing documents directly in the browser, which offers direct access to local printers. For more information, see Preview PDF documents with an embedded viewer.

Previous announcements about removed or deprecated features

To learn more about features that have been removed or deprecated in previous releases, see Removed or deprecated features in previous releases.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Removed or deprecated features in previous releases

2/18/2021 • 51 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

IMPORTANT

This topic is no longer updated. To see a current list of features that have been removed or deprecated from Finance and Operations apps, search for **"Removed or deprecated features"** content that relates to the app you're using.

This topic describes features that have been removed or deprecated from Dynamics 365 for Finance and Operations and previous releases of that product.

- A *removed* feature is no longer available in the product.
- A *deprecated* feature is not in active development and may be removed in a future update.

This list is intended to help you consider these removals and deprecations for your own planning.

Detailed information about objects in Finance and Operations apps can be found in the Technical reference reports. You can compare the different versions of these reports to learn about objects that have changed or been removed in each version of Finance and Operations apps.

Finance 10.0.7 with Platform update 31

Chinese voucher types without Account groups selection

Reason for deprecation/removal	Changed to the feature with account groups selection.
Replaced by another feature?	Yes
Product areas affected	Application
Deployment option	All
Status	Deprecated: By December 1, 2020, we plan to no longer support Chinese voucher types setup without Account groups selection. Find more details about new feature design in What's new in 10.0.7

Finance and Operations 10.0.6 with Platform update 30

DimensionHash.getHash(str _message)

Reason for deprecation/removal	Windows is deprecating the use of SHA1, as documented in Windows Enforcement of SHA1 Certificates.
Replaced by another feature?	Yes
Product areas affected	Application
Deployment option	All
Status	Deprecated: By April 1, 2020, developers must use the platform APIs found in the class **HasFunction**.

Hash.ComputeSHA1Hash(string message)

Reason for deprecation/removal	Windows is deprecating the use of SHA1, as documented in Windows Enforcement of SHA1 Certificates.
Replaced by another feature?	Yes
Product areas affected	Platform
Deployment option	All
Status	Deprecated: By April 1, 2020, developers must use the platform APIs found in the class **HasFunction**.

FormDateTimeControl.setUtcString()

Reason for deprecation/removal	We are retiring the **setUtcString()** method, because a better replacement method is available.
Replaced by another feature?	Yes
Product areas affected	Platform
Deployment option	All
Status	Deprecated: By October 1, 2020, we plan to no longer support the **setUtcString()** method. Developers should be using the **setUtcDateTime()** method instead.

Blacklist report (IT) – Feature reference IT-00001

Reason for deprecation/removal	Not legally required.
Replaced by another feature?	No
Product areas affected	Italian localization
Deployment option	All
Status	Deprecated: By October 1, 2020, we plan to no longer support the **Blacklist report (IT) – Feature reference IT-00001**.

Domestic tax report – Feature reference IT-00003

Reason for deprecation/removal	Not legally required.
Replaced by another feature?	No
Product areas affected	Italian localization
Deployment option	All
Status	Deprecated: By October 1, 2020, we plan to no longer support the **Domestic tax report – Feature reference IT-00003**.

Finance and Operations 10.0.5 with Platform update 29

US Payroll tax updates

Reason for deprecation/removal	We are retiring tax updates for the US Payroll functionality due to low usage and enhanced functionality that is now offered via strategic integrations.
Replaced by another feature?	Yes
Product areas affected	Payroll
Deployment option	All
Status	Deprecated: By July 31, 2024, we plan to no longer provide tax updates to US Payroll customers. The functionality will remain in the product, but enhancements will no longer keep the functionality up to date, and any product defects will be evaluated on a case-by-case basis.

> **NOTE**
>
> This represents a change from the original discontinuation date of October 1, 2021. For more information, see Tax updates being retired for US Payroll feature in Microsoft Dynamics 365 for Finance and Operations.

Data management staging clean up

Reason for deprecation/removal	Does not meet the core requirements that are needed for scheduling periodic cleanup.
Replaced by another feature?	Yes, the Job history cleanup feature is being added to meet the scenarios holistically.
Product areas affected	Data management
Deployment option	All
Status	Deprecated: Target timeframe for the functionality to be removed is December 2020.

Finance and Operations 10.0.4 with Platform update 28

France: FEC Accounting data export in XML

Reason for deprecation/removal	Replaced by TXT format, **French FEC audit file** is available through **General ledger** > **Periodic tasks** > **Data export**.
Replaced by another feature?	Yes
Product areas affected	General ledger
Deployment option	All
Status	Deprecated. Target timeframe for the functionality to be removed is July 2020.

Legacy navigation bar

Reason for deprecation/removal	Header alignment with other Dynamics and Office products. For more details, see Updated navigation bar that aligns with the Office header.
Replaced by another feature?	Starting in Platform update 24, a restyled navigation bar that features search was introduced.
Product areas affected	Web client

Deployment option	All
Status	Deprecated: Starting in April 2020, the legacy navigation bar will no longer be available. Until that point, customers can revert to the legacy navigation bar through the **Client performance options** page.

Finance and Operations 10.0.2 with Platform update 26

Legacy default action behavior

Reason for deprecation/removal	The legacy behavior for default actions in grids results in an unexpected column having the default action link after grid columns have been reordered via personalization. The new sticky default action feature corrects this. For more details, see Sticky default actions in grids.
Replaced by another feature?	Starting in Platform update 21, a feature for "sticky default actions" was introduced. This feature can be enabled on the **Client performance options** page.
Product areas affected	Grids in the web client
Deployment option	All
Status	Deprecated: Starting in April 2020, sticky default actions will be the default behavior, without a mechanism to revert to the legacy behavior.

Legacy "is one of" filtering experience

Reason for deprecation/removal	The "is one of" filtering experience went through a redesign in Platform update 22,with the plan for this to eventually be the only "is one of" filtering experience.
Replaced by another feature?	Starting in Platform update 22, an improved "is one of" filtering experience became available on the **Client performance options** page. For more information, see Optimized is one of filtering experience.
Product areas affected	Web client
Deployment option	All
Status	Deprecated: Starting in April 2020, the improved "is one of" experience will be the default behavior, without a mechanism to revert to the legacy behavior.

Parameter to enable sales orders with multiple project contract funding sources

Support for creating project-based sales orders where the project contract has multiple funding sources is

enabled with the **Project management parameters** setting **Allow sales orders for project with multiple funding sources**. By default, this parameter is not enabled.

Reason for deprecation/removal	The functionality will always be enabled after the parameter is removed.
Replaced by another feature?	No. The functionality to support project-based sales orders with multiple funding sources will always be enabled.
Product areas affected	The **Allow sales orders for projects with multiple funding sources** parameter will be removed. The following methods will be modified when the parameter is removed: **ctrlSalesOrderTable** method in **ProjStatusType** class, **validate** method for **ProjId** field, and **run** method in **SalescreateOrder** form. The following methods will be deprecated when the parameter is removed: **IsSalesOrderAllowedForMultipleFundingSources** in **ProjTable** table file, **IsAllowSalesOrdersForMultipleFundingSourcesParamEnabled** method in **ProjTable** table file, **AllowSalesOrdersForMultipleFundingSources** data field in **ProjParameters** form and **ProjParameterEntity** files, **IsAssociatedToMultipleFundingSourcesContract** private method in **ProjTable** table file.
Deployment option	All
Status	Deprecation is planned for the April 2020 release wave.

Legacy workflow reports for tracking and instance status

Reason for deprecation/removal	The legacy workflow reports for tracking and instance status are being deprecated because they are no longer referenced from the navigation. The report names are WorkflowWorkflowInstanceByStatusReport and WorkflowWorkflowTrackingReport.
Replaced by another feature?	The workflow history form can be used instead.
Product areas affected	Web client
Deployment option	All
Status	Deprecated: Target timeframe for the functionality to be removed is April 2020.

Finance and Operations 10.0.1 with Platform update 25

Deprecated APIs and potential breaking changes

Deriving from internal classes is deprecated

Reason for deprecation/removal	Before Platform update 25, it was possible to create a class or table that derives from an internal class/table that is defined in another package/module. This is not a safe coding practice. As of Platform update 25, the compiler will display a warning.
Replaced by another feature?	The compiler warning will be replaced by an error in Platform update 26. This change is backward compatible at runtime, which means that Platform update 25 or newer can be deployed on any sandbox or production environment without the need to modify custom code. This change only affects development and compile time.
Product areas affected	Visual Studio development tools
Deployment option	All
Status	Deprecated: The warning will become a compilation error in Platform update 26.

Overriding internal methods is deprecated

Reason for deprecation/removal	Before Platform update 25, it was possible to override an internal method in a derived class that is defined in another package/module. This is not a safe coding practice. As of Platform update 25, the compiler will display a warning.
Replaced by another feature?	This warning will be replaced by a compile error in Platform update 26. This change is backward compatible at runtime, which means that Platform update 25 or newer can be deployed on any sandbox or production environment without the need to modify custom code. This change only affects development and compile time.
Product areas affected	Visual Studio development tools
Deployment option	All
Status	Deprecated: The warning will become a compilation error in Platform update 26.

Finance and Operations 10.0.0 with Platform update 24

Renaming released products

Reason for deprecation/removal	When you use the **Rename primary key** function to change the ItemId of a released product, only direct foreign key references are updated. Any other references to the released product, such as from production orders, will retain the old ItemId. As a result, there could be inconsistent data that will eventually block business processes.

Replaced by another feature?	No.
Product areas affected	Product information management
Deployment option	All
Status	Removed as of Finance and Operations 10.0.0 with Platform update 24.

Finance and Operations 8.1.3 with Platform update 23

SQL Server Reporting Services ReportViewer Control

Customers can use the **Export** action provided by the embedded SQL Server Reporting Services (SSRS) ReportViewer control to download documents produced by Finance and Operations applications. This HTML-based presentation of the report offers users a non-paginated preview of the document.

Reason for deprecation/removal	The non-paginated nature of the HTML-based preview experience does **not** deliver fidelity with the physical documents ultimately produced by Finance and Operations. By fully embracing PDF as the standard format for business documents, users are able to take advantage of a modern viewing experience with improved performance when producing application reports.
Replaced by another feature?	Going forward, PDF documents will be the default format for reports rendered by Finance and Operations.
Product areas affected	This change does **not** impact customer scenarios where reports are distributed electronically or sent directly to printers.
Deployment option	All
Status	Deprecated: A removal date has not been set for this feature. The functionality to automatically preview application reports using an embedded PDF viewer is planned for the May 2019 Platform update.

Client KPI controls

Embedded key performance indicators (KPIs) could be modeled in Visual Studio by a developer and further customized by the end user.

Reason for deprecation/removal	The native client controls used to define KPIs have low customer uptake and rely on a developer to add trackable metrics.

Replaced by another feature?	PowerBI.com service delivers world-class tooling for defining and managing KPIs based on data from external sources. In an upcoming release, we plan to enable you to embed solutions hosted on PowerBI.com in application workspaces.
Product areas affected	This update will prevent developers from introducing new KPI controls in Visual Studio designer.
Deployment option	All
Status	Deprecated: A removal date has not been set for this feature.

Deprecated APIs and future breaking changes

Field groups containing invalid field references

Reason for deprecation/removal	It is possible for table metadata definitions to have field groups containing invalid field references. If deployed, this can cause runtime failures in Financial Reporting and SQL Server Reporting Services (SSRS). This issue is currently categorized as a *compiler warning* rather than an *error*, meaning that the deployable package creation and deployment can proceed without fixing the issue. To fix this issue: 1. Remove the invalid field reference from the table field group definition. 2. Recompile. 3. Ensure any warnings or errors are addressed.
Replaced by another feature?	This warning will be replaced by a compile error in the future.
Product areas affected	Visual Studio development tools
Deployment option	All
Status	Deprecated: The warning is a compile-time error with platform updates for version 10.0.11 of Finance and Operations apps.

Complete list

To access the full list of APIs that are being deprecated, see Deprecation of methods and metadata elements.

Finance and Operations 8.1 with Platform update 20

Batch transfer rules for subledger journal account entries

The Synchronous transfer mode is being deprecated in the General ledger parameters. This mode is replaced by Asynchronous and scheduled batch only, which already exist as options for transfer. For additional information, see the General Ledger Parameters – Batch transfer rules blog.

Reason for deprecation/removal	We are removing the synchronous option due to performance impact to the system.
Replaced by another feature?	Asynchronous and scheduled batch are options to use in place of Synchronous.
Product areas affected	General Ledger, Accounts payable, Accounts Receivable, Procurement, Expense
Deployment option	All
Status	Deprecated: Target timeframe for the functionality to be removed is the 10.0 version.

Electronic reporting for Russia

Feature for configuring .txt and .xml file formats of declarations.

Reason for deprecation/removal	Replaced with Electronic reporting.
Replaced by another feature?	Yes.
Product areas affected	General Ledger
Deployment option	All
Status	Removed as of Finance and Operations 8.1 with Platform update 20.

Financial reports generator for Russia

A tool for setting up data collection for accounting and tax reports, and to export data to XLS and DOC report templates. Functional parts: Export data to XLS and DOC report templates, queries, fixed requisites are removed.

Reason for deprecation/removal	Removed parts are replaced with Electronic reporting.
Replaced by another feature?	Yes. Financial reports setup user interface should be used for setting up data collection rules by GL accounts or tax registers. Export data to various file types, fixed requisites and query-like data collection rules should be configured in Electronic reporting.
Product areas affected	General ledger.
Deployment option	All
Status	Removed as of Finance and Operations 8.1 with Platform update 20.

Integration with external providers for sending electronic reporting through communication channels for

Russia

Feature exporting generated electronic files of declarations to folder for further sending to official providers of electronic reporting as well as importing state back.

Reason for deprecation/removal	Replaced with electronic messages configurable feature.
Replaced by another feature?	Yes.
Product areas affected	General Ledger, Tax
Deployment option	All
Status	Removed as of Finance and Operations 8.1 with Platform update 20.

Profit tax register wizard

Feature for creating templates for new profit tax registers. This feature creates X++ objects for new registers, which are then created as templates with the appropriate calculation logic added in.

Reason for deprecation/removal	Feature is not compatible with the Finance and Operations extensibility model.
Replaced by another feature?	No
Product areas affected	Tax
Deployment option	All
Status	Removed as of Finance and Operations 8.1 with Platform update 20.

Finance and Operations 8.0 with Platform update 15

No features have been removed or deprecated with this release. Platform update 15 is cumulative and contains new or changed features from Platform update 13, Platform update 14, and Platform update 15.

Finance and Operations, Enterprise edition 7.3 with Platform update 12

Personalized product recommendations

Starting February 15, 2018, retailers will no longer be able to display personalized product recommendations on a point of sale (POS) device. For more information, see Product recommendations overview.

Reason for deprecation/removal	We are removing the current version of the product recommendation service as we redesign this feature with a better algorithm and newer retail-oriented capabilities.

Replaced by another feature?	No. However, after Spring 2018, we plan to bring back this feature to leverage a new recommendation service.
Product areas affected	Personalized product recommendations in POS.
Deployment option	All
Status	Removed as of February 15, 2018. This affects customers running Dynamics 365 for Operations 1611 and later.

Extension of the list of Electronic reporting (ER) functions

The possibility to introduce custom functions to be used in the ER expression builder (for more information, see Extend the list of Electronic reporting (ER) functions) is not supported any more. Due to changes of the ER APIs, the API to call built-in functions from the ER expression builder became internal and can't be extended any longer.

Reason for deprecation/removal	Code sealing initiative
Replaced by another feature?	None. Whenever a new built-in function is needed, a new extension request must be addressed to the ER framework team. As a temporary work around while the requested function is under development by the ER team, the required logic can be programmed as a method of a custom application class. This method can be accessed in an ER expression as a property of the added ER data source of the **Application\Class** type that refers to that custom application class.
Product areas affected	Electronic reporting framework
Deployment option	All
Status	Removed as of Finance and Operations, Enterprise edition 7.3.

Inventory by item group and Inventory by inventory dimension aging reports

These two reports are no longer supported in Finance and Operations. Instead, the **Inventory aging** report can be used to improve the user experience.

Reason for deprecation	Duplicate functionality
Replaced by another feature?	Yes. The two reports have been replaced by the **Inventory aging** report.
Product areas affected	Inventory management, Cost management

Deployment option	All
Status	Deprecated: The menu items for the two reports have been removed in version 7.3. However, the code for the reports remains in the product. The plan is to remove the code in a future release.

Power BI content packs available on AppSource

The **Cost management**, **Financial performance**, and **Retail channel performance** content packs, available on the Microsoft AppSource site, are deprecated as a consequence of product updates in Microsoft Power BI. System administration forms used to deploy these content packs to PowerBI.com are also being deprecated in Finance and Operations.

Reason for deprecation/removal	Product updates in Microsoft Power BI.
Replaced by another feature?	The **Cost management**, **Financial performance**, and **Retail channel performance** content packs, available on the AppSource site, are being replaced by analytical applications which allow for solution integrations at the database level. For more information about analytical applications, see Embedded Power BI in workspaces.
Product areas affected	Cost management, Finance, and Retail
Deployment option	Cloud only (Integration with PowerBI.com is not supported in on-premises deployments.)
Status	Deprecated: Target timeframe for the functionality removal is Q2 2018.

Standard UI in data management workspace

The standard UI in data management is the legacy UI, which is the default UI presented to the users when they visit the data management workspace.

Reason for deprecation/removal	We are investing in providing new user experiences in the new UI.
Replaced by another feature?	The new UI called *Enhanced views* is replacing the old UI.
Product areas affected	Data management workspace
Deployment option	All
Status	Deprecated: Target timeframe for the functionality to be removed is Q2 2018.

Excise, Sales Tax, Service Tax for India

These taxes have been subsumed into Indian GST.

Reason for removal or deprecation	These taxes have been subsumed into Indian GST.
Replaced by another feature?	Indian GST
Product areas affected	Tax
Deployment option	All modules
Status	Deprecated: A removal date has not been set for this feature.

File Validation Utility (FVU) for India

Reason for removal or deprecation	Lack of customer usage
Replaced by another feature?	No
Product areas affected	Indian withholding tax
Deployment option	All modules
Status	Deprecated: A removal date has not been set for this feature.

TDS/TCS certificate for India

Users can download this from the government portal.

Reason for removal or deprecation	Lack of customer usage
Replaced by another feature?	No
Product areas affected	Indian withholding tax
Deployment option	All modules
Status	Deprecated: A removal date has not been set for this feature.

Export/import (EXIM) incentive scheme for India

Reason for removal or deprecation	Lack of customer usage
Replaced by another feature?	No
Product areas affected	Import and export

Deployment option	All modules
Status	Deprecated: A removal date has not been set for this feature.

Dynamics 365 for Retail 7.2

Personalized product recommendations

Starting February 15, 2018, retailers will no longer be able to display personalized product recommendations on a point of sale (POS) device. For more information, see Product recommendations overview.

Reason for deprecation/removal	We are removing the current version of the product recommendation service as we redesign this feature with a better algorithm and newer retail-oriented capabilities.
Replaced by another feature?	No. However, after Spring 2018, we plan to bring back this feature to leverage a new recommendation service.
Product areas affected	Personalized product recommendations in POS.
Deployment option	All
Status	Removed as of February 15, 2018. This affects customers running Dynamics 365 for Retail 7.2 and later.

Finance and Operations, Enterprise edition July 2017 with Platform update 8

Currency conversion for accounting and reporting currencies

Currency conversion for accounting and reporting currencies was introduced when the euro was introduced.

Reason for deprecation/removal	Limited usage and addition of the Copy legal entity functionality as a replacement.
Replaced by another feature?	No, but the Copy legal entity and Configurations features were added to make it easier to move to a company that has changing core requirements.
Product areas affected	Financial management
Status	Deprecated: A removal date has not been set for this feature.

Warehouse mobile devices portal

Warehouse mobile devices portal (WMDP) was a standalone component that was intended for on-premises self-deployment. This component is no longer supported in Finance and Operations. A native app that improves the user experience has replaced the functionality of WMDP.

Reason for deprecation/removal	Duplicate functionality.
Replaced by another feature?	Yes. This feature has been replaced by Finance and Operations - Warehousing. For more information about setup and prerequisites, see Install and configure the Warehousing app overview.
Product areas affected	Warehouse management, Transportation management
Deployment option	Warehouse mobile devices portal (WMDP) was a standalone component that was intended for on-premises self-deployment.
Status	Deprecated: Target timeframe for the functionality to be removed is Q4 2019.

Advanced bank reconciliation matching rule for manual matching

A matching rule was used to select and mark a bank document when documents were manually matched in the reconciliation worksheet.

Reason for deprecation/removal	Limited usage.
Replaced by another feature?	No. Column filtering capabilities should be used to find documents for reconciliation.
Product areas affected	Cash and bank management
Deployment option	All
Status	Removed as of July 2017.

Dynamics 365 for Operations 1611 with Platform update 3

AEB payment formats for Spain

The Consejo Superior Bancario payment formats were used to send remittance files to the bank for customer payments and vendor payments. The content of these formats was determined by the Asociación Española de Banca. It covers Cuaderno 19, 32, 58, 34.

Reason for deprecation/removal	The payment formats are no longer used.
Replaced by another feature?	Yes, ISO20022 Credit transfer and Direct debit payment formats for Spain
Product areas affected	Accounts payable, Accounts receivable

Status	Deprecated: A removal date has not been set for this feature.

Bank payments transfer for Lithuania

Bank payment transfers were generated and printed by using the Payment transfer (LT) export format for Lithuania. The Lithuanian market began to use LITAS, the unified electronic banking system, in 2005.

Reason for deprecation/removal	The payment formats are no longer used.
Replaced by another feature?	Yes, ISO20022 Credit transfer payment format for Lithuania
Product areas affected	Accounts payable
Status	Deprecated: A removal date has not been set for this feature.

BBS Direkte Remittering payment formats for Norway

BBS Direkte Remittering payment formats include customer payment collection export (direct debit) and return message import.

Reason for deprecation/removal	The payment formats are no longer used.
Replaced by another feature?	The AvtaleGiro customer payment format for Norway can be used to generate direct debit messages. Return message import will be implemented in future releases.
Product areas affected	Accounts payable, Accounts receivable
Status	Deprecated: A removal date has not been set for this feature.

Chart of Accounts tool for Spain

This tool is used when a chart of accounts in Spain requires major changes. Users can import a new chart of accounts in Microsoft Excel or text format, and can also import financial statements.

Reason for deprecation/removal	Limited usage
Replaced by another feature?	No
Product areas affected	General ledger
Status	Deprecated: A removal date has not been set for this feature.

Dom80 payment format for Belgium

Legacy Belgian payment format for payment collection (direct debit).

Reason for deprecation/removal	The payment format is no longer used.
Replaced by another feature?	Yes, ISO 20022 Direct debit payment format for Belgium
Product areas affected	Accounts receivable
Status	Deprecated: A removal date has not been set for this feature.

DTA/EZAG payment formats for Switzerland

DTA/EZAG formats are integrated into the ESR system, because they can carry on the reference number. Because the reference number isn't mandatory, these formats can be used to process any vendor payments. These formats are used by companies that have a bank account in a location other than "Postfinance."

Reason for deprecation/removal	The payment formats are no longer used.
Replaced by another feature?	Yes, ISO20022 Credit transfer payment format for Switzerland
Product areas affected	Accounts payable
Status	Deprecated: A removal date has not been set for this feature.

EDIFACT-DIRDEB payment format for Austria

EDIFACT-DIRDEB payment format for payment collection (direct debit).

Reason for deprecation/removal	The payment format is no longer used.
Replaced by another feature?	Yes, ISO 20022 Direct debit payment format for Austria
Product areas affected	Accounts receivable
Status	Deprecated: A removal date has not been set for this feature.

EDIVAT for Belgium

EDIVAT is an obsolete Belgian standard for electronic declaration via secure mail. Dynamics AX 2012 retains the read-only solution to enable access to the historical data.

Reason for deprecation/removal	The functionality is no longer used.
Replaced by another feature?	No

Product areas affected	General ledger
Status	Deprecated: A removal date has not been set for this feature.

eGiro EDIFACT CREMUL payment import format for Norway

eGiro is based on the international UN EDIFACT CREMUL (Multiple Credit Advice Message) standard that is used for automatic posting of customer payments. In Dynamics AX, eGiro is implemented as a customer payment import format.

Reason for deprecation/removal	The payment format is no longer used.
Replaced by another feature?	Yes, the ISO20022 Camt.054 notification import.
Product areas affected	Accounts receivable
Status	Deprecated: A removal date has not been set for this feature.

External inventory for Poland

Evidence of goods that are taken from a vendor for sales without purchase. Goods that are handled in external inventory don't affect standard inventory, and can be sold and then purchased automatically. This process creates real inventory movements.

Reason for deprecation/removal	Replaced by another feature
Replaced by another feature?	Yes, the core Inbound consignment functionality
Product areas affected	Accounts payable, Inventory management
Status	Deprecated: A removal date has not been set for this feature.

Financial reports generator for Eastern Europe

A tool is used to set up data collection for accounting and tax reports, and to export data to XLS and DOC report templates.

Reason for deprecation/removal	Limited usage
Replaced by another feature?	No. The tool will be replaced by Electronic reporting configurations in future releases.
Product areas affected	General Ledger

Status	Deprecated: A removal date has not been set for this feature.

Import of customer payment transactions for Finland

You can select an import format for Finnish payments to import customer payment transactions from an external file that the bank provides.

Reason for deprecation/removal	The payment format is no longer used.
Replaced by another feature?	Yes, the ISO20022 Camt.054 notification import.
Product areas affected	Accounts receivable
Status	Deprecated: A removal date has not been set for this feature.

Import of payment transactions into a general ledger journal for Finland

A format that is specific to Finland is used to import accounting transactions into the general ledger.

Reason for deprecation/removal	The payment format is no longer used.
Replaced by another feature?	Yes, the ISO20022 Camt.053 bank statement import using Advanced Bank Reconciliation.
Product areas affected	Accounts receivable
Status	Deprecated: A removal date has not been set for this feature.

Integration with Isabel synchronized (CIS) for Belgium

Isabel is the framework for electronic banking in Europe and is a de-facto standard in Belgium.

Reason for deprecation/removal	Integration with Isabel client has been discontinued.
Replaced by another feature?	No. The payment formats that are no longer used are replaced by ISO20022 Credit transfer payment format for Belgium.
Product areas affected	Accounts payable
Status	Deprecated: A removal date has not been set for this feature.

Modifications in the chart of accounts and accounting rules for Spain

This feature is used for changes in the chart of accounts and accounting rules in Spain. It maps accounts to help

transform the old chart of accounts into the new chart of accounts, and compares the previous fiscal year with the new fiscal year, even if they were posted to different account numbers.

Reason for deprecation/removal	Limited usage
Replaced by another feature?	No
Product areas affected	General ledger
Status	Deprecated: A removal date has not been set for this feature.

Pagamento Fornittori vendor payment format

Legacy Italian payment format for credit transfers.

Reason for deprecation/removal	The payment format is no longer used.
Replaced by another feature?	Yes, ISO20022 Credit transfer payment format for Italy
Product areas affected	Accounts payable
Status	Deprecated: A removal date has not been set for this feature.

Payment export formats for Estonia

The Telehansa and Teleservice formats are used for bank payment export.

Reason for deprecation/removal	The payment formats are no longer used.
Replaced by another feature?	Yes, ISO20022 Credit transfer payment format for Estonia
Product areas affected	Accounts payable
Status	Deprecated: A removal date has not been set for this feature.

Payment file archive for Norway

When payment files are generated, the file archive automatically archives all files that are created, even files that were previously written or read.

Reason for deprecation/removal	Replaced by another feature
Replaced by another feature?	Yes, Electronic reporting archived jobs

Product areas affected	Accounts payable, Accounts receivable, Organization administration
Status	Deprecated: A removal date has not been set for this feature.

Payment import formats for Estonia

The Telehansa and TeleTeenus formats are used for bank payment import.

Reason for deprecation/removal	The payment formats are no longer used.
Replaced by another feature?	Yes, the ISO20022 Camt.054 bank notification import.
Product areas affected	Accounts receivable
Status	Deprecated: A removal date has not been set for this feature.

Payroll information in Human Resources

Human Resources Payroll information

Reason for deprecation/removal	This functionality has been replaced by core Payroll and Human Resources pages.
Replaced by another feature?	**Benefits**, **Earnings**, and other related pages that were previously in US Payroll have been reconfigured, and are now part of the core Human Resources configuration to help support external payroll processing. This functionality is accessed by using the **Human Resources 1** > **Payroll** configuration key.
Product areas affected	Human Resources, Payroll
Status	Removed as of Dynamics 365 for Operations version 1611.

Performance management goal workflow

Performance management includes goal management and integration with performance reviews.

Reason for deprecation/removal	Performance management was redesigned, and the number of goal pages was reduced to simplify the process.
Replaced by another feature?	No. Goals are visible to managers through the Manager Self Service portal, and can be changed and viewed by the manager.
Product areas affected	Human capital management

Status	Removed as of Dynamics 365 for Operations version 1611.

Postgirot and Postgirot Utland payment formats for Sweden

Postgirot and Postgirot Utland payment formats for Sweden.

Reason for deprecation/removal	The payment formats are no longer used.
Replaced by another feature?	Yes, ISO20022 Credit transfer payment format for Sweden
Product areas affected	Accounts payable
Status	Deprecated: A removal date has not been set for this feature.

Radio frequency identifier

Radio Frequency Identification (RFID) is a data-collection technology that uses electronic tags to store identification data and a no-line-of-sight requirement reader to capture the identification data.

Reason for deprecation/removal	Low customer usage and a limited feature set.
Replaced by another feature?	No
Product areas affected	Inventory management
Status	Removed as of Dynamics 365 for Operations 1611.

Report about state invoices numbering for Latvia

Latvian legislation provides specific rules about the numbering of sales invoices. The functionality lets you assign specific numbers to sales invoices, based on the user or user group. You can then generate a report or an XML file. You can also print a report about invoice numbers that are used.

Reason for deprecation/removal	The state invoice numbering no longer has to be maintained. The report about used invoice numbers is no longer required.
Replaced by another feature?	No
Product areas affected	Accounts receivable
Status	Deprecated: A removal date has not been set for this feature.

Set up the names of the manager and general accountant of a company for Lithuania

The names of the manager and the general accountant of a company can be specified in the company

information and used in different local report printouts.

Reason for deprecation/removal	Replaced by another feature
Replaced by another feature?	Yes, the setup of officials can be used for the same purpose.
Product areas affected	Accounts payable, Accounts receivable, Cash and bank management
Status	Deprecated: A removal date has not been set for this feature.

Shipping carrier interface

Reason for deprecation/removal	Duplicate functionality
Replaced by another feature?	Partially replaced by Transportation management
Product areas affected	Sales and marketing, Inventory management
Status	Removed as of Dynamics 365 for Operations version 1611.

Telepay payment formats for Norway

Telepay payment formats include vendor payment export (credit transfer) and customer payment collection (direct debit).

Reason for deprecation/removal	The payment formats are no longer used.
Replaced by another feature?	Yes, ISO20022 Credit transfer payment format and AvtaleGiro customer payment format for Norway, as well as pain.002 and camt.054 bank notification return files import.
Product areas affected	Accounts payable, Accounts receivable
Status	Deprecated: A removal date has not been set for this feature.

Vendor payment export formats for Finland

Two formats for exporting payments are available for Finland. LM02 (FI) is used for domestic payments, and LUM2 (FI) is used for foreign payments.

Reason for deprecation/removal	The payment formats are no longer used.
Replaced by another feature?	Yes, ISO20022 Credit transfer payment format for Finland

Product areas affected	Accounts payable
Status	Deprecated: A removal date has not been set for this feature.

Warehouse management II

Reason for deprecation/removal	The Warehouse management II solution (WMS II) that was available in the **Inventory management** module duplicates functionality that is in the **Warehouse management** module that was released in Dynamics AX 2012 R3.
Replaced by another feature?	The **Warehouse management** module that was released in AX 2012 R3, Dynamics AX 2012 R3 CU8, and Dynamics AX 2012 R3 CU9 replaces the Warehouse management II features. The new module has more advanced features and more flexible warehouse management processes than Warehouse management II.
Product areas affected	Inventory management, Sales and marketing, Procurement and sourcing
Status	Removed as of Dynamics 365 for Operations version 1611.

Worker reminders in Human Resources

Human Resources Payroll information

Reason for deprecation/removal	Low usage
Replaced by another feature?	No
Product areas affected	Human resources
Status	Removed as of Dynamics 365 for Operations version 1611

Workflow for creating goals

A workflow for managing the creation of employee goals is one of several workflows that were available to help coordinate the performance management process.

Reason for deprecation/removal	Performance management has been completely redesigned in Finance and Operations.

Replaced by another feature?	The redesigned Performance management feature gives more control over the content of the goals, the measurements that are used to track progress, and the attachment of supporting documentation. Goals can be stored as templates and then reused. This feature can help you set up additional goals for your employees more quickly.
Product areas affected	Human capital management
Status	Removed as of Dynamics 365 for Operations version 1611.

Dynamics AX 7.0

Ability to cancel changes to a vendor invoice

Reason for deprecation/removal	Performance enhancement
Replaced by another feature?	No
Product areas affected	Accounts payable
Status	Removed as of Dynamics AX 7.0.

AIF, AxD, and AxBC integrations

In Application Integration Framework (AIF), data can be exchanged with external systems through business logic that is exposed as services. Dynamics AX includes services that are based on documents and .NET Business Connector (AxBC). A document is created by using XML. The XML includes header information that is added to create a *message* that can be transferred into or out of Dynamics AX. Examples of documents include sales orders and purchase orders. However, almost any entity, such as a customer, can be represented by a document. Services that are based on documents use the **Axd <Document>** classes.

Reason for deprecation/removal	The architecture of AIF and AxDs could not be scaled to a cloud service. There were performance issues around bulk import.
Replaced by another feature?	This feature is replaced by the Data Import/Export framework, which supports recurring bulk import/export. For AxBC, we recommend that you use the actual tables.
Product areas affected	AxDs, AxBCs, and AIF
Status	Removed as of Dynamics AX 7.0.

Billing code rate scripts

Billing scripts were used to calculate billing rates for billing codes. This scripts required custom development in the C Sharp or Visual Basic programming language. In the current version of Dynamics AX, the **billing code rate scripts** are not supported.

Reason for deprecation/removal	The support for the custom C Sharp or Visual Basic scripts was not added in Dynamics AX 7.0.
Replaced by another feature?	No
Product areas affected	Public sector, Accounts receivable
Status	Removed as of Dynamics AX 7.0.

BOMs without BOM versions

When the **BOM versions** configuration key was disabled, bill of materials (BOM) versions were hidden in all forms, and the system forced a 1:1 relationship between released products and BOMs. In the current version of Dynamics AX, the **BOM versions** configuration key can't be disabled.

Reason for deprecation/removal	Using a configuration key to control BOM versions doesn't scale in a cloud environment.
Replaced by another feature?	No
Product areas affected	Product information management, Inventory management
Status	Removed as of Dynamics AX 7.0.

Brazilian Bordero

Specific method of payment for Brazilian companies

Reason for deprecation/removal	Support for the Brazilian Bordero method of payment has been discontinued from Brazilian localization
Replaced by another feature?	No
Product areas affected	Accounts payable
Status	Deprecated: A removal date has not been set for this feature.

Brazilian Sintegra statement

Federal tax statement for ICMS tax

Reason for deprecation/removal	This statement is no longer applicable in some Brazilian states.
Replaced by another feature?	No. Users can use Generic Electronic reporting tool to configure the statement if required under specific situations.

Product areas affected	Fiscal books
Status	Deprecated: A removal date has not been set for this feature.

Brazilian SCAN contingency mode for NF-e

(SCAN) contingency environment is used to generate, export, and import the status of a Nota Fiscal eletrônica (NF-e) when the environment of Secretaria da Fazenda (SEFAZ) is not available.

Reason for deprecation/removal	This method of contingency is no longer applicable in all Brazilian states
Replaced by another feature?	No
Product areas affected	Accounts receivable
Status	Deprecated: A removal date has not been set for this feature.

Business Analyzer

This mobile application let users review key business metrics.

Reason for deprecation/removal	This functionality has been replaced by another feature.
Replaced by another feature?	The Monitor financial performance content pack for Microsoft Power BI will include key financial metrics that were previously available in Business Analyzer.
Product areas affected	General ledger
Status	Deprecated: The use of Business Analyzer has been deprecated.

Business statistics

The setup of business statistics inquiries that can help you analyze the performance of the organization

Reason for deprecation/removal	Legacy approach to business intelligence (BI), low customer usage, and a limited feature set
Replaced by another feature?	New BI solutions for the current version of Dynamics AX
Product areas affected	Procurement and sourcing, Accounts payable, Sales and marketing, Accounts receivable
Status	Removed as of Dynamics AX 7.0.

Change document date function in Invoice approval journal

Reason for deprecation/removal	Low usage
Replaced by another feature?	Yes. The document date on the posted vendor transaction can be changed.
Product areas affected	Accounts payable
Status	Removed as of Dynamics AX 7.0.

ClieOp03 payment format for the Netherlands

Reason for deprecation/removal	The format is no longer applicable in the Netherlands, because it has been replaced by SEPA functionality.
Replaced by another feature?	SEPA payments export
Product areas affected	All modules
Status	Deprecated: A removal date has not been set for this feature.

Compliance Center

The Compliance Center was an Enterprise Portal site for managing the documentation requirements for compliance initiatives that are related to the Sarbanes-Oxley law.

Reason for deprecation/removal	Lack of customer usage. Microsoft SharePoint includes the same capability that was available in the Compliance Center.
Replaced by another feature?	No
Product areas affected	Compliance and internal controls
Status	Removed as of Dynamics AX 7.0.

Connector for Microsoft Dynamics

This tool was used to integrate key data from Microsoft Dynamics CRM to Dynamics ERP applications.

Reason for deprecation/removal	This functionality has been replaced by another feature.
Replaced by another feature?	Dataverse
Product areas affected	Connector for Dynamics

Status	Removed as of Dynamics AX 7.0.

Container unit and multi dimension on-hand

Reason for deprecation/removal	Duplicate functionality
Replaced by another feature?	Yes. Since AX 2012, this functionality has been replaced by the consolidated batch orders feature set. This feature set includes the consolidated on-hand view.
Product areas affected	Product information management, Production control, Inventory management, Sales and marketing
Status	Removed as of Dynamics AX 7.0.

Cue group metadata

Reason for deprecation/removal	Cue groups were used to display one or more Cues in the FactBox area. There was limited uptake, and there were also performance concerns, because a record change in a parent form caused one query per Cue in the Cue group.
Replaced by another feature?	No
Product areas affected	All modules
Status	Removed as of Dynamics AX 7.0.

Cue metadata

Reason for deprecation/removal	Cue metadata was limited to count or sum information.
Replaced by another feature?	Tile metadata was introduced to provide more flexibility for modeling. For example, you can model current counts, navigation, and key performance indicators (KPIs). Count tile metadata is the direct replacement of the Cue metadata.
Product areas affected	All modules
Status	Removed as of Dynamics AX 7.0

Danish check format

Reason for deprecation/removal	Support for the Danish check format layout has been discontinued, and the report has been removed from DK localization.
Replaced by another feature?	No
Product areas affected	All modules
Status	Deprecated: A removal date has not been set for this feature.

Data partitions

Data partitions provide a logical separation of data in the Dynamics AX database.

Reason for deprecation/removal	Data partitions were introduced in Dynamics AX 2012 R2 to enable data isolation. In a common scenario, a company has subsidiaries, and the data from one subsidiary should not be visible to another subsidiary, even though both subsidiaries are managed by the same IT department. However, extra scripts and management overhead throughout the program were required in order to create new partitions and populate them with data, and to back up partition data. In the cloud, where we have access to platform as a service (PaaS) database services (Microsoft Azure SQL Database), it's much more efficient to use a database as the isolation container than to do isolation in the program. Regardless of whether data partitioning is required for subsidiaries, for multiple tenants, or just for scale, we believe that the scenarios can be handled better through multiple instances of Finance and Operations.
Replaced by another feature?	Customers using data partitions must use multiple instances of Finance and Operations if database level separation is a critical issue.
Product areas affected	All modules
Status	Removed as of Dynamics AX 7.0.

Database and file share storage for attachments

Dynamics AX 2012 allowed storage of attachments in the database and in file shares. Both of those options are no longer supported.

Reason for deprecation/removal	Files share storage is no longer supported because cloud-hosted environments cannot communicate with local file shares. Database storage has been deprecated in favor of Azure Blob storage. Azure Blob storage is equivalent to storage in the database, as documents can only be accessed through Finance and Operations client forms. This provides the added benefit of providing storage that doesn't negatively affect the performance of the database. Blob storage is the default storage mechanism for Document Management and works immediately.
Replaced by another feature?	Database storage has been deprecated in favor of Azure Blob storage.
Product areas affected	All modules
Status	Removed as of Dynamics AX 7.0.

Delimitation

Reason for deprecation/removal	No use of the functionality was found.
Replaced by another feature?	No
Product areas affected	Time and attendance
Status	Removed as of Dynamics AX 7.0.

Desktop client

Reason for deprecation/removal	The Dynamics AX client experience has been redesigned to improve usability across multiple platforms and devices.
Replaced by another feature?	The new web client is based on the desktop Form metadata and programming model that have been modified to provide a rich web platform.
Product areas affected	All modules
Status	Removed as of Dynamics AX 7.0.

Direct database connection

In Dynamics AX 2012 R3, Retail Modern POS could connect directly to the Channel DB in similar fashion to Enterprise POS. This was in addition to the standard communication method of Retail Modern POS communicating through Retail Server.

Reason for deprecation/removal	Direct database connectivity required lower security protocols and was primarily used to achieve the highest levels of performance. Due to the performance and security enhancements that have occurred in Finance and Operations, this functionality now causes more issues than it solves.
Replaced by another feature?	No. Only standard Retail Server communication is now supported.
Product areas affected	Channel DB/Retail Modern POS
Status	Removed as of Dynamics AX 7.0.

Dutch SWIFT MT940

Reason for deprecation/removal	Generic functionality is now used instead of localized functionality.
Replaced by another feature?	Yes, this functionality has been replaced by Advanced bank reconciliation functionality.
Product areas affected	All modules
Status	Deprecated: A removal date has not been set for this feature.

eBilanz (XBRL for Germany)

This functionality provided eXtensible Business Reporting Language (XBRL) output that is intended specifically for the German eBilanz taxonomy.

Reason for deprecation/removal	Lack of customer usage
Replaced by another feature?	This feature hasn't been replaced by another feature, but multiple specialized XBRL packages that provide rich XBRL functionality are available for the German market.
Product areas affected	Management Reporter
Status	Deprecated: A removal date has not been set for this feature.

Enterprise Portal client

Reason for deprecation/removal	A single client platform has been provided.

Replaced by another feature?	The new web client is based on the desktop form metadata and programming model that have been modified to provide a rich web platform.
Product areas affected	All modules
Status	Removed as of Dynamics AX 7.0.

Environmental sustainability

Reason for deprecation/removal	Low customer usage and a limited feature set
Replaced by another feature?	No
Product areas affected	Compliance and internal controls, Accounts payable
Status	Removed as of Dynamics AX 7.0.

Form ActiveX and Managed Host controls

Reason for deprecation/removal	The ActiveX and Managed Host controls are based on the deprecated desktop client.
Replaced by another feature?	The extensible control framework supports building new controls that are based on HTML, CSS, and JavaScript, and is a first-class control in the Microsoft Visual Studio Tooling environment.
Product areas affected	All modules
Status	Removed as of Dynamics AX 7.0.

Generate prenotes by using a batch

Prenote generation can't be done by using a batch, but it can still be done by a user.

Reason for deprecation/removal	No form exists to persist and display the resulting prenote file when it's generated by using a batch.
Replaced by another feature?	Prenotes can still be generated, and the user has control over the location where the file is saved.
Product areas affected	Accounts payable, Accounts receivable, Cash and bank management
Status	Removed as of AX 7.0.

German DTAUS payment export and account statement import (totals and transactions)

Reason for deprecation/removal	The format is no longer applicable in Germany, because it has been replaced by Single Euro Payments Area (SEPA) functionality.
Replaced by another feature?	Yes, this functionality has been replaced by SEPA payment export and advanced bank reconciliation functionality for importing account statements.
Product areas affected	All modules
Status	Deprecated: A removal date has not been set for this feature.

German DTAZV payment format in domestic Currency

Reason for deprecation/removal	The format is no longer applicable in Germany, because it has been replaced by SEPA functionality.
Replaced by another feature?	SEPA payments export
Product areas affected	Accounts payable
Status	Deprecated: A removal date has not been set for this feature.

German MT940 import

Reason for deprecation/removal	Generic functionality is now used instead of localized functionality.
Replaced by another feature?	Yes, this functionality has been replaced by Advanced bank reconciliation functionality.
Product areas affected	All modules
Status	Deprecated: A removal date has not been set for this feature.

German XML EU Sales list

Reason for deprecation/removal	The XML format for German EU Sales List reporting is no longer supported. Only the ELMA5 text file format can be used to submit the EU Sales List report to the German Tax Office.
Replaced by another feature?	No

Product areas affected	Tax
Status	Deprecated: A removal date has not been set for this feature.

GL SSRS reports

Reports that include the following menu items have been removed: **Summary trial balance**, **Detailed trial balance**, **Chart of accounts**, **Audit trail**, **Balances**, and **Balance list**.

Reason for deprecation/removal	Financial Microsoft SQL Server Reporting Services (SSRS) reports have been replaced by Management Reporter capabilities and default reports.
Replaced by another feature?	Management Reporter (labeled **Financial reporting** in the current version of Dynamics AX)
Product areas affected	General ledger
Status	Removed as of Dynamics AX 7.0.

InfoPart and FormPart metadata

Reason for deprecation/removal	InfoPart and FormPart metadata enabled the creation of FactBoxes for two different clients.
Replaced by another feature?	InfoPart metadata, which was a simplified form definition, is converted into a Form by upgrade tooling. FormPart metadata, which referenced a Form, is replaced by a more direct reference that is created by upgrade tooling.
Product areas affected	All modules
Status	Removed as of Dynamics AX 7.0.

Main account list page

A list of accounts for the legal entity and related balance information

Reason for deprecation/removal	Balance information is available on the **Trial balance** list page by account and dimension.
Replaced by another feature?	**Main accounts** contains the same list of accounts that the **Main account** list page contained. The grid view in **Main accounts** also shows an even smaller, grid-like view.
Product areas affected	General ledger

Status	Removed as of Dynamics AX 7.0.

Malaysia and Singapore bank cash flow report

This feature let the user print a cash flow report that shows transactions and details of the cash inflows and outflows for a specific date range for selected bank accounts.

Reason for deprecation/removal	The same information can be obtained from the Inquiry bank transaction.
Replaced by another feature?	The Inquiry bank transaction
Product areas affected	Cash and bank management
Status	Deprecated: A removal date has not been set for this feature.

Mexican CFD electronic invoice

This feature enabled the generation of Mexican electronic invoices by using the Comprobante Fiscal Digital (CFD) method, where the company signs the invoice by requesting the related authorization from the government. This feature also provides a monthly report that includes all electronics invoices that were issued in the period.

Reason for deprecation/removal	The method is no longer applicable. The generation of electronic invoices by using the CFD method was deprecated by the tax authorities and replaced by the Comprobante Fiscal Digital a través de Internet (CFDI) method, where the signing is delegated to the third-party provider (PAC). The monthly report has been removed, and an inquiry option lets users inquire about historical transactions.
Replaced by another feature?	No
Product areas affected	Account receivables, Project
Status	Deprecated: A removal date has not been set for this feature.

Mexico realized and unrealized VAT

Dynamics AX 2012 managed unrealized value-added tax (VAT) by using Mexico-specific functionality for unrealized tax.

Reason for deprecation/removal	Duplicate functionality
Replaced by another feature?	Yes, this functionality has been replaced by standard conditional sales tax functionality that is provided by Core.

Product areas affected	Tax
Status	Deprecated: A removal date has not been set for this feature.

Microsoft Outlook integration

Reason for deprecation/removal	This functionality has been replaced by Microsoft Exchange Server integration.
Replaced by another feature?	Yes
Product areas affected	Sales and marketing
Status	Removed as of Dynamics AX 7.0.

Private blocking of inventory and warehouse management journals

The inventory and warehouse journals no longer support the ability to mark a journal as private for a selected user. Only the process of blocking journals as private for user groups and blocking during editing is supported.

Reason for deprecation/removal	No use of the functionality was found.
Replaced by another feature?	No
Product areas affected	Inventory management
Status	Removed as of Dynamics AX 7.0.

Product builder

Product builder was used to dynamically configure items from a sales order, purchase order, production order, sales quotation, project quotation, or item requirement. Based on a product model that had modeling variables, the user could select values to meet the customer requirements and get a unique product variant that had a BOM and route.

Reason for deprecation/removal	Product builder exposed X++ code to end users and isn't supported in the current version of Dynamics AX. It has been removed to avoid duplicate maintenance efforts on overlapping, sizeable codebases.
Replaced by another feature?	Yes. The constraint-based configuration was introduced in Dynamics AX 2012 where the depreciation of Product builder in future versions was already announced. The constraint-based configuration technology is selected on the product masters to enable the configuration. To learn more, see Product configuration overview.

Product areas affected	Product information management, Sales and marketing
Status	Removed as of Dynamics AX 7.0.

Production Floor app

This is the app for tablet devices running Windows 8.1 RT and Windows 8.1 Pro.

Reason for deprecation/removal	With the change to a web-based client, it is possible to deliver similar functionality through the native Dynamics AX 7.0 client. The Job Card Device provides a production floor user interface that is optimized for touch and tablet form factors.
Replaced by another feature?	Yes. The Job Card Device, which is a native part of Dynamics AX 7.0.
Product areas affected	Production control
Status	Deprecated: A removal date from the Microsoft store has not yet been set for this feature.

Rename product dimension

This feature let you change the name of one of the three standard product dimensions (size, color, or style) to a name that better suited your business requirements. Renaming included all the labels where the product dimension name was used.

Reason for deprecation/removal	The current version of Dynamics AX doesn't support label changes at run time.
Replaced by another feature?	No
Product areas affected	Product information management
Status	Removed as of Dynamics AX 7.0.

Retail Server connectivity using HTTP

In Dynamics AX 2012 R3, the Retail Server could function using HTTP communication (non-secured). This was in addition to the standard communication using HTTPS.

Reason for deprecation/removal	Due to new security requirements, only secured communication using TLS 1.2 (or above, as available) is now supported. The self-service installer will automatically configure the computer for this communication.
Replaced by another feature?	No. Only standard HTTPS communication is now supported.

Product areas affected	Retail Server
Status	Removed as of Dynamics AX 7.0.

Role Center pages

Reason for deprecation/removal	Role Center pages were built on the deprecated Enterprise Portal platform, which has been replaced by the new web client platform in the current version of Dynamics AX.
Replaced by another feature?	The new Workspace form pattern provides users with a process-centered design that provides easy access to commonly used tasks within that process.
Product areas affected	All modules
Status	Removed as of Dynamics AX 7.0

Sales tax jurisdictions

Reason for deprecation/removal	Low customer usage and a limited feature set
Replaced by another feature?	No
Product areas affected	US sales tax
Status	Removed as of Dynamics AX 7.0.

Sites Services

Sites Services let you build websites that extend your business processes to the Internet without IT support.

Reason for deprecation/removal	The Microsoft Azure infrastructure that is used by Dynamics AX has new capabilities that can be used instead (for example, Azure sites).
Replaced by another feature?	No
Product areas affected	HR recruiting, Case management, Request for quotes, Vendor registration, Collaborative workspaces for opportunities and campaigns
Status	Removed as of Dynamics AX 7.0.

SSAS demand forecasting strategy

Reason for deprecation/removal	The design of the feature cannot be supported in the new cloud architecture.
Replaced by another feature?	Azure Machine Learning demand forecasting strategy
Product areas affected	Master planning
Status	Removed as of Dynamics AX 7.0.

Vendor invoice pool excluding posting details

Reason for deprecation/removal	Low usage. This functionality has been replaced by the Invoice journal that has workflow functionality.
Replaced by another feature?	Workflow capabilities of the Invoice journal.
Product areas affected	Accounts payable
Status	Removed as of Dynamics AX 7.0.

Virtual company accounts

The virtual companies feature is no longer supported in Dynamics AX. The virtual companies feature let users set up tables that could be shared by a set of companies. For a description of the feature, see Company accounts and Virtual company accounts. The feature works by grouping tables into collections that are assigned to virtual companies, which are groups of existing "real" companies. Queries are created so that all the companies in the virtual company can access the data in the tables of the associated table collections.

Reason for deprecation/removal	- Virtual companies must be set up before data is stored in the tables. Retrofitting virtual companies onto an existing implementation is very difficult. - Because there has been so much data normalization in the current version of Dynamics AX, it has become difficult to know what to add to the table collections. For example, it's difficult to know which tables to share. All the tables referenced from tables that are in a virtual company must also added. Because of table normalization, even simple master data that is spread across multiple tables must be part of the virtual company. Any mistake that is made here will cause functional issues. - When a table is part of a virtual company, it loses information about the origin of the data, and only the virtual company is recorded.
Replaced by another feature?	Global tables can be used to make tables accessible from all companies. Currently, there is no replacement.
Product areas affected	All modules

Status	Removed as of Dynamics AX 7.0.

Windows 8 tablet app

The Windows 8 tablet app provided functionality for expense entry and approval.

Reason for deprecation/removal	Finance and Operations is compatible with tablets. The tablet app is no longer required.
Replaced by another feature?	No.
Product areas affected	Expense management
Status	Removed: This functionality is only available for Dynamics AX 2012 R3.

Workplanner

Reason for deprecation/removal	Low usage
Replaced by another feature?	No, but the **Profile relation** page, which is opened from the **Profile groups** page, supports the same business scenario as the deprecated **Workplanner** page.
Product areas affected	Time and attendance
Status	The code has not been removed. However, the form, JmgWorkPlanner, was not migrated.

X++ financial statements

Reason for deprecation/removal	This functionality has been replaced by another feature.
Replaced by another feature?	Management Reporter (labeled **Financial reporting** in the current version of Dynamics AX)
Product areas affected	General ledger
Status	Removed as of Dynamics AX 2012

AX 2012 features that were postponed

2/18/2021 • 11 minutes to read • Edit Online

This topic lists features of Microsoft Dynamics AX 2012 that were postponed. These features weren't implemented in Microsoft Dynamics AX 7.0. In the following table, the **Current status** column indicates whether the feature has been implemented since the AX 7.0 release.

For a detailed list of the release date for each version, see Software lifecycle policy and cloud releases.

AX 2012 FEATURE THAT WAS POSTPONED	DESCRIPTION	CURRENT STATUS (AS OF FEBRUARY 2019)
Absence management in Human resources	Functionality for entering absence transactions isn't included. Additionally, functionality for approving absence transactions as a manager isn't included. Setup capabilities that are required for integration with other modules are available through the **Human Resources 2** configuration key.	Implemented in Dynamics 365 Human Resources
Alerts	Alerts help users keep track of data changes in the system.	Implemented in Platform update 15
Bank payment order for Latvia and Lithuania	You can print a payment order for Latvia and Lithuania. This feature will be available in a future update.	Not implemented
Bankgirot AP return format for Sweden	The Bankgirot return format is used to import bank return messages. This feature will be available in a future update.	Not implemented
Client drag-and-drop	The web client controls have application programming interfaces (APIs) for drag-and-drop operations, but these APIs are based on the deprecated desktop client technology and they require a redesign so that they work on the new web client platform. APIs that support drag-and-drop operations will be reviewed for inclusion in a future update.	Not implemented
Client right-to-left (RTL) layout	RTL layout is now supported.	Implemented in Platform update 2

AX 2012 FEATURE THAT WAS POSTPONED	DESCRIPTION	CURRENT STATUS (AS OF FEBRUARY 2019)
Cost accounting	The **Cost accounting** module is designed to meet the requirements of internal costs and profitability reports at multiple organizational levels. To define the cost object level, the module depends on a correct mapping of financial dimensions. The module lets you perform advanced allocations of cost origin from expenditures that are registered in the general ledger or budget. It also lets you compare realized costs and budgeted costs.	Implemented in version 1611
Customer self-service (CSS)	CSS lets you create approved customer records. It also allows users to view selected product catalogs, order items, and view the status of invoices. Additionally, CSS lets you create and follow return orders.	Not implemented
Customizable help topics	The ability to create customized help topics has not yet been implemented. Custom task guides and custom field help are available. This feature will be available in a future update.	Not implemented
Employee self-service (ESS)	ESS shows employees several tiles that have task-related and career-related information on a single page. Employees can view pending work items and click links that open pages where they can take action on their tasks. ESS pages also show employees the status of their certifications, when their next performance reviews are scheduled, skills, goals, and compensation information, and other information, such as balances for vacation and sick time. Employees can also access a company directory from their ESS page.	Implemented in version 1611
External questionnaire and recruiting functionality	Functionality for externally posting questionnaires and open jobs will be added to Human Resources in a future update.	External questionnaire functionality hasn't been implemented. Recruiting functionality is available in Microsoft Dynamics 365 Talent: Attract.
Fiscal printers for Poland	Integration with Polish fiscal printers enables the required information to be sent to the fiscal printer in the correct format during invoice posting. Examples of Polish fiscal printers include the Posnet Thermal and Elzab Omega printer types. This feature will be available in a future update.	Not implemented

AX 2012 FEATURE THAT WAS POSTPONED	DESCRIPTION	CURRENT STATUS (AS OF FEBRUARY 2019)
General budget reservations	The General budget reservations document is sometimes referred to as a commitment. Public sector entities often use this document to set aside or earmark budgeted funds so that they aren't available for other purposes.	Implemented in version 8.1
Graphics tab on the **Fixed asset value model** and **Depreciation book profile** pages	The chart shows the depreciation, accumulated depreciation, and net book value over time. Users can click the **Data** tab to view more detailed information than the chart shows. This chart will be redesigned in a future update.	Not implemented
Intelligent Data Management Framework (IDMF)	IDMF is an add-on tool that lets system administrators optimize performance. IDMF assesses the health of the application, analyzes current usage patterns, and helps reduce database size.	Not implemented
Microsoft Project client integration	The Microsoft Project client is integrated with projects.	Implemented in version 7.2 (July 2017 update)
Procurement site	In previous versions, the Employee self-service procurement site lets you enter requisitions for employees, view the status of an order (created, received, or receipt confirmed), and request onboarding of a new vendor. You could configure different procurement catalogs to show on the site depending on policy. You could also design procurement catalogs by adding new nodes. In the current version, procurement catalog capabilities are reduced and are used only to limit the products that can be ordered for an organization. The structure is always based on the Procurement categories hierarchy. Additionally, on the procurement site the employee could approve a vendor invoice and confirm receipts in relation to the requisitions and derived purchase orders.	Not implemented
Secure global address book	The ability to help secure the global address book by legal entity and address book is not available. This feature will be available in a future update.	Not implemented
Specifications for Electronic reporting (ER) payment formats	Currently, you must enter the payment format specifications manually. In a future update, you will be able to select payment format specifications in	Not implemented

AX 2012 FEATURE THAT WAS POSTPONED	DESCRIPTION	CURRENT STATUS (AS OF FEBRUARY 2019)
	a list. The following payment specifications are currently supported per payment format. [!NOTE] Values for these supported payment specifications are used as payment specification parameters on the **Payment specification** page for a selected method of payment. **BTL91 for the Netherlands** PAYMENT SPECIFICATION (USED IN ER) / EXPORT FORMAT DESCRIPTION ChqBen / Cheque, Begunstigde ChqOff / Cheque, Kantoor opdrachtgever ChqPri / Cheque, Opdrachtgever TrfBenBen / Overboeking Begunstigde/Begunstigde TrfBenBenUrg / Overboeking Begunstigde/Begunstigde Spoed TrfEurBen / Overboeking Euro/Begunstigde TrfEurBenUrg / Overboeking Euro/Begunstigde Spoed TrfEurEur / Overboeking Euro/Euro TrfEurEurUrg / Overboeking Euro/Euro Spoed TrfForBen / Overboeking VV-rekening/Begunstigde TrfForBenUrg / Overboeking VV-rekening/Begunstigde Spoed TrfForFor / Overboeking VV-rekening/VV-rekening	

AX 2012 FEATURE THAT WAS POSTPONED	DESCRIPTION	CURRENT STATUS (AS OF FEBRUARY 2019)

PAYMENT SPECIFICATION (USED IN ER)	EXPORT FORMAT DESCRIPTION
TrfForForUrg	Overboeking VV-rekening/VV-rekening Spoed

Betalingsservice for Denmark

PAYMENT SPECIFICATION (USED IN ER)	EXPORT FORMAT DESCRIPTION
B0112	BS-B 0112: Lang tekst & adresse
B0113	BS-B 0113: Erstat. bet. lang tekst
T0112	BS-T 0112: Lang tekst & adresse
T0117	BS-T 0117:FIK;kort frist;lang tekst&adr.

Nordea vendor for Denmark

PAYMENT SPECIFICATION (USED IN ER)	EXPORT FORMAT DESCRIPTION
56	Currency account transfer between Nordea accounts in Denmark
47	Domestic check
45	Domestic transfer
50	Express transfer
55	Intercompany transfer (domestic)
51	Intercompany transfer to a foreign bank
54	International check
52	Nordea intercompany payment

AX 2012 FEATURE THAT WAS POSTPONED	DESCRIPTION	CURRENT STATUS (AS OF FEBRUARY 2019)

PAYMENT SPECIFICATION (USED IN ER)	EXPORT FORMAT DESCRIPTION
43	Request for transfer
46	Transfer form/giro payment

ISO20022 Credit transfer (CH)

PAYMENT SPECIFICATION (USED IN ER)	EXPORT FORMAT DESCRIPTION
Tp1.ESROPS	Type 1 - ESR orange payment slip
Tp21.ISR1SPS	Type 2.1 - IS red 1 stage payment slip
Tp22.ISR2SPS	Type 2.2 - IS red 2 stage payment slip
Tp7.Dmstc	Type 7 - Domestic postal order
TpE1.PSWR	Type E1 - Payment slip with reference
TpE2.PSWN	Type E2 - Payment slip with notifications

AvtaleGiro (NO)

PAYMENT SPECIFICATION (USED IN ER)	EXPORT FORMAT DESCRIPTION
Varsling	AvtaleGiro-trans with notification

AutoGiro (NO)

PAYMENT SPECIFICATION (USED IN ER)	EXPORT FORMAT DESCRIPTION
Melding	Autogiro-trans with notification

eFaktura (NO)

<table>
<tr><th>AX 2012 FEATURE THAT WAS POSTPONED</th><th>DESCRIPTION</th><th>CURRENT STATUS (AS OF FEBRUARY 2019)</th></tr>
<tr><td></td><td>
<table>
<tr><th>PAYMENT SPECIFICATION (USED IN ER)</th><th>EXPORT FORMAT DESCRIPTION</th></tr>
<tr><td>Reklame</td><td>Include advertising flag</td></tr>
</table>
ISO20022 Credit transfer (DK)
<table>
<tr><th>PAYMENT SPECIFICATION (USED IN ER)</th><th>EXPORT FORMAT DESCRIPTION</th></tr>
<tr><td>EasyAccountTransfer</td><td>Easy-account with CVR (NKV)</td></tr>
<tr><td>Paym_slip</td><td>Transfer forms (OCR)</td></tr>
</table>
ISPAG-CNAB240 format (BR)
<table>
<tr><th>PAYMENT SPECIFICATION (USED IN ER)</th><th>EXPORT FORMAT DESCRIPTION</th></tr>
<tr><td>A</td><td>OP (payment order), DOC (wire transfer), TED (other type of wire transfer), and direct credit in the account</td></tr>
<tr><td>J</td><td>Bar code payments (invoice with bar code or other type of documents with bar code)</td></tr>
<tr><td>O</td><td>Tax payments or other publics services payments</td></tr>
</table>
</td><td></td></tr>
<tr><td>US Payroll</td><td>US Payroll provides gross-to-net processing for employees in the United States. In Payroll, you can set up, enter, and maintain all payroll records and transactions.</td><td>Implemented in version 1611</td></tr>
</table>

AX 2012 FEATURE THAT WAS POSTPONED	DESCRIPTION	CURRENT STATUS (AS OF FEBRUARY 2019)
Vendor collaboration (Vendor Portal)	Dynamics AX 2012 provided vendor portal capabilities via Enterprise Portal. Financial and Operations also provides these capabilities. In version 7.1 (also known as Dynamics 365 for Operations 1611), a vendor could view and respond to purchase orders. In version 7.3, the vendor can view and respond to RFQs. Vendors can also view and edit selected information from the vendor record such as addresses, contact information, and contact persons, and they can upload documents in relation to their certifications.	Implemented in version 7.3
Vendor requests - external request to become a new vendor	Dynamics AX 2012 provided the ability for an anonymous user to sign up to be a vendor in the system, which could lead to a vendor request for adding a new vendor to the vendor master. In version 7.3, the anonymous request from a prospective vendor can be imported via an entity (Data Management/OData), which can lead to inviting the vendor - or the vendor's contact person - to register more details about the prospective vendor. The information provided is included in a new vendor request that can be reviewed and approved via a workflow process. An approval of the vendor request leads to creation of a new vendor account.	Implemented in version 7.3
Vendor requests in general	Dynamics AX 2012 had a concept of vendor requests that served various purposes related to updating vendor-related information, such as requesting new procurement categories for the vendor, internal employees requesting new vendors, or requesting to add a vendor to another company. Only the vendor's request of being added as a vendor has been implemented in version 7.3.	Not implemented

AX 2012 FEATURE THAT WAS POSTPONED	DESCRIPTION	CURRENT STATUS (AS OF FEBRUARY 2019)
[Russia] Tax registers	Legal entities can use registers to disclose their revenues and expenses. The registers are used to track revenue and expense data from the time that primary documents, such as sales invoices and delivery notes, are first entered by using the calculation of cost prices for production. The data from the registers is used to confirm the declared profit of the legal entity. This functionality includes the following features: • Current period incomes • Tax expenses • Other expenses of current period • Unrealized expenses of current period • Other unrealized expenses • Accounts receivable debt – inventory • Bad debts reserve calculation • Bad debts reserve movement • Accounts receivable movement • Procedure for writing-off AR bad debts • Accounts payable debt - inventory • Accounts payable debt movement • Procedure for writing-off AP bad debts • Goods cost calculation • FA object information • IA object information • FA depreciation • IA depreciation • FA/IA sale • Depreciation bonus recovery	Implemented in version 8.1.3
[Russia] Electronic export/import format for Client-Bank interface and reconciliation procedure	Electronic formats for export of outgoing payments, and import of incoming payments.	Implemented in version 8.1.3
[Russia] VAT declaration	Electronic format of VAT declaration.	Implemented in version 10.0.1
[Russia] Cash Flow Management	The functionality which obtains a cash flow forecast and performs an analysis, manages payments on a daily basis using payment schedule journals, controls the company's cash position, and maintains the company's cash flows with centralized control,	Implemented in version 10.0.1

AX 2012 FEATURE THAT WAS POSTPONED	DESCRIPTION	CURRENT STATUS (AS OF FEBRUARY 2019)
[Russia] Accounting reporting format	Electronic format of the following accounting reports: BalanceSheet, IncomeStatement, CashFlow, EquityStatement, TargetUsageMoney	Implemented in version 10.0.1
[Russia] Assessed tax reporting	Assessed tax declaration.	Implemented in version 10.0.1
[Russia] Land tax reporting	Land tax declaration. Creation of Land tax declaration by separate divisions.	Implemented in version 10.0.1
[Russia] Transport tax reporting	Transport tax declaration.	Implemented in version 10.0.1
[Russia] Indirect tax return (VAT and Excise) on import of goods	Indirect (withholding) tax return (VAT and Excise) on import of goods from state members of Customs union.	Implemented in version 10.0.1
[Russia] Journal of Alcohol sales in Retail	Daily Alcohol journal.	Implemented in version 10.0.1
[Russia] Optional posting of transfer orders to General ledger	Option to post/not post transactions to General ledger when posting a transfer order.	Implemented in version 8.1.2
[Russia] Inventory owner	Inventory dimension used to track owner of inventory (consignment stock, bailment, tolling, etc.).	Implemented in version 10.0.1
[Russia] AP/AR - Third-party miscellaneous charges	Registration of third-party miscellaneous charges and allocation by the following regimes: Inclusion into cost of purchased goods (allocation to invoices lines from other vendors), and redrawing to other parties re-allocation to other expense accounts.	Implemented in version 8.1.1
[Russia] Goods in transit from vendor	Registering goods in transit from vendor by special posting profile with Item type "purchased items en route". Creating Act of inventory holdings en route. (INV-6)	Implemented in version 8.1.2
[Russia] Goods in transit - sales to customer with postponed passing of property	Post sales invoice with postponed property transfer: no customer debts posted, all outgoing taxes are posted, items are transferred to transit warehouse. Register passing of property with posting debts and items sale from transit warehouse.	Implemented in version 8.1.2
[Russia] Bailment - accounting at bailee side	Accounting of inventory receipt for bailment as required by the Law and generation of primary form MX-1. Accounting of inventory return from bailment and generation of primary form MX-3. Bailment costs calculation from bailee side.	Implemented in version 8.1.2

AX 2012 FEATURE THAT WAS POSTPONED	DESCRIPTION	CURRENT STATUS (AS OF FEBRUARY 2019)
[Russia] Bailment - accounting at owner side	Accounting of inventory transfer to bailment and inventory return from bailment on goods owner side under bailment service contract.	Implemented in version 8.1.2
[Russia] Localization of Process Industries solution	Basic localization in two areas: correspondence of accounts for all new general ledger postings, and functional coexistence of Process Industries features and Russian country context (no issues when both Process Industries and Russian country context are enabled).	Implemented in version 10.0.1
[Russia] Alcohol sales declarations: Application 6, 7, 8 for wholesale. Applications 11, 12 for retail	Keeping track of alcoholic beverages types including producers, unit of measures, licenses for retail and wholesale trade. Preparing data for alcoholic beverages activities, including printing declarations and exporting them in XML format through e-reporting.	Implemented in version 10.0.1

End of mainstream support for Microsoft Dynamics AX 2009, Dynamics AX 2012, and Dynamics AX 2012 R2

2/18/2021 • 4 minutes to read • Edit Online

Mainstream support for Dynamics AX 2009 Service Pack 1 (SP1), Dynamics AX 2012, and Dynamics AX 2012 R2 ends on October 9, 2018. After that date, only security hotfixes will continue to be provided for these three versions through the extended support period that continues until October 12, 2021. For more information, see support.microsoft.com.

Mainstream support for Dynamics AX 2012 R3 continues through October 12, 2021. Microsoft will continue making security hotfixes, non-security hotfixes, and regulatory updates for Dynamics AX 2012 R3 throughout that mainstream support period. The source code for these non-binary, non-security hotfixes and regulatory updates will continue to be available for customers, and their partners, active on the Enhancement Plan or Software Assurance.

Dynamics AX 2009 SP1, Dynamics AX 2012, and Dynamics AX 2012 R2 customers can selectively integrate those changes as required. Customers and partners can get the source code from packages attached to relevant Dynamics AX 2012 R3 KB articles published on Lifecycle Services (LCS) and discoverable through Issue Search.

Customers are advised to upgrade to the latest version of Finance and Operations apps, such as Dynamics 365 Finance, Supply Chain Management, Retail, and Human Resources:

- Dynamics AX 2009 Service Pack 1 customers should use the migration tool that is available.
- Dynamics AX 2012 and Dynamics AX 2012 R2 customers should upgrade to Finance and Operations apps through Dynamics AX 2012 R3 using the upgrade tool that is available. Additional upgrade information is available in the Upgrade from AX 2012 to Finance and Operations apps topic.

Frequently asked questions

When does the mainstream support for Dynamics AX 2009 Service Pack 1, Dynamics AX 2012, and Dynamics AX 2012 R2 end?

Mainstream support ends on October 9, 2018.

Was the information of the end date of the mainstream support for Dynamics AX 2009 Service Pack 1, Dynamics AX 2012, and Dynamics AX 2012 R2 available before?

Yes, it was always publicly available on the Microsoft Support Lifecycle site at support.microsoft.com.

Can customers on Premier Extended Hotfix Support or on Unified Support Advanced and Performance Levels get a non-security hotfix or regulatory update?

No. Neither non-security hotfixes nor regulatory updates will be available for the Dynamics AX products during the Extended Support phase of the product lifecycle (Dynamics AX 2009 SP1, Dynamics AX 2012, or Dynamics AX 2012 R2).

While the ability to request a non-security hotfix for select products is included with Unified Support Advanced and Performance Levels, Microsoft has determined that non-security hotfixes cannot be provided with a *commercially reasonable* effort for these products. As a result, no requests for non-security hotfixes or regulatory updates will be accepted. However, Microsoft will continue making security hotfixes, non-security hotfixes, and regulatory updates for Dynamics AX 2012 R3 throughout that mainstream support period. The source code for these non-binary, non-security hotfixes and regulatory updates will continue to be available for

customers, and their partners, active on the Enhancement Plan or Software Assurance. Dynamics AX 2009 SP1, Dynamics AX 2012, and Dynamics AX 2012 R2 customers can selectively integrate those changes as required. Customers and partners can get the source code from packages attached to relevant Dynamics AX 2012 R3 KB articles published on LCS and discoverable through LCS Issue Search.

I knew about the regulatory change before October 9, 2018, but it has the law enforcement date after October 9, 2018. Will I still get a regulatory update for Dynamics AX 2009 Service Pack 1, Dynamics AX 2012, and Dynamics AX 2012 R2?

No, we will only provide regulatory updates for Dynamics AX 2009 Service Pack 1, Dynamics AX 2012, and Dynamics AX 2012 R2 for regulatory changes with the law enforcement dates on or earlier than October 9, 2018.

A customer or partner can already download a fix through LCS and inspect the code by installing it into a test Dynamics AX 2012 R3 environment. Is there any difference with the approach that you have proposed?

No, there is no difference.

What happens if a new bug is found by a customer in Dynamics AX 2009 Service Pack 1, Dynamics AX 2012, or Dynamics AX 2012 R2?

The bug must be reproducible in Dynamics AX 2012 R3. If it is reproducible and accepted, then a hotfix will be provided for Dynamics AX 2012 R3 and the customers can elect to integrate this hotfix in their version themselves, or work with their partners to integrate the changes.

How are binary hotfixes handled for Dynamics AX 2009 Service Pack 1, Dynamics AX 2012, and Dynamics AX 2012 R2?

If a hotfix is needed for a part of the system where Microsoft does not provide the source code and it is not a security bug, the hotfix will not be provided.

Business events overview

2/18/2021 • 14 minutes to read • Edit Online

Business events provide a mechanism that lets external systems receive notifications from Finance and Operations applications. In this way, the systems can perform business actions in response to the business events.

Business events occur when a business process is run. During a business process, users who participate in it perform business actions to complete the tasks that make up the business process.

A business action that a user performs can be either a workflow action or a non-workflow action. Approval of a purchase requisition is an example of a workflow action, whereas confirmation of a purchase order is an example of a non-workflow action. Both types of actions can generate business events that external systems can use in integration and notification scenarios.

Prerequisites

Business events can be consumed using Microsoft Power Automate and Azure messaging services. Therefore, customers must bring their subscriptions to such assets to use business events

> **IMPORTANT**
>
> Business events must not be considered a mechanism for exporting data. By definition, business events are supposed to be lightweight and nimble. They aren't intended to carry large payloads to fulfill data export scenarios.

Business events that are implemented

Business events are implemented in some business processes out of the box. These business events include both workflow and non-workflow business events. For more information, see Application business events, Workflow business events, and Alerts as business events.

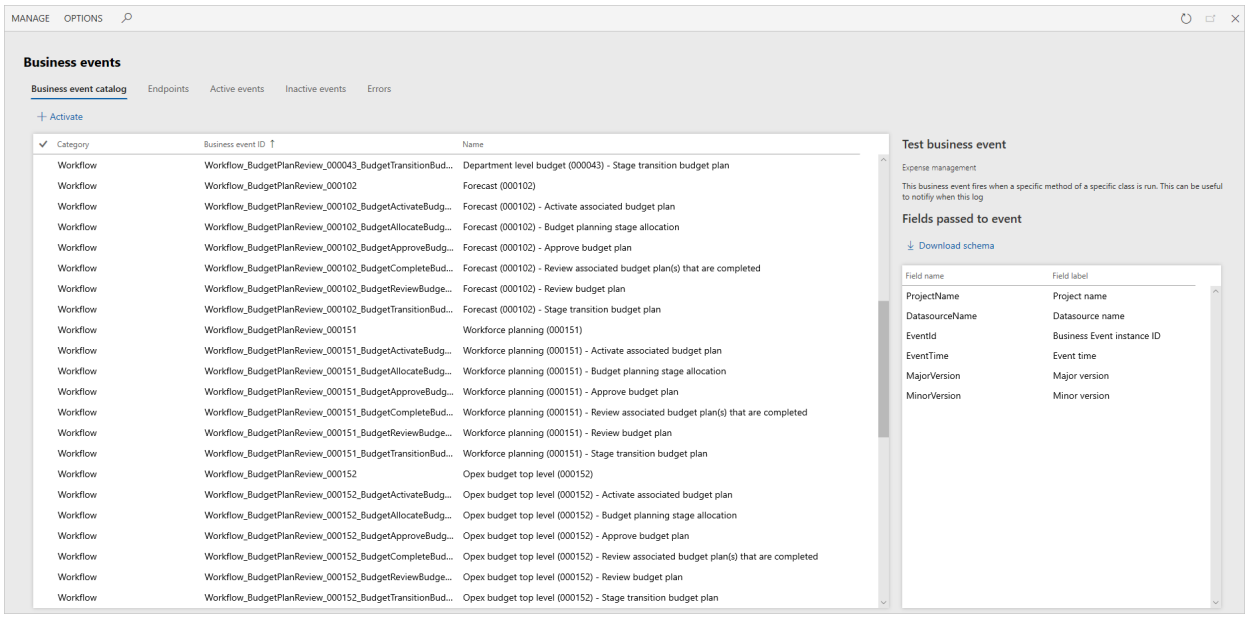
A developer must use extensions to implement new business events. For more information, see Business events developer documentation.

Business event catalog

The business events catalog can be accessed from **System administration > Set up > Business events**. The business event catalog lists the business events that are available in the instance that you're using. The catalog is useful because it shows which business events are available, and you can filter it by category, business event ID, and name.

The category of a business event identifies its source. Business events that originate from the workflow system are assigned to the **Workflow** category. For business events that originate from other modules, the module name is used as the category name.

The business event catalog is built during database synchronization at the time of deployment. Therefore, users should see the complete list of business events in the catalog. However, if an explicit update of the catalog is required, you can select **Manage > Rebuild business events catalog**.



For each business event, the business event catalog shows a description. This description can help you better understand the business event and its context in the business process. The catalog also shows the list of data fields that will be sent out in the event.

In scenarios where external integration systems require the schema of the payload for a business event during development, you can select **Download schema** to download the JavaScript Object Notation (JSON) schema.

In summary, the business event catalog helps identify the business events that are required for an implementation. It also helps identify the schema for each business event.

The next step is to manage the endpoints.

Business events parameters and processing

The application allocates dedicated batch threads to process business events in near real time. The maximum number of threads cannot exceed the total threads available in the system (**System administration > Server configuration**). Because threads are a shared resource for all batch processing, care must be taken when deciding to change the thread allocation for business events. The total threads allocated for business events is controlled using a parameter in the business events parameter table. This setting is not exposed from the user interface (UI), so a support case must be created to get this count changed in production environments as this will need database access.

> **IMPORTANT**
>
> There may be reliability issues with dedicated batch threads. Microsoft is working to resolve this issue and as a result, we recommend that you schedule the manual batch job to process business events, as explained below. We will update this topic when this issue is resolved.

The business events batch processing job is available as a workaround to mitigate issues with the dedicated processing, if needed. The batch job can be enabled and scheduled from the **Business events parameters** page.

In the event of an error while sending business events to its end point, the system retries to send the business events three times with an interval of one second per retry. This is the default setting that can be changed on the **Business events parameters** page.

The number of endpoints that can subscribe to the same business event in a legal entity is limited to ten by default. This can be changed on the **Business events parameters** page.

The out-of-the-box default settings for the above described parameters can be restored on the **Business events parameters** page.

Managing endpoints

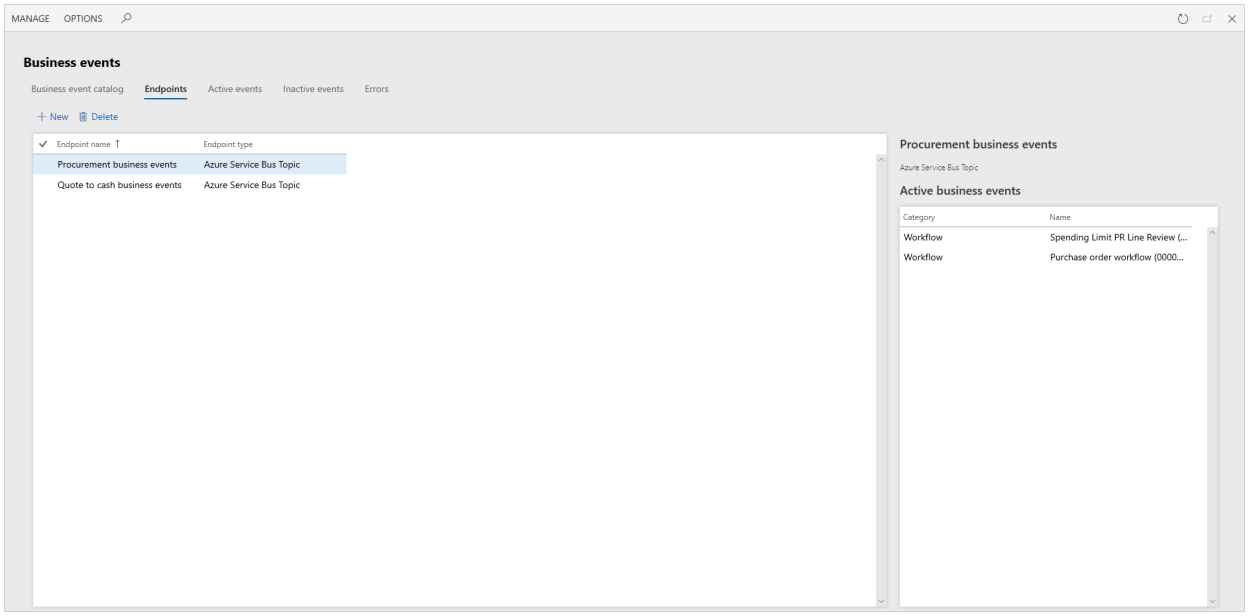
Endpoints let you manage the destinations for sending business events to. The following types of endpoints are currently supported. Endpoints can be created for these messaging and event brokers out of the box.

- Azure Service Bus Queue
- Azure Service Bus Topic
- Azure Event Grid
- Azure Event Hub
- HTTPS
- Microsoft Power Automate

Some scenarios might require multiple endpoints for organized distribution of business events to consumers. You can create multiple endpoints to support these scenarios.

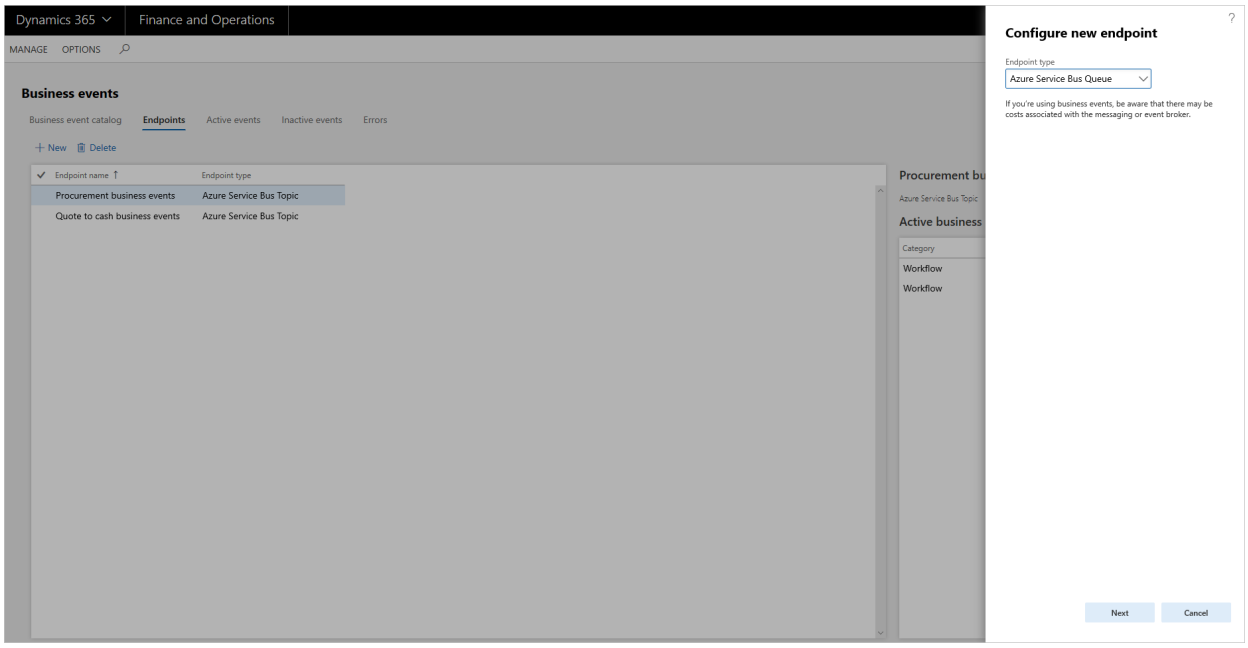
The Azure-based endpoints must be in the customer's Azure subscription. For example, if Event Grid is used as an endpoint, the endpoint must be in the customer's Azure subscription.

The application doesn't provision the endpoints. It just sends events to the endpoints that are provided. Customer might incur additional costs if they use these endpoints in their Azure subscription.

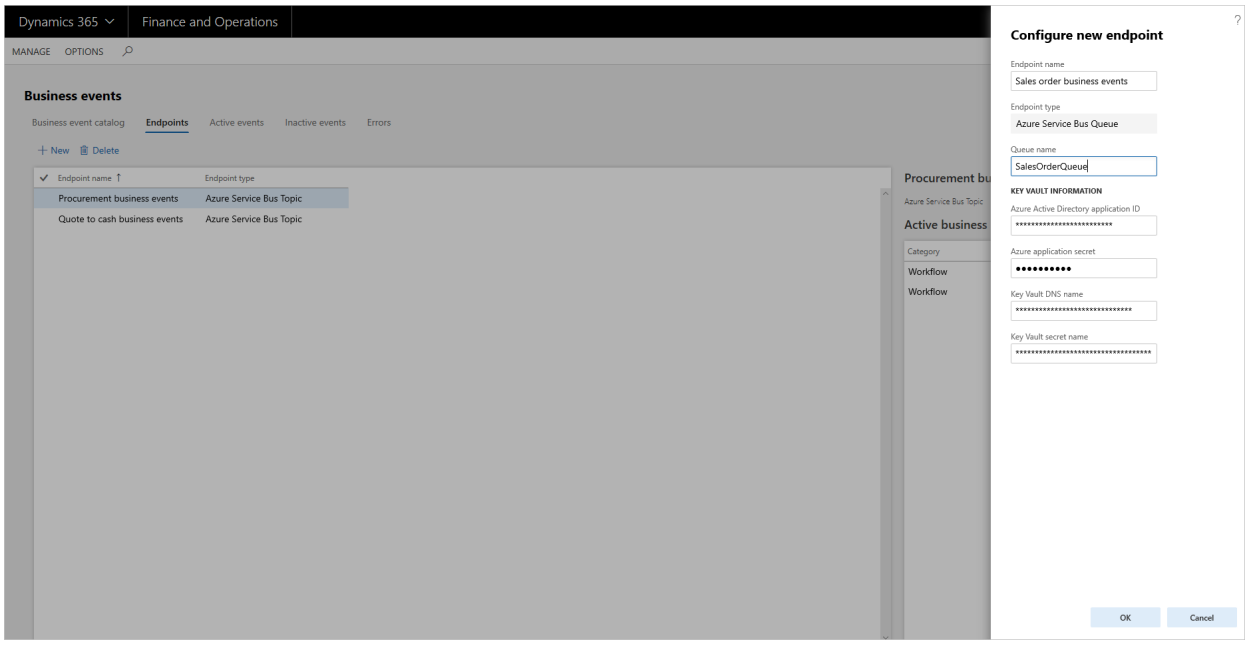


Create an Azure Service Bus Queue endpoint

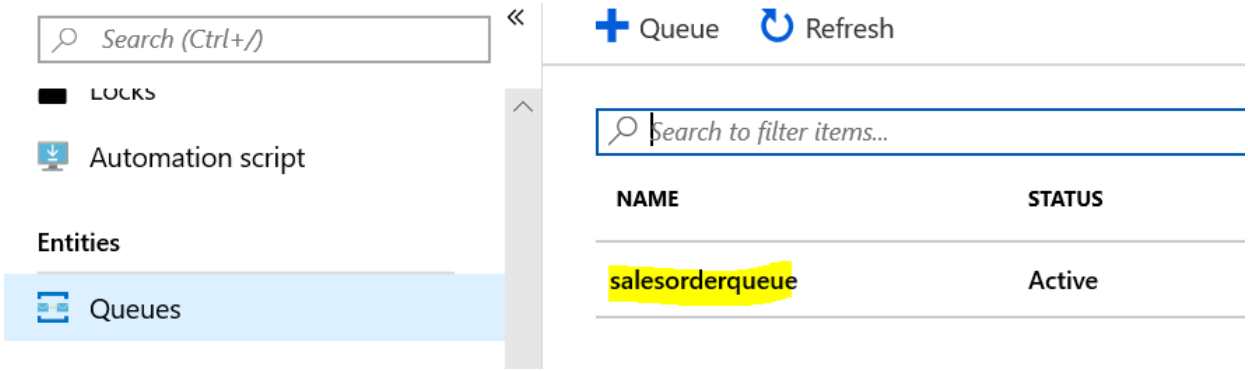
To create a new endpoint, select **New**. Then, in the **Endpoint type** field, select the appropriate endpoint type. To create an endpoint to a Service Bus queue, select **Azure Service Bus Queue**.



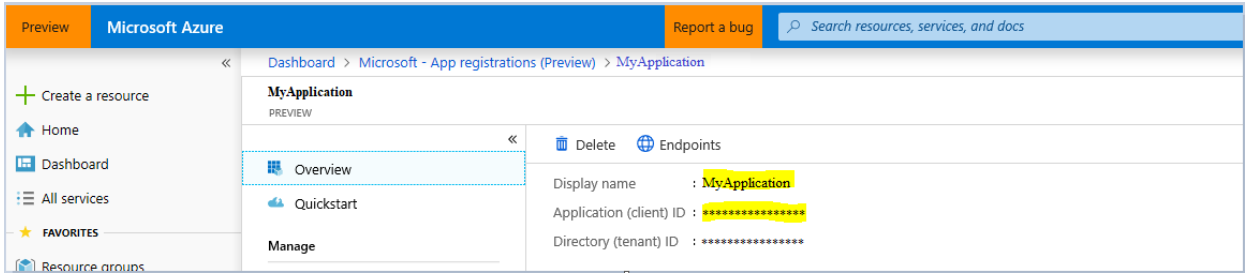
Select **Next**, and specify the name of the endpoint and the Service Bus queue. In addition, you must set up Azure Key Vault to provide the secret to the Azure messaging resource. You must also set up the Azure Active Directory (Azure AD) application ID and application secret.



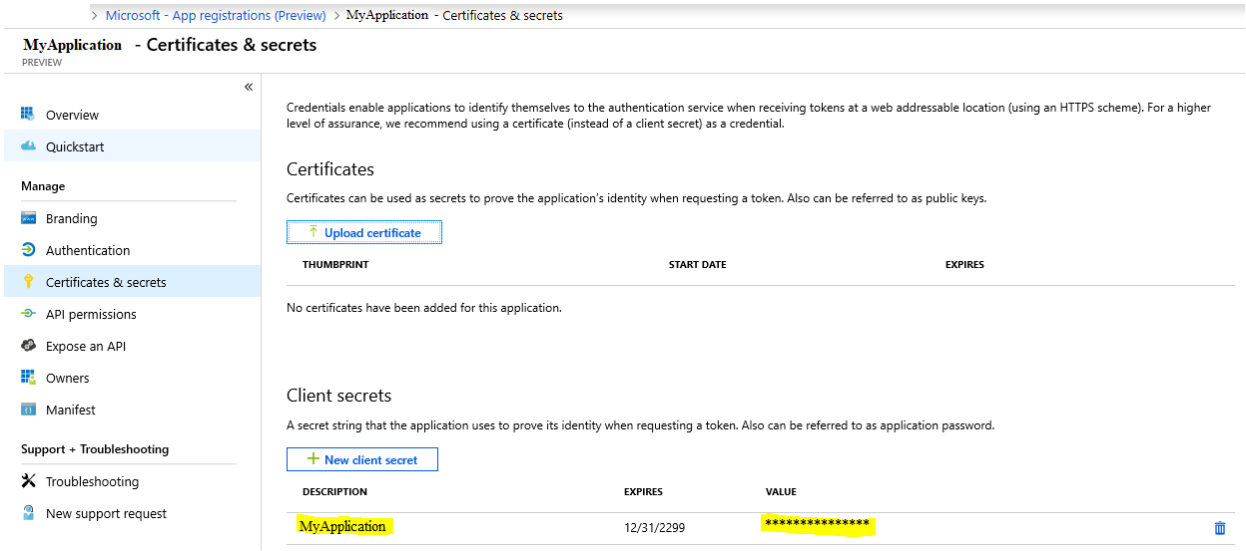
In the **Queue Name** field, enter the **Azure Service Bus Queue** name that you created in the Azure Service Bus Queue configuration in Azure.



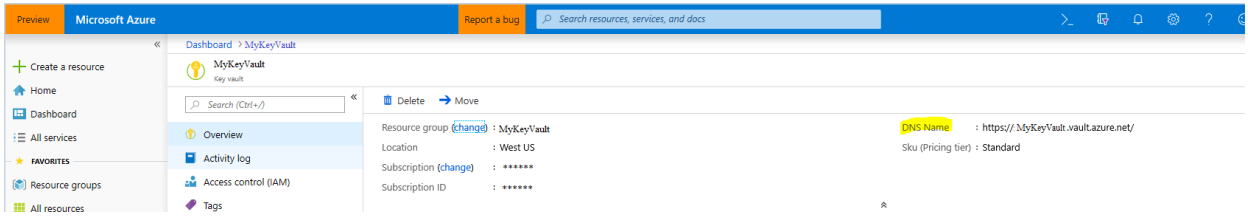
In the **Azure Active Directory application ID** field, enter the application ID that is created in Azure AD in the Azure portal.



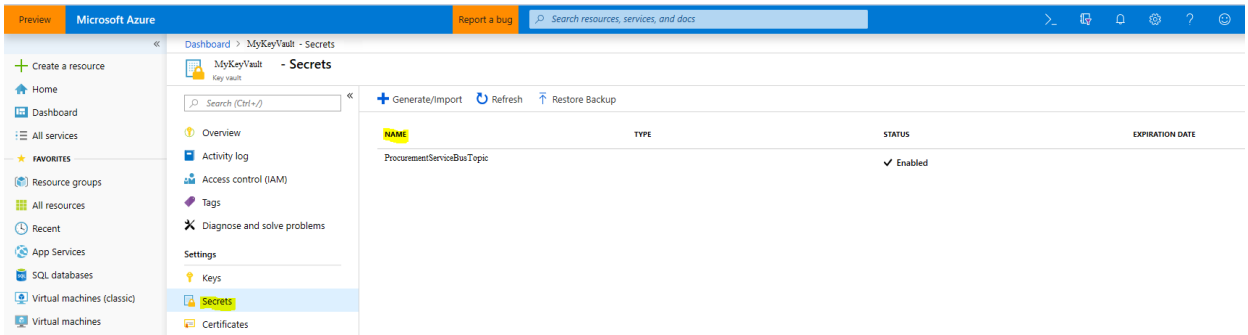
In the **Azure application secret** field, enter the secret value for the application.



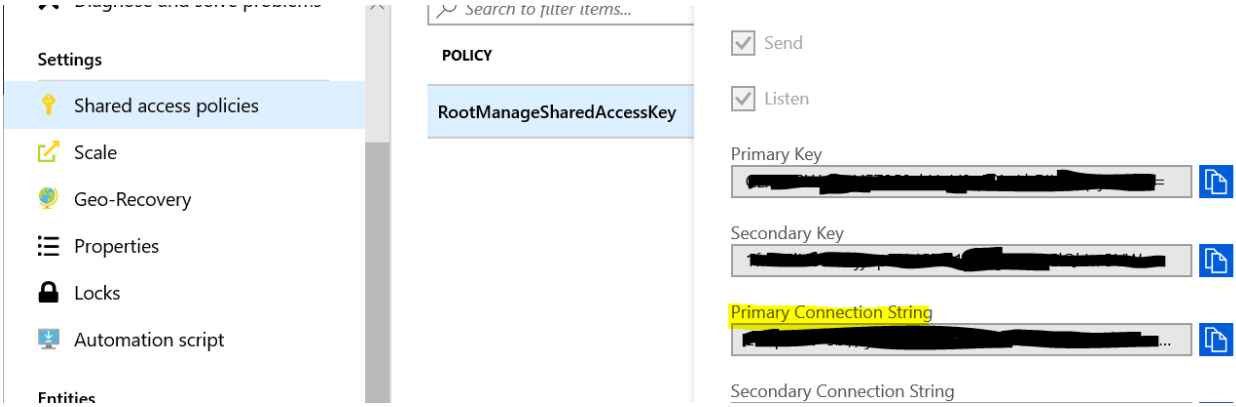
In the **Key vault DNS name** field, enter the name from your Key Vault setup.



In the **Key vault secret name** field, enter the secret name for the endpoint resource that must be created in Key Vault.



The **Key Vault Secret** value, in Azure, will be the Azure Service Bus **Primary Connection String** value. This value is found in the Azure Service Bus that you configured in **Shared Access Policies** > **RootManagedSharedAccessKey**.



IMPORTANT

The Azure application that was registered must be also added to the Key Vault set up under Access policies in the Key Vault. For this setup to be complete, select the **Key, Secret & Certificate Management** template and then select the application as the **principal**.

Create an Azure Service Bus Topic endpoint

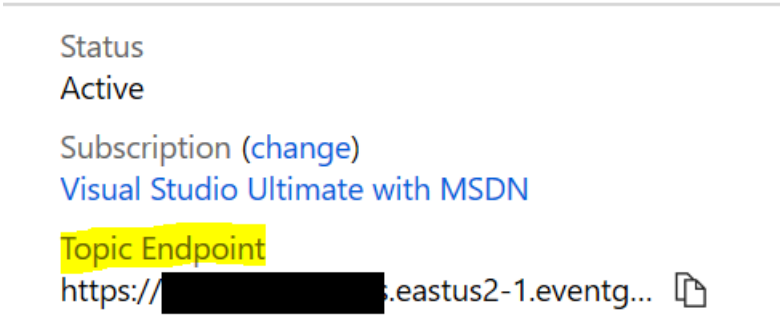
To create an endpoint to a Service Bus topic, select **New**, and then, in the **Endpoint type** field, select **Azure Service Bus Topic**. The **Topic name** field must be set to the name of the Service Bus topic. Key Vault information is set up in the same way that it is set up for an Azure Service Bus Queue endpoint.

Create an Azure Event Grid endpoint

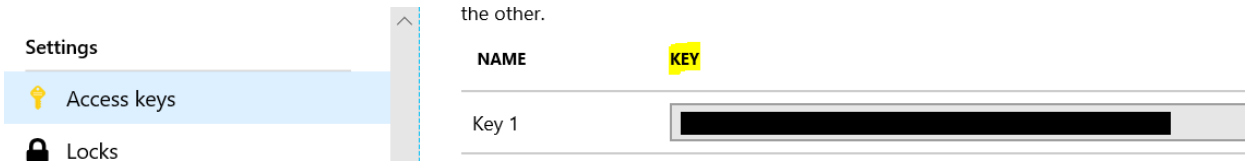
To create an endpoint, you need to create and configure an **Azure Event Grid Topic** in Azure Portal, and then create an endpoint to the Event Grid Topic in the **Business Events Workspace**. Go to the **Endpoints** tab, select **New**, and then select **Azure Event Grid** as the **Endpoint type**. In the **Endpoint URL** field, enter the URL from the **Azure Event Grid Topic**. This is the **Topic Endpoint** value in the **Overview** section of your Event Grid Topic.

IMPORTANT

The Azure application that was registered must be also added to the Key Vault set up under Access policies in the Key Vault. For this setup to be complete, select the **Key, Secret & Certificate Management** template and then select the application as the **principal**.



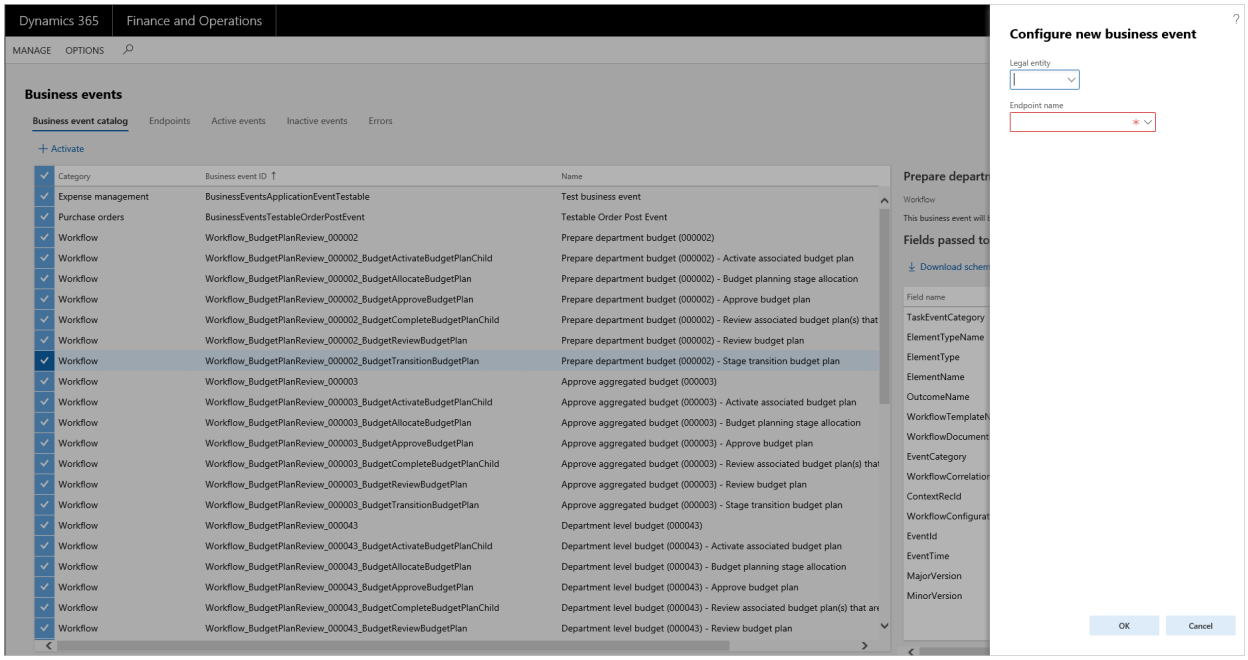
Key Vault information is set up in the same way that it is set up for an Azure Service Bus Queue endpoint, except the Key Vault secret should now point to the Event Grid credential, rather than the Service Bus connection string. The Event Grid Credential can be found under the Event Grid you created in the **Access Keys** section under Settings.



After you've created the endpoints that you require, the next step is to activate the business events.

Activating business events

Business events in the business event catalog aren't active by default. From the catalog, you can activate any business events that you require. Select one or more business events, and then select **Activate**.

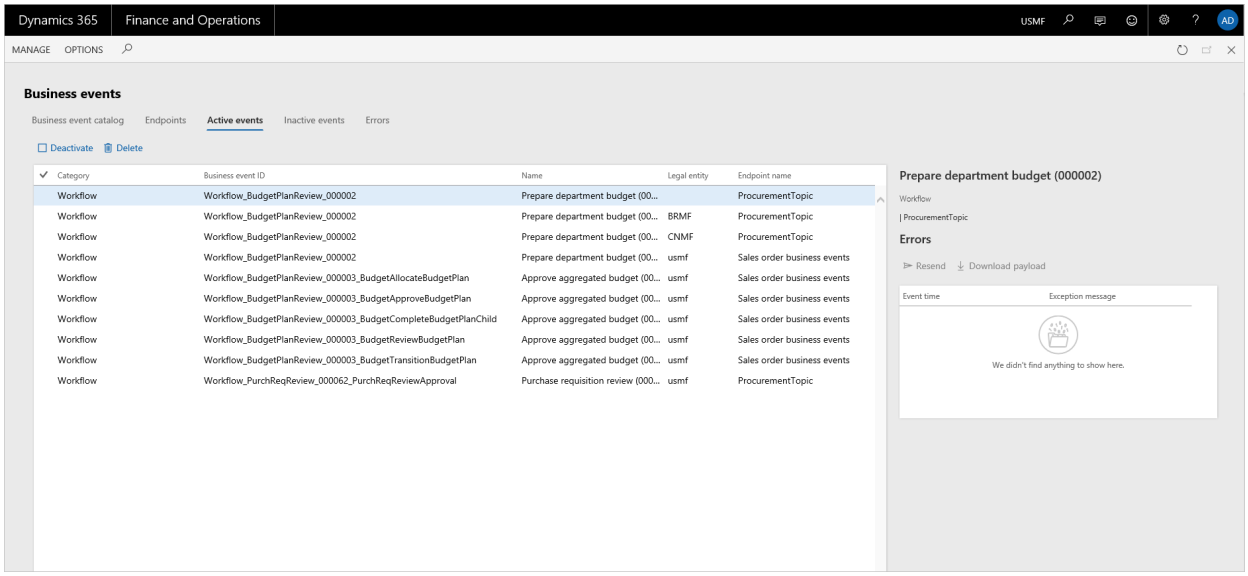


Business events can be activated either in all legal entities or in specific legal entities. If you leave the **Legal entity** field blank, the selected business events will be activated in *all* legal entities. If a business event is required only for specific legal entities, it must be configured separately for each legal entity.

Endpoints must be assigned to the business events that are activated.

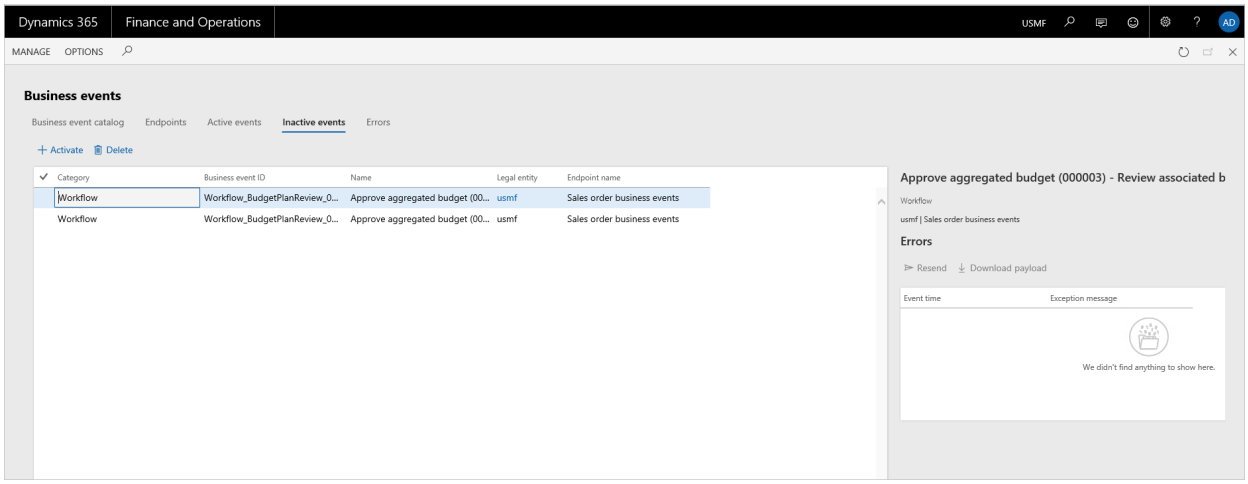
When business events occur as business processes are run, the system will do outbound processing only for business events that have been activated.

After business events are activated, they appear on the **Active events** tab.



From the **Active events** tab, you can inactivate business events. The system won't do outbound processing for inactivated events.

After business events are inactivated, they appear on the **Inactive events** tab.



Business events can be inactivated when processing of business events must be paused for a period because of specific system maintenance activities in the integration landscape.

When business requirements change, some business events might no longer be required. In this case, you can inactivate them instead of deleting them from the list of active events. This approach is useful if the history of errors for the business events must be preserved. Inactivated business events can be deleted later, when there is no longer a business need to keep them inactivated.

Errors

While the system does outbound processing of business events, errors can occur. These errors might prevent the system from successfully delivering a business event to the endpoint. If an error occurs, the system retries several times to successfully process the business event. However, if all attempts are unsuccessful, the business event is saved in an error log.

Error logs can be accessed from the **Active events**, **Inactive events**, and **Errors** tabs. The **Errors** tab shows all errors across all business events, whereas the other two tabs show errors in the context of a specific business event.

You can do on-demand outbound processing on each error by using the **Resend** action. This action invokes the outbound processing logic. This logic includes retries. If the outbound processing is still unsuccessful, the error is logged in the error log. In this case, the **Last process time** field on the **Errors** tab indicates when the last attempt to process the event occurred.

If an error can't be successfully processed, you can use the **Download payload** option to download the payload from the event for offline processing, as you require.

> **NOTE**
>
> If an endpoint is deleted and a new endpoint is associated with business events, all errors that are associated with the business events can still be resent. In this case, the system will do outbound processing to send to the new endpoint that is associated with the corresponding business event. This functionality allows for graceful recovery from misconfiguration or other error states.

Business event consumption models

The integration requirements and integration solution design for implementations vary. The integration

requirements play a role in identifying the consumption model for business events. In summary, you must consider the following points when you design integrations that use business events:

- Business events can be consumed using Power Automate, Service Bus, Event Grid, or other endpoint types.
- Customers must bring their own subscriptions to use Power Automate, Service Bus, Event Grid, or other endpoint types.
- A business event can be activated in all legal entities or in specific legal entities.
- A business event can be sent to a unique endpoint or the same endpoints.
- Power Automate can directly subscribe to business events.

Idempotency

Business events enable idempotent behavior on the consuming side by having a control number in the payload. The control number is an upwardly increasing number, which can be tracked by the consuming application to detect duplication and/or out of order delivery. The control number cannot be misread as the sequence number because the control number cannot be sequential. There can be gaps in the numbering space.

Filtering in Azure Event Grid and Azure Service Bus

Azure Service Bus and Azure Event Grid supports subscribing to topics by specifying criteria on the incoming message. For more information, see Topic filters and actions and Understand event filtering for Event Grid subscriptions.

A business event that is sent to an Azure Service Bus or Azure Event Grid has the following fields made available for this purpose. Subscribers can use this information to subscribe to more specific topics as required.

- **Category** – This is the business event category as displayed in the business event catalog. This is useful as a filter criterion when a common topic is used for receiving business events from multiple categories and subscribers want to only receive business events for the category that they are interested in.
- **Business event ID** – This is the class name of the business event implementation as displayed in the business event catalog. This uniquely identifies the business event (not the instance of the business event) and thus helps in validation of received business events on the consumer side to ensure the expected business event is what is being received and processed.
- **Legal entity** – This is the legal entity in which the business event happened. This is a useful information to base the consuming logic on if the processing and distribution of business events on the consumption side must be driven by a legal entity.

> **NOTE**
> The filterable fields that are sent in a business event can be modified to include custom fields. This is a developer experience.

Role-based security for business events

Starting in Platform update 29, role-based security can be applied to business events to meet the following requirements using the appropriate security artifact.

REQUIREMENT	PRIVILEGE	DUTY
Only certain users must have access to view the business events catalog.	**BusinessEventsCatalogView**	None

REQUIREMENT	PRIVILEGE	DUTY
Only certain users must have access to activate business events.	**BusinessEventsCatalogMaintain**	None
Only certain users must have access to create and manage endpoints.	**Business events security privilege**	**Business events security duty**
Users must only be able to subscribe to business events which they have been granted access to from external applications like Power Automate.	**Subscribe to business events from service**	None
Only certain users must be able to view the business events security setup.	**BusinessEventsCatalogSecuritySetupView**	None
Only certain users must be able to manage business events security.	**Maintain business events catalog security**	None

These privileges can be added to the required duties to grant corresponding roles appropriate access levels.

Enabling role-based security for business events

Role-based security for business events must be enabled via Feature management.

1. Go to **System administration > Feature management**.
2. Select the **Business events catalog security** feature.
3. Enable the feature.
4. Go to the business events catalog via **System administration > Set up > Business events > Business events catalog.**
5. The **Security** tab in the catalog is where a business event must be mapped to one or more roles. You must complete the configuration as required.
6. Enable security by selecting the **Enable** menu button on the **Security** menu on the top navigation pane. An informational message will confirm if security is enabled or disabled.
7. Modify the necessary security role to add the appropriate privilege or the duty based on security noted in the informational message.

Subscribe to business events from service

Users having access to the privilege **Subscribe to business events from service** via their roles will be able to only see and subscribe to business events that have been assigned to their roles, which is described below. The organizational assignments that are done, if any, as part of role-based security is honored in the context of business events by letting users to only subscribe to business events in the organizations to which they have access to via their roles. This behavior is effective using any service calls like from Power Automate or Logic Apps.

Backward compatibility

To ensure backward compatibility of business events with versions prior to Platform update 29, the following behavior must be understood.

- Role-based security for business events will be disabled by default.
- Even if the feature is enabled in Feature management, role-based security will not take effect.

- Role-based security must be explicitly enabled in the business events catalog via the **Security** menu.
- After role-based security is enabled completely, security will be enforced henceforth. This will mean that any user with administration role will not notice any change in behavior. However, any non-admin users will either only see business events to which their roles were assigned to in the business events catalog security configuration or they will not see any business events because their roles were not assigned to any business events.

> **NOTE**
>
> To ensure uninterrupted functionality, it is important to understand the backward-compatibility behavior described above before you enable security on business events.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Application business events

2/18/2021 • 2 minutes to read • Edit Online

This topic lists application business events.

Procure to pay

BUSINESS EVENT	DESCRIPTION	MODULE	
Vendor invoice matched	This event is triggered when invoice matching validation is completed for a vendor invoice as part of the Procure to pay process.	Accounts payable	
Vendor invoice posted	This event business event is triggered when a user posts a vendor invoice as part of the Procure to Pay process.	Accounts payable	
Vendor payment posted	This event is triggered when a user posts a vendor payment as part of the Procure to pay process.	Accounts payable	
Invoice register journal posted	This event is triggered when a user posts an invoice register journal as part of the Procure to pay process.	Accounts payable	
Invoice journal posted	This event is triggered when a user posts an invoice journal as part of the Procure to pay process.	Accounts payable	
Invoice approval journal posted	This event is triggered when a user posts an invoice approval journal as part of the Procure to pay process.	Accounts payable	

BUSINESS EVENT	DESCRIPTION	MODULE	
Purchase order confirmed	This event is triggered when a purchase order is confirmed by a vendor. One of the following actions triggers the event: the user manually confirms a purchase order in the user interface for purchase orders, when the purchase order confirmation is executed in a batch, or when the confirmation is executed programmatically in intercompany scenarios. In scenarios where vendor collaboration is used, and the vendor collaboration policy is set to autoconfirm a purchase order, the trigger occurs when the **Accept** button is clicked on the **Purchase order confirmation** page in the **Vendor collaboration** portal.	Procurement and sourcing	
Purchase order received	This event is triggered when goods or services are registered as received against one or more purchase orders. One of the following actions triggers the event: a product receipt is generated for one or more purchase orders manually in the user interface for purchase orders and product receipts, when product receipts are generated in a batch, or when product receipts are generated programmatically in intercompany scenarios.	Procurement and sourcing	

Quote to cash

BUSINESS EVENT	DESCRIPTION	MODULE
Invoice is created from a sales order	This event is triggered when a user posts a sales order invoice as part of the Quote to Cash process.	Accounts receivable
Free text invoice posted	This event is triggered when a user posts a free text invoice as part of the Quote to Cash process.	Accounts receivable

BUSINESS EVENT	DESCRIPTION	MODULE
Payment posted	This event is triggered when a user posts a payment as part of the Quote to Cash process.	Accounts receivable
Transaction is written off	This event is triggered when a user writes off a customer transaction as part of the Quote to Cash process.	Accounts receivable
Collection status of a transaction changed	This event is triggered when a user updates the collection status of a transaction as part of the Quote to Cash process.	Accounts receivable
Interest note posted	This event is triggered when a user posts an interest note as part of the Quote to Cash process.	Accounts receivable
Collection letter created	This event is triggered when a user creates a collection letter for a customer as part of the Quote to Cash process.	Credit and collections

Workflow business events

2/18/2021 • 6 minutes to read • Edit Online

Workflow business events are generated at various points in the processing of a workflow.

Workflow construction

To construct a workflow, a developer can define workflows components in metadata and code in the Visual Studio tools.

An administrator can create workflows in the web client and then design them in the workflow designer. For more information, see Create workflows.

Workflow components

Workflow components are defined in metadata as:

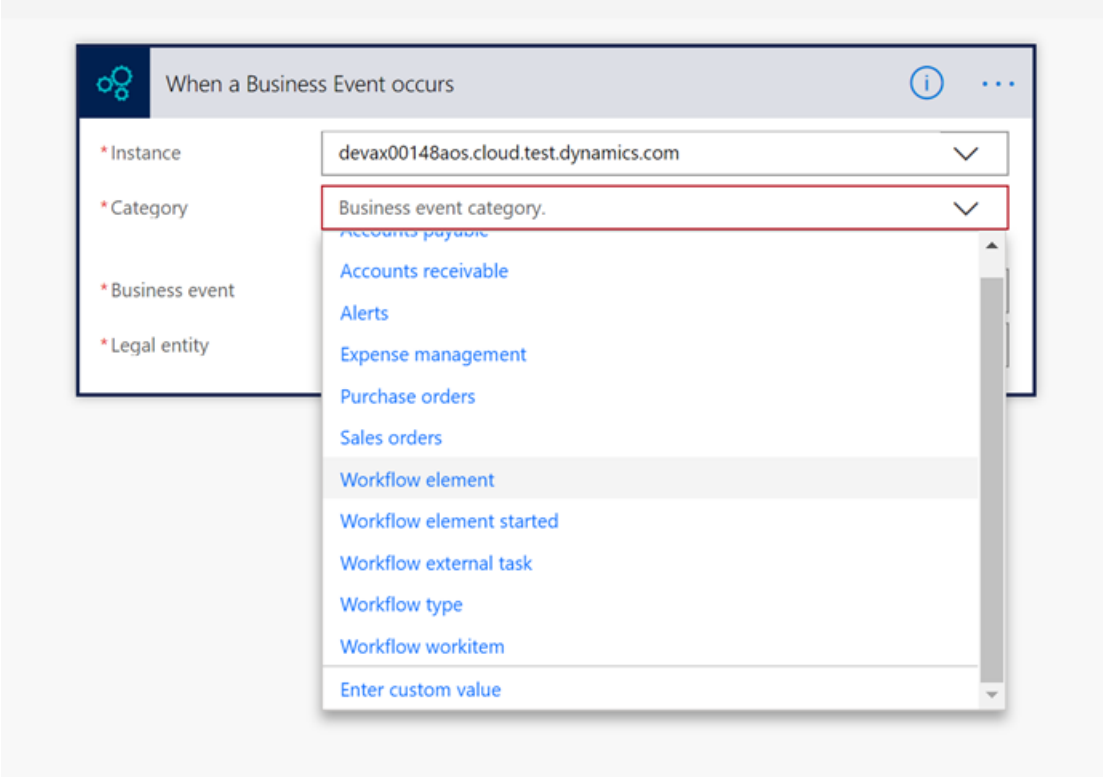
- **Workflow types** - Also known as templates, workflow types define the elements allowed in a workflow. The administrator decides which elements are actually used when they create the workflow design.
  - In the Application Explorer, go to **AOT > Business Process and Workflow > Workflow Types**.
- **Workflow elements** - Workflow elements are the executable pieces that make up a workflow. For details, see Workflow elements.
  - Tasks (aka manual tasks) - Go to **AOT > Business Process and Workflow > Workflow Tasks**.
  - Approvals - Go to **AOT > Business Process and Workflow > Workflow Approvals**.
  - Automated tasks - Go to **AOT > Business Process and Workflow > Workflow Automated Tasks**.

Workflow runtime

After a workflow is submitted by a user, it is added to a queue and run using the **Workflow message processing** batch job. As the workflow runs, it will progress through all the connected workflow elements until it reaches the end. When the workflow runtime encounters a manual task element, it will create a work item for the user assigned to the task. When the workflow runtime encounters an approval element, it will create a work item for each user assigned to each approval step.

Workflow business event categories

There are five different categories of workflow business events. The category will show up in Microsoft Power Automate to help with event selection.



- **Category: Workflow type**
  - These events will fire on workflow events like started and completed. All workflow instances will be represented in this category.
  - **ID format** - "Workflow_" + Workflow name + Workflow instance ID, for example "Workflow_BudgetPlanReview_000002"
  - **Name format** - Workflow label + " (" + Workflow instance ID ")", for example "Prepare department budget (000002)"
- **Category: Workflow element started**
  - These events will fire when a workflow element is started. All enabled workflow elements within a workflow instance will be represented in this category.
  - **ID format** - "Workflow_" + Workflow name + Workflow instance ID + "_" + Workflow element name + "_Started", for example "Workflow_BudgetPlanReview_000002_BudgetActivateBudgetPlanChild_Started"
  - **Name format** - Workflow label + " (" + Workflow instance ID ") - " + Workflow element label, for example "Prepare department budget (000002) - Activate associated budget plan"
- **Category: Workflow element**
  - These events will fire on workflow element events other than started, such as completed. All enabled workflow elements within a workflow instance will be represented in this category.
  - **ID format** - "Workflow_" + Workflow name + Workflow instance ID + "_" + Workflow element name, for example "Workflow_BudgetPlanReview_000002_BudgetActivateBudgetPlanChild"
  - **Name format** - Workflow label + " (" + Workflow instance ID ") - " + Workflow element label, for example "Prepare department budget (000002) - Activate associated budget plan"
- **Category: Workflow external task**
  - These events will fire when a workflow automated task element is started. All enabled workflow automated task elements within a workflow instance will be represented in this category.
  - **ID format** - "Workflow_" + Workflow name + Workflow instance ID + "_" + Workflow element name + "_ExternalTask", for example "Workflow_BudgetPlanReview_000002_BudgetActivateBudgetPlanChild_ExternalTask"
  - **Name format** - Workflow label + " (" + Workflow instance ID ") - " + Workflow element label, for example "Prepare department budget (000002) - Activate associated budget plan"

- **Category: Workflow workitem**
  - These events will fire when a workflow work item is created for a user. All enabled workflow tasks and workflow approvals within a workflow instance will be represented in this category.
  - **ID format** - "Workflow_" + Workflow name + Workflow instance ID + "_" + Workflow element name + "_WorkItem", for example "Workflow_BudgetPlanReview_000002_BudgetActivateBudgetPlanChild_WorkItem"
  - **Name format** - Workflow label + " (" + Workflow instance ID ") - " + Workflow element label, for example "Prepare department budget (000002) - Activate associated budget plan"

Completion of a work item in Power Automate

Workflow business events are a good target for triggering approval flows. The **workflow workitem** event can be used in conjunction with the validate and complete OData actions to facilitate completion of a work item in Power Automate.

An approval or task work item can be completed in Power Automate using the following steps:

- Trigger the Power Automate using the **when a business event occurs** trigger targeting the appropriate **workflow workitem** event.
- Validate that the **workflow workitem** contains a valid set of information so it is ready for completion by calling the **Validate** method on the **WorkflowWorkItems** entity.
- If the workitem is not ready for completion, then send a notification to the assigned user to let them know that there is a workitem that needs their attention.
- If the workitem is ready for completion, then request a response from the assigned user by sending the available response options to the user.
- After a response is provided, complete the workitem with that response by calling the **Complete** method on the **WorkflowWorkItems** entity.

To enable external completion of work items, the work item action manager class needs to implement the IValidateWorkflowWorkItemAction interface. The standard WorkflowWorkItemActionManager class has implemented this interface. In Platform update 32, the TrvWorkflowWorkItemActionManager class was updated to implement the IValidateWorkflowWorkItemAction interface. Use the existing IValidateWorkflowWorkItemAction implementations as examples to notify updates about other WorkflowWorkItemActionManager classes.

For a step-by-step guide to setting up work item completion in Microsoft Power Automate, see Consume workflow approval business events.

Templates for work item completion in Power Automate

The following templates for work item completion in Power Automate are available:

- Complete Dynamics 365 for Finance and Operations workflow work items (PU26)
- Complete Dynamics 365 for Finance and Operations workflow work items (PU29)

The Platform update 29 version gets completion options from the business event payload. These options were added in Platform update 29, and presented to the user via the approval action.

Troubleshooting workflow business events

Troubleshooting workflow issues

Ensure that the workflow is running correctly and creating work items as expected. If the workflow doesn't work inside the application so that state changes are occurring, then the events won't occur. Adjust the workflow configuration as needed. If needed, review the workflow details in the **Workflow History** form.

Troubleshooting Power Automate issues

Ensure that the Power Automate subscription is available in the **System administration > Setup > Business events > Business events catalog** on the **Active events** tab. If the Power Automate subscription isn't there, then check Power Automate and recreate it if needed.

Troubleshooting business events issues

Ensure that other business events are occurring by creating a Power Automate to trigger off another business event. For example, the Free Text Invoice Posted event can be triggered by simply creating a Free Text Invoice with a single line and posting it. For more information, see Troubleshoot business events.

Troubleshooting work item approval via Power Automate

If a flow is trying to handle approval for work items, but it isn't firing, then verify these steps:

- Are the work items being created so the applicable user can see them waiting for approval in the web client?
- Is the event subscription from Flow visible in the Business Events form?
- Are the workflow configuration and the event subscription from Flow for the correct legal entity (company)?

Alerts as business events

2/18/2021 • 2 minutes to read • Edit Online

There are two kinds of alerts that can be configured by users. These are change-based alerts and due date alerts. For more information about the alerts functionality, see Alerts.

The change-based alerts and due date alerts can be configured to send out a business event as a mechanism to notify or trigger external applications or systems. This allows alerts to participate in advanced user notification scenarios and also in business process integration across systems.

To generate a business event from an alert, in the **Create alert rule** dialog box, set **Alert me with** > **Send externally** to **Yes**.

In order for alerts to be processed, the batch processes for change-based and/or due-date alerts should be set for batch processing for due-date events. For more information, see Batch processing for due-date events.

The business event for the change-based alert and/or the due date alert must also be active for the alert to be sent out as a business event. To learn more about the activation process, see Activating business events.

Business events in Microsoft Power Automate

2/18/2021 • 2 minutes to read • Edit Online

Business events can be consumed in Microsoft Power Automate via the application connector. The connector has a trigger that is named **when a business event occurs**. This trigger can be used to subscribe to any of the business events that are available in the target instance of the application.

Prerequisite

It's important that you understand business events. For more information, see the Business events documentation.

> **IMPORTANT**
>
> The **when a business event happens** trigger works only with application instances that run Platform update 24 or later, because business events aren't available in earlier platform releases.

Subscribing to business events and unsubscribing from them

After the **when a business event happens** trigger is added to a flow, the following information must be provided:

- **Instance** – Specify the host name of the instance where business events occur. Environment instances should be available in the provided drop-down menu, but if an environment is not listed it can be entered as a custom value.
- **Category** – Select the category of business events. The **Business event** field then shows the business events in that category.
- **Business event** – The available business events in the selected category.
- **Legal entity** – Specify the legal entity where the business event is being subscribed to. The flow will be triggered when the business event occurs in that legal entity. By default, this field is blank and the business event is subscribed to in **all** legal entities.

When the flow is saved, a subscription to the selected business event is added into the environment instance. As part of the subscription process, the required endpoint is set up, and the corresponding business event is activated.

If the trigger is deleted or the flow is turned off, then the business event endpoint will be automatically deleted.

Multiple flows can subscribe to the same business event in different legal entities or in the same legal entity. Note that the default endpoint limit per event is ten. If needed, adjust the **Endpoints allowed per event** on the **Business event parameters** page.

> **NOTE**
>
> The Power Automate endpoint must not be configured manually. The endpoint will automatically get created from Power Automate as explained above.

For how-to information about using business events in Microsoft Flow, see Consume business events in Microsoft Flow.

Other ways to consume business events in Power Automate

The previous section explains how you can subscribe to business events directly from Power Automate by using the trigger in the connector. However, you can also consume business events in Microsoft Power Automate from Microsoft Azure Event Grid, by using the Event Grid connector for Microsoft Power Automate.

Event Grid might be a viable approach for consuming business events in Power Automate if it's already being used for other integrations in an implementation. If a business event in the same legal entity must trigger multiple flows, you should consider consuming the business event from Event Grid.

This approach is applicable to any messaging or event platform that is used as an endpoint for business events, provided that a connector is available for it in Power Automate.

Business events developer documentation

2/18/2021 • 19 minutes to read • Edit Online

This topic walks you through the development process and best practices for implementing business events.

What is a business event, and what isn't a business event?

This question comes up every time that we start to think about use cases where business events can help. Is the creation of a vendor a business event? Is confirmation of a purchase order a business event? Is it a business event if you capture the event at the table level? Or should business events be captured only at the business logic level in a business process? These questions aren't just valid, but they are also a key topic of discussion when a solution is planned and architected for integration. The following guidelines can help with this thought process and decision making.

Intent

The intent behind capturing a business event must be clearly understood. In other words, what is the reason for capturing the business event, and how it will be used by the recipient?

If your intent is to capture a business event so that you can take a business action outside Dynamics 365 Finance and Operations apps in response to a business event that occurs in Finance and Operations, you have a good use case for business events. The business action that is taken in response to the business event can be to notify users about the business event and/or to call into another business application to take a business action, such as creation of a sales order. It's important that you look at the business action generically and not base the need for a business event on the type of business action that will be taken.

If your intent is to transfer data to a recipient and, in effect, realize a data export scenario, you don't have a good use case for business events. In fact, the use of business events for data transfer scenarios is a misuse of the business events framework. Such scenarios must continue to use data export mechanisms that are already available in data management.

Fidelity

When the intent is clear, and a legitimate need for a business event is established, the next step is to evaluate the approach that must be used to capture the business event. This section summarizes the approach that must be evaluated.

Regardless of the approach that is used, the fidelity of business events is significant, because it helps guarantee that the following aspects are taken care of. Therefore, it must influence the design choice for implementing the business event. However, the design choice that you make to implement a business event must not influence the concept of business events. In other words, the chosen design must not be used as a decision-making tool, to determine whether an event is a business event. The intent must be used to make those decisions.

- **Durable business events** – No false business events should be sent to the recipient. If a purchase order confirmation business event is sent out, the recipient expects and must trust that the purchase order was really confirmed. The design choice must help guarantee this transactional nature. Therefore, you must not make a design choice that violates the recipient's expectations.
- **Targeted** – Business events must be designed to optimize the consumption story for the recipient. In other words, you should make it as easy as possible for the recipient to consume business events. Therefore, business events must be as specific as possible and must be targeted to specific use cases. They must not be generic, so that the consumer has to determine what the business event is for by trying to understand the

payload. The design choice must allow for preservation of targeted business events.

- **Noiseless** – The design should include very little effort to filter out noise. To make business events very specific, avoid writing filtering logic to filter out conditions that don't match the expected business event. The chosen approach must help guarantee that the business event is implemented in code at a sufficiently specific point so that no filtering of noise is required. Any attempt to filter noise by adding logic can affect performance and might also become complicated in specific use cases.

The following table compares capture at the business and table levels.

CAPTURE AT THE BUSINESS LOGIC LEVEL	CAPTURE AT THE TABLE LEVEL
Capture at this level helps guarantee durability because it occurs in the transaction.	Capture at this level helps guarantee durability because it occurs in the transaction.
Capture at this level allows for targeted business events.	Because events are captured at a lower level, it's difficult to provide targeted business events.
It's easy to remain noiseless.	It's difficult to remain noiseless unless additional effort is made to implement sound logic that filters out noise.
Capture at this level provides additional context for the business process, and can significantly improve the durability and quality of the payload.	Because events are captured at a lower level, business process context is probably lost.

> **NOTE**
> In general, if you implement business events at the table level, you might face other challenges, in addition to the challenges that are described in the preceding table. For example, if the business logic is run via a stored procedure that updates data in the underlying table, the business event might not even be generated, because it was implemented in the table insert method in X++. You might encounter additional challenges in specific use cases. Therefore, we don't recommend that you implement business events at the table level.

Implement a business event

The process for implementing a business event and sending it is fairly straightforward.

1. Build the contract.
2. Build the event.
3. Add code to send the event.

Two classes must be implemented:

- **Business event** – This class extends the **BusinessEventsBase** class. It supports constructing the business event, building the payload, and sending the business event.
- **Business event contract** – This class extends the **BusinessEventsContract** class. It defines the payload of the business event and allows for population of the contract at runtime.

BusinessEventsBase extension

Naming convention

The names of business events should follow the pattern <noun or noun phrase><past tense action>BusinessEvent. The <noun or noun phrase> part of the name should comply with existing definitions for application area prefixes.

Examples

- VendorInvoicePostedBusinessEvent
- CollectionLetterSentBusinessEvent

Implementation

The process of implementing an extension of the **BusinessEventsBase** class is straightforward. It involves extending the **BusinessEventsBase** class, and implementing a static constructor method, a private **new** method, methods to maintain internal state, and the **buildContract** method.

1. Implement a static **newFrom<my_buffer>** method. The <my_buffer> part of the method name is typically the table buffer that is used to initialize the business event contract.

```
static public SalesInvoicePostedBusinessEvent
newFromCustInvoiceJour(CustInvoiceJour _custInvoiceJour)
{
    SalesInvoicePostedBusinessEvent businessEvent = new
    SalesInvoicePostedBusinessEvent();
    businessEvent.parmCustInvoiceJour(_custInvoiceJour);
    return businessEvent;
}
```

2. Extend the **BusinessEventsBase** class.

```
[BusinessEvents(classStr(SalesInvoicePostedBusinessEventContract),
'AccountsReceivable:SalesOrderInvoicePostedBusinessEventName','AccountsReceivable:SalesOrderInvoicePo
stedBusinessEventDescription',ModuleAxapta::SalesOrder)]
public class SalesInvoicePostedBusinessEvent extends BusinessEventsBase
```

   Note the **BusinessEvents** attribute. This attribute provides the business events framework with information about the business event's contract, name, and description. It also provides the module that the business event is part of. Labels must be defined for the name and description arguments. However, these labels should be referenced without the at symbol (@) to avoid storing localized data.

3. Implement a private **new** method. This method is called only from the static constructor method.

```
private void new()
{
}
```

4. Implement private **parm** methods to maintain internal state.

```
private CustInvoiceJour parmCustInvoiceJour(CustInvoiceJour _custInvoiceJour = custInvoiceJour)
{
    custInvoiceJour = _custInvoiceJour;
    return custInvoiceJour;
}
```

5. Implement the **buildContract** method. Note that you need an **EventContract** stub for this step.

```
[Wrappable(true), Replaceable(true)]
public BusinessEventsContract buildContract()
{
    return
    SalesInvoicePostedBusinessEventContract::newFromCustInvoiceJour(custInvoiceJour);
}
```

   For extensibility, the **buildContract** method must have the **Wrappable(true)** and **Replaceable(true)**

attributes. The **buildContract** method is called only when a business event is enabled for a company.

Here is the complete implementation of the "Sales order invoice posted" business event.

```
/// <summary>
/// Sales order invoice posted business event.
/// </summary>
[BusinessEvents(classStr(SalesInvoicePostedBusinessEventContract),
'AccountsReceivable:SalesOrderInvoicePostedBusinessEventName',
'AccountsReceivable:SalesOrderInvoicePostedBusinessEventDescription',
ModuleAxapta::SalesOrder)]
public class SalesInvoicePostedBusinessEvent extends BusinessEventsBase
{
    private CustInvoiceJour custInvoiceJour;
    private CustInvoiceJour parmCustInvoiceJour(CustInvoiceJour _custInvoiceJour =
    custInvoiceJour)
    {
        custInvoiceJour = _custInvoiceJour;
        return custInvoiceJour;
    }
    /// <summary\>
    /// Creates a SalesInvoicePostedBusinessEvent from a CustInvoiceJour record.
    /// <summary>
    /// param name = "_custInvoiceJour"> CustInvoiceJour record <param>
    /// <returns>A SalesInvoicePostedBusinessEvent </returns>
    static public SalesInvoicePostedBusinessEvent
    newFromCustInvoiceJour(CustInvoiceJour _custInvoiceJour)
    {
        SalesInvoicePostedBusinessEvent businessEvent = new
        SalesInvoicePostedBusinessEvent();
        businessEvent.parmCustInvoiceJour(_custInvoiceJour);
        return businessEvent;
    }
    private void new()
    {
    }
    [Wrappable(true), Replaceable(true)]
    public BusinessEventsContract buildContract()
    {
        return SalesInvoicePostedBusinessEventContract::newFromCustInvoiceJour(custInvoiceJour);
    }
}
```

BusinessEventsContract extension

A business event contract class extends the **BusinessEventsContract** class. It defines and populates the payload of the business event. Although there is some variation across business events, the basic structure of the business event contract is consistent.

The process of implementing a business event contract involves extending the **BusinessEventContract** class, defining internal state, implementing an initialization method, implementing a static constructor method, and implementing **parm** methods to access the contract state.

1. Extend the **BusinessEventContract** class.

   ```
   [DataContract]
   public final class SalesInvoicePostedBusinessEventContract extends
   BusinessEventsContract
   ```

   The class must have the **DataContract** attribute.

2. Add private variables to hold the contract state.

```
private CustInvoiceAccount invoiceAccount;
private CustInvoiceId invoiceId;
private SalesIdBase salesId;
private TransDate invoiceDate;
private DueDate invoiceDueDate;
private AmountMST invoiceAmount;
private TaxAmount invoiceTaxAmount;
private LegalEntityDataAreaId legalEntity;
```

3. Implement a private initialization method.

```
private void initialize(CustInvoiceJour _custInvoiceJour)
{
    invoiceAccount = _custInvoiceJour.InvoiceAccount;
    invoiceId = _custInvoiceJour.InvoiceId;
    salesId = _custInvoiceJour.SalesId;
    invoiceDate = _custInvoiceJour.InvoiceDate;
    invoiceDueDate = _custInvoiceJour.DueDate;
    invoiceAmount = _custInvoiceJour.InvoiceAmountMST;
    invoiceTaxAmount = _custInvoiceJour.SumTaxMST;
    legalEntity = _custInvoiceJour.DataAreaId;
}
```

The **initialize** method is responsible for setting the private state of the business event contract class, based on data that is provided through the static constructor method.

4. Implement a static constructor method.

```
public static SalesInvoicePostedBusinessEventContract
newFromCustInvoiceJour(CustInvoiceJour _custInvoiceJour)
{
    var contract = new SalesInvoicePostedBusinessEventContract();
    contract.initialize(_custInvoiceJour);
    return contract;
}
```

The static constructor method calls a private **initialize** method to initialize the private class state.

5. Implement **parm** methods to access the contract state.

```
[DataMember('InvoiceAccount'), BusinessEventsDataMember("@AccountsReceivable:InvoiceAccount")]
public CustInvoiceAccount parmInvoiceAccount(CustInvoiceAccount _invoiceAccount = invoiceAccount)
{
    invoiceAccount = _invoiceAccount;
    return invoiceAccount;
}
```

The **parm** methods should have the **DataMember('<name>')** and **BusinessEventsDataMember('<description>')** attributes. The name that you provide on the **DataMember** attribute (for example, **'InvoiceAccount'**) will be visible to data contract consumers. The description that you provide in the **BusinessEventsDataMember** attribute will be visible in the Business Events catalog user interface (UI) and used to describe this contract's data members.

NOTE

- **RecId** values should not be part of a business event's payload. Use the alternate key (AK) instead.
- Enumeration (enum) values must be converted to their symbol value before they can be published. Use the **enum2Symbol** method to convert an enum's value to the symbol string. Here is an example:

```
status = enum2Symbol(enumNum(CustVendDisputeStatus), _custDispute.Status);
```

In some cases, population of the data contract's internal state requires that you implement additional retrieval methods. These retrieval methods should be implemented as private methods, and they should be called from the **initialize** method.

Here is the complete implementation of the "Sales order invoice posted" business event contract.

```
/// <summary>
/// The data contract for a SalesInvoicePostedBusinessEvent
/// </summary>
[DataContract]
public final class SalesInvoicePostedBusinessEventContract extends
BusinessEventsContract
{
    private CustInvoiceAccount invoiceAccount;
    private CustInvoiceId invoiceId;
    private SalesIdBase salesId;
    private TransDate invoiceDate;
    private DueDate invoiceDueDate;
    private AmountMST invoiceAmount;
    private TaxAmount invoiceTaxAmount;
    private LegalEntityDataAreaId legalEntity;
    /// <summary>
    /// Creates a SalesInvoicePostedBusinessEventContract from a CustInvoiceJour record.
    /// </summary>
    /// <param name = "_custInvoiceJour"> CustInvoiceJour record</param>
    /// <returns>A SalesInvoicePostedBusinessEventContract </returns>
    public static SalesInvoicePostedBusinessEventContract
    newFromCustInvoiceJour(CustInvoiceJour _custInvoiceJour)
    {
        var contract = new SalesInvoicePostedBusinessEventContract();
        contract.initialize(_custInvoiceJour);
        return contract;
    }
    private void initialize(CustInvoiceJour _custInvoiceJour)
    {
        invoiceAccount = _custInvoiceJour.InvoiceAccount;
        invoiceId = _custInvoiceJour.InvoiceId;
        salesId = _custInvoiceJour.SalesId;
        invoiceDate = _custInvoiceJour.InvoiceDate;
        invoiceDueDate = _custInvoiceJour.DueDate;
        invoiceAmount = _custInvoiceJour.InvoiceAmountMST;
        invoiceTaxAmount = _custInvoiceJour.SumTaxMST;
        legalEntity = _custInvoiceJour.DataAreaId;
    }
    private void new()
    {
    }
    [DataMember('InvoiceAccount'), BusinessEventsDataMember("@AccountsReceivable:InvoiceAccount")]
    public CustInvoiceAccount parmInvoiceAccount(CustInvoiceAccount _invoiceAccount
    = invoiceAccount)
    {
        invoiceAccount = _invoiceAccount;
        return invoiceAccount;
    }
    [DataMember('InvoiceId'), BusinessEventsDataMember("@AccountsReceivable:BusinessEventInvoiceId")]
```

```
    [DataMember('InvoiceId'), BusinessEventsDataMember("@AccountsReceivable:BusinessEventInvoiceId")]
    public CustInvoiceId parmInvoiceId(CustInvoiceId _invoiceId = invoiceId)
    {
    invoiceId = _invoiceId;
    return invoiceId;
    }
    [DataMember('SalesOrderId'), BusinessEventsDataMember("@AccountsReceivable:SalesOrderId")]
    public SalesIdBase parmSaleOrderId(SalesIdBase _salesId = salesId)
    {
        salesId = _salesId;
        return salesId;
    }
    [DataMember('InvoiceDate'), BusinessEventsDataMember("@AccountsReceivable:BusinessEventInvoiceDate")]
    public TransDate parmInvoiceDate(TransDate _invoiceDate = invoiceDate)
    {
        invoiceDate = _invoiceDate;
        return invoiceDate;
    }
    [DataMember('InvoiceDueDate'), BusinessEventsDataMember("@AccountsReceivable:InvoiceDueDate")]
    public DueDate parmInvoiceDueDate(DueDate _invoiceDueDate = invoiceDueDate)
    {
        invoiceDueDate = _invoiceDueDate;
        return invoiceDueDate;
    }
    [DataMember('InvoiceAmountInAccountingCurrency'),
BusinessEventsDataMember("@AccountsReceivable:InvoiceAmountInAccountingCurrency")]
    public AmountMST parmInvoiceAmount(AmountMST _invoiceAmount = invoiceAmount)
    {
        invoiceAmount = _invoiceAmount;
        return invoiceAmount;
    }
    [DataMember('InvoiceTaxAmount'), BusinessEventsDataMember("@AccountsReceivable:InvoiceTaxAmount")]
    public TaxAmount parmInvoiceTaxAmount(TaxAmount _invoiceTaxAmount =
    invoiceTaxAmount)
    {
        invoiceTaxAmount = _invoiceTaxAmount;
        return invoiceTaxAmount;
    }
    [DataMember('LegalEntity'), BusinessEventsDataMember("@AccountsReceivable:LegalEntity")]
    public LegalEntityDataAreaId parmLegalEntity(LegalEntityDataAreaId _legalEntity
    = legalEntity)
    {
        legalEntity = _legalEntity;
        return legalEntity;
    }
}
```

Sending a business event

You must modify application code so that it sends the business event at the appropriate point. Often, you can use a common point in a framework. Documents that extend **SourceDocument** have a common point for creating and sending a business event. For more information, see the Source document framework support section later in this topic.

Other frameworks also provide common points for sending business events. For example, the **CustVendVoucher** class hierarchy in the Application Object Tree (AOT) has a **post** method that is used to send business events that are related to posting customer or vendor vouchers. Overrides of the base class implementation provide specialization of the logic for sending business events. For an example, see **CustVoucher.createBusinessEvent** or **VendVoucher.createBusinessEvent** in the AOT.

The sending of a business event is linked to the commit of the underlying transaction. If the underlying transaction is aborted, the business event won't be sent. Therefore, applications can send the business event at the point where the payload information is available.

The business events framework determines whether a business event is published to a consumer. As a general rule, applications should always send a business event, regardless of whether the business event is enabled. If significant additional logic is required, or if the logic for sending a business event has a performance impact, an application can check whether a specific business event is enabled before it runs business logic that is associated with sending business events. This check is done through the **BusinessEventsConfigurationReader::isBusinessEventEnabled** method.

```
if
(BusinessEventsConfigurationReader::isBusinessEventEnabled(classStr(CollectionStatusUpdatedBusinessEvent)))
{
    while select dispute
    where dispute.Status == CustVendDisputeStatus::PromiseToPay
    && dispute.FollowUpDate _currentDate
    exists join custTrans
    where custTrans.RecId == dispute.CustTrans
    && !custTrans.Closed
    exists join _tmpCustAging
    where _tmpCustAging.AccountNum == custTrans.AccountNum
    {
        CollectionStatusUpdatedBusinessEvent::newFromCustDispute(dispute).send();
    }
}
```

Source document framework support

The source document framework supports sending business events automatically as part of the transition from an in-process state to a completed state for the document. To take advantage of this capability, documents that extend the source document framework must implement an extension of the **SourceDocumentStateInProcess.getBusinessEvent** method to create and return the correct **BusinessEventsBase** extension type.

Extending a business event payload

You might want to publish additional information as part of the payload of a business event. To send this additional information, you must extend the business event's standard payload.

Example scenario

This example shows how to extend the **CustFreeTextInvoicePostedBusinessEventContract** class so that it includes a customer classification. This customer classification is an industry-based custom classification.

Step 1: Create an extended business event contract

Create a contract that consists of the standard business event contract plus any additional information that must be included in the payload.

```
[DataContract]
public class CustFreeTextInvoicePostedBusinessEventExtendedContract
extends BusinessEventsContract
{
    // standard contract
    private CustFreeTextInvoicePostedBusinessEventContract
    custFreeTextInvoicePostedBusinessEventContract;
    // contract extensions
    private str customerClassification;
}
```

Step 2: Create an initialize method

Create an **initialize** method that initializes the value of the private contract.

```
private void initialize(CustFreeTextInvoicePostedBusinessEventContract
_custFreeTextInvoicePostedBusinessEventContract)
{
    custFreeTextInvoicePostedBusinessEventContract =
    _custFreeTextInvoicePostedBusinessEventContract;
}
```

Step 3: Create a static newFrom method

Create a static **newFrom** method that takes the standard contract as an argument and calls the **initialize** method.

```
public static CustFreeTextInvoicePostedBusinessEventExtendedContract
newFromCustFreeTextInvoicePostedBusinessEventContract(CustFreeTextInvoicePostedBusinessEventContract
_custFreeTextInvoicePostedBusinessEventContract)
{
    var contract = new CustFreeTextInvoicePostedBusinessEventExtendedContract();
    contract.initialize(_custFreeTextInvoicePostedBusinessEventContract);
    return contract;
}
```

Step 4: Map parm methods

Copy the **parm** methods from the standard data contract, and modify each method so that it gets and sets values in the class's standard contract instance.

```
[DataMember('InvoiceAccount')]
public CustInvoiceAccount parmInvoiceAccount(CustInvoiceAccount _invoiceAccount
= custFreeTextInvoicePostedBusinessEventContract.parmInvoiceAccount())
{
    return
    custFreeTextInvoicePostedBusinessEventContract.parmInvoiceAccount(_invoiceAccount);
}
[DataMember('InvoiceId')]
public CustInvoiceId parmInvoiceId(CustInvoiceId _invoiceId =
custFreeTextInvoicePostedBusinessEventContract.parmInvoiceId())
{
    return custFreeTextInvoicePostedBusinessEventContract.parmInvoiceId(_invoiceId);
}
```

Step 5: Add parm methods for additional payload data

```
[DataMember('CustomerClassification')]
public CustomerClassification parmCustomerClassification(CustomerClassification
_customerClassification = customerClassification)
{
    customerClassification = _customerClassification;
    return customerClassification;
}
```

Here is the complete implementation of the extended business contract.

```
[DataContract]
public class CustFreeTextInvoicePostedBusinessEventExtendedContract
extends BusinessEventsContract
{
    // standard contract
    private CustFreeTextInvoicePostedBusinessEventContract
    custFreeTextInvoicePostedBusinessEventContract;
    // contract extensions
    private str customerClassification;
    public static CustFreeTextInvoicePostedBusinessEventExtendedContract
```

```
newFromCustFreeTextInvoicePostedBusinessEventContract(CustFreeTextInvoicePostedBusinessEventContract
_custFreeTextInvoicePostedBusinessEventContract)
{
    var contract = new CustFreeTextInvoicePostedBusinessEventExtendedContract();
    contract.initialize(_custFreeTextInvoicePostedBusinessEventContract);
    return contract;
}
private void initialize(CustFreeTextInvoicePostedBusinessEventContract
_custFreeTextInvoicePostedBusinessEventContract)
{
    custFreeTextInvoicePostedBusinessEventContract =
    _custFreeTextInvoicePostedBusinessEventContract;
}
private void new()
{
}
[DataMember('InvoiceAccount')]
public CustInvoiceAccount parmInvoiceAccount(CustInvoiceAccount _invoiceAccount
= custFreeTextInvoicePostedBusinessEventContract.parmInvoiceAccount())
{
    return
    custFreeTextInvoicePostedBusinessEventContract.parmInvoiceAccount(_invoiceAccount);
}
[DataMember('InvoiceId')]
public CustInvoiceId parmInvoiceId(CustInvoiceId _invoiceId =
custFreeTextInvoicePostedBusinessEventContract.parmInvoiceId())
{
    return custFreeTextInvoicePostedBusinessEventContract.parmInvoiceId(_invoiceId);
}
[DataMember('InvoiceDate')]
public TransDate parmInvoiceDate(TransDate _invoiceDate =
custFreeTextInvoicePostedBusinessEventContract.parmInvoiceDate())
{
    return
    custFreeTextInvoicePostedBusinessEventContract.parmInvoiceDate(_invoiceDate);
}
[DataMember('InvoiceDueDate')]
public DueDate parmInvoiceDueDate(DueDate _invoiceDueDate =
custFreeTextInvoicePostedBusinessEventContract.parmInvoiceDueDate())
{
    return
    custFreeTextInvoicePostedBusinessEventContract.parmInvoiceDueDate(_invoiceDueDate);
}
[DataMember('InvoiceAmountInAccountingCurrency')]
public AmountMST parmInvoiceAmount(AmountMST _invoiceAmount =
custFreeTextInvoicePostedBusinessEventContract.parmInvoiceAmount())
{
    return
    custFreeTextInvoicePostedBusinessEventContract.parmInvoiceAmount(_invoiceAmount);
}
[DataMember('InvoiceTaxAmount')]
public TaxAmount parmInvoiceTaxAmount(TaxAmount _invoiceTaxAmount =
custFreeTextInvoicePostedBusinessEventContract.parmInvoiceTaxAmount())
{
    return
    custFreeTextInvoicePostedBusinessEventContract.parmInvoiceTaxAmount(_invoiceTaxAmount);
}
[DataMember('LegalEntity')]
public LegalEntityDataAreaId parmLegalEntity(LegalEntityDataAreaId _legalEntity
= custFreeTextInvoicePostedBusinessEventContract.parmLegalEntity())
{
    return
    custFreeTextInvoicePostedBusinessEventContract.parmLegalEntity(_legalEntity);
}
// contract extensions
[DataMember('CustomerClassification')]
public CustomerClassification parmCustomerClassification(CustomerClassification
_customerClassification = customerClassification)
{
```

```
        customerClassification = _customerClassification;
        return customerClassification;
    }
}
```

Step 6: Wrap the buildContract method

Provide a build contract implementation that calls **next** to load the standard business event contract and populates any payload extensions. Here is the complete class.

```
[ExtensionOf(classStr(CustFreeTextInvoicePostedBusinessEvent))]
public final class FreeTextInvoicePostedBusinessEventContract_Extension
{
    public BusinessEventsContract buildContract()
    {
        var businessEventContract =

CustFreeTextInvoicePostedBusinessEventExtendedContract::newFromCustFreeTextInvoicePostedBusinessEventContrac
t(next
        buildContract());

businessEventContract.parmCustomerClassification(CustomerClassifier::deriveCustomerClassification(businessEv
entContract.parmInvoiceAccount()));
        return businessEventContract;
    }
}
```

Extending filters so that they have custom fields (if the middleware supports this extension)

Some middleware systems allow for filtering of the events. For example, Microsoft Azure Service Bus has a property bag that can be populated with key-value pairs. These key-value pairs can be used to filter events when reading from the Service Bus Queue or Topic. Additionally, Azure Event Grid has filterable message properties such as **Subject**, **Event Type**, and **ID**. To support these various properties for the different systems, the business events framework uses a concept that is named *payload context*. This concept can be extended so that it includes custom fields that the different eventing systems can use for filtering.

Payload context

The business events framework supports the payload context concept. Payload context provides a way to decorate messages that the framework sends with context about the payload. In some scenarios, additional context might be required when messages are sent to endpoints. Therefore, the framework has hookpoints where the context can be overwritten and the adapters can be customized.

Step 1: Add a custom payload context

A custom payload context must be extended from the **BusinessEventsCommitLogPayloadContext** class.

```
class CustomCommitLogPayloadContext extends BusinessEventsCommitLogPayloadContext
{
    private utcdatetime eventTime;
    public utcdatetime parmEventTime(utcdatetime \_eventTime = eventTime)
    {
        eventTime = \_eventTime;
        return eventTime;
    }
}
```

Step 2: Construct the custom payload context

A Chain of Command (CoC) extension must be written for the **BusinessEventsSender.buildPayloadContext**

method, so that it can construct the new payload context type.

```
[ExtensionOf(classStr(BusinessEventsSender))]
public final class CustomPayloadContextBusinessEventsSender_Extension
{
    protected BusinessEventsCommitLogPayloadContext
    buildPayloadContext(BusinessEventsCommitLogEntry \_commitLogEntry)
    {
        BusinessEventsCommitLogPayloadContext payloadContext = next
        buildPayloadContext(_commitLogEntry);
        CustomCommitLogPayloadContext customPayloadContext = new
        CustomCommitLogPayloadContext();
        customPayloadContext.initFromBusinessEventsCommitLogEntry(_commitLogEntry);
        customPayloadContext.parmEventTime(_commitLogEntry.parmEventTime());
        return customPayloadContext;
    }
}
```

Step 3: Consume the custom payload context from an adapter

Adapters that consume payload context are written in such a way that they expose CoC methods that allow for the consumption of new payload contexts. The following example shows what this step looks like for the Service Bus adapter. This adapter has a generic property bag that Service Bus consumers can filter on.

The **BusinessEventsServiceBusAdapter** class has the CoC method that is named **addProperties**.

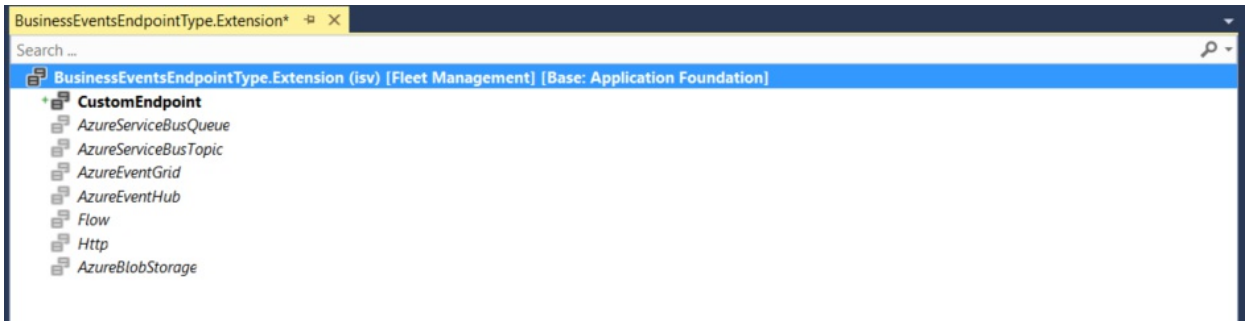
```
[ExtensionOf(classStr(BusinessEventsServiceBusAdapter))]
public final class CustomBusinessEventsServiceBusAdapter_Extension
{
    protected void addProperties(BrokeredMessage \_message,
    BusinessEventsEndpointPayloadContext \_context)
    {
        if (_context is CustomCommitLogPayloadContext)
        {
            CustomCommitLogPayloadContext customPayloadContext = \_context as
            CustomCommitLogPayloadContext;
            var propertyBag = \_message.Properties;
            propertyBag.Add('EventId', customPayloadContext.parmEventId());
            propertyBag.Add('BusinessEventId', customPayloadContext.parmBusinessEventId());
            // Convert the enum to string to be able to serialize the property.
            propertyBag.Add('BusinessEventCategory', enum2Symbol(enumNum(ModuleAxapta),
            customPayloadContext.parmBusinessEventCategory()));
            propertyBag.Add('LegalEntity', customPayloadContext.parmLegalEntity());
            propertyBag.Add('EventTime', customPayloadContext.parmEventTime());
        }
    }
}
```

Adding a custom endpoint type

The business events framework supports the addition of new endpoint types to the out-of-box endpoint types. This section provides an example that shows how to add new custom endpoint types.

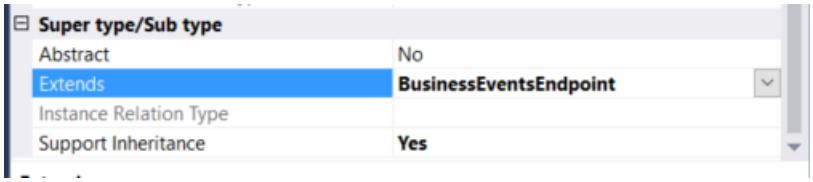
Step 1: Add a new endpoint type

Each endpoint type is represented by the **BusinessEventsEndpointType** enum. The first step in the process of adding a new endpoint is to extend this enum, as shown in the following illustration.

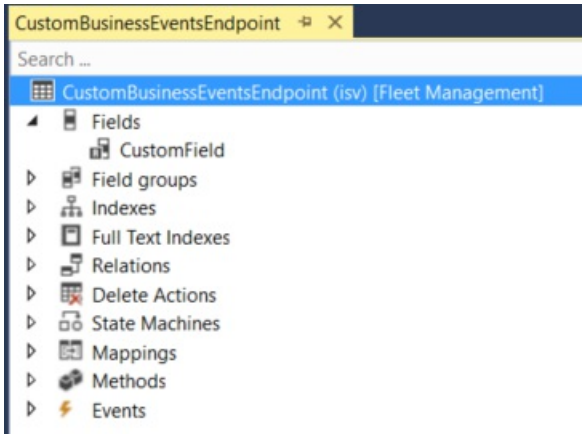


Step 2: Add a new endpoint table to the hierarchy

All endpoint data is stored in a hierarchy table. The root of this table is the BusinessEventsEndpoint table. A new endpoint table must extend this root table by setting the **Support Inheritance** property to **Yes** and the **Extends** property to **"BusinessEventsEndpoint"** (or any other endpoint in the BusinessEventsEndpoint hierarchy).



The new table then holds the definition of the custom fields that are required to initialize and communicate with the endpoint in code. To help avoid conflict, you should qualify field names to the specific endpoint where they belong. For example, two endpoints can have the concept of a **URL** field. To distinguish the fields, names should be specific to the custom endpoint. For example, name the field for the custom endpoint **CustomURL**.



Step 3: Add a new endpoint adapter class that implements the IBusinessEventsEndpoint interface

The new endpoint adapter class must implement the **IBusinessEventsEndpoint** interface. It must also be decorated with the **BusinessEventsEndpointAttribute** attribute.

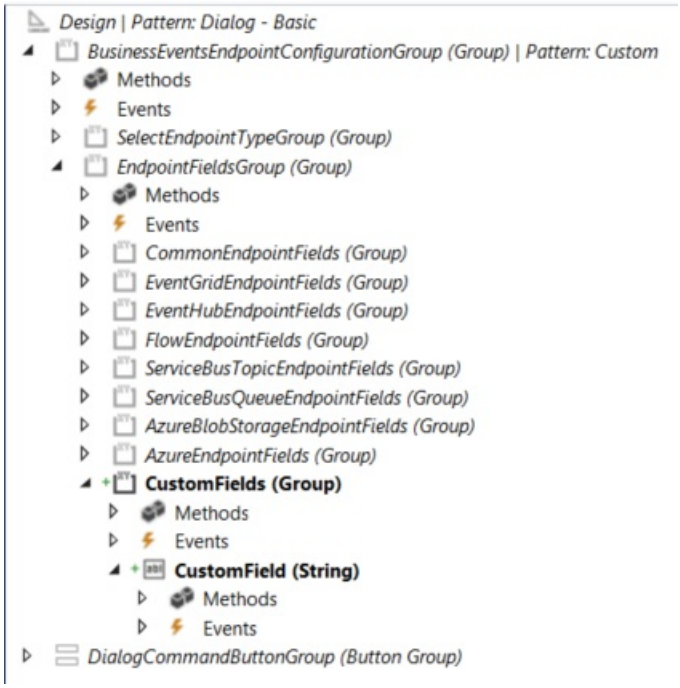
```
[BusinessEventsEndpoint(BusinessEventsEndpointType::CustomEndpoint)]
public class CustomEndpointAdapter implements IBusinessEventsEndpoint
{
```

The **initialize** method should be implemented to check the type of the **BusinessEventsEndpoint** buffer that is passed in, and then, if the buffer is of the correct type, initialize it, as shown in the following example.

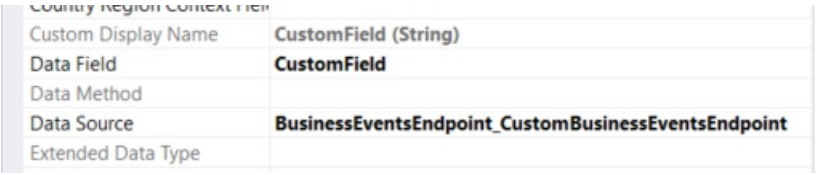
```
if (!(_endpoint is CustomBusinessEventsEndpoint))
{
    BusinessEventsEndpointManager::logUnknownEndpointRecord(tableStr(CustomBusinessEventsEndpoint),
    \_endpoint.RecId);
}
CustomBusinessEventsEndpoint customBusinessEventsEndpoint = \_endpoint as
CustomBusinessEventsEndpoint;
customField = customBusinessEventsEndpoint.CustomField;
if (!customField)
{
    throw warning(strFmt("\@BusinessEvents:MissingAdapterConstructorParameter",
    classStr(CustomEndpointAdapter), varStr(customField)));
}
```

Step 4: Extend the EndpointConfiguration form

Add a new group control under FormDesign/BusinessEventsEndpointConfigurationGroup/EndpointFieldsGroup/ to hold your custom field input.



The custom field input should be bound to the new table and field that you created in the previous step. Create a class extension to extend the **getConcreteType** and **showOtherFields** methods of **BusinessEventsEndpointConfiguration** form, as shown in the following example.



```
[ExtensionOf(formStr(BusinessEventsEndpointConfiguration))]
final public class CustomBusinessEventsEndpointConfiguration_Extension
{
    public TableName getConcreteTableType(BusinessEventsEndpointType \_endpointType)
    {
        TableName tableName = next getConcreteTableType(_endpointType);
        if (_endpointType == BusinessEventsEndpointType::CustomEndpoint)
        {
            tableName = tableStr(CustomBusinessEventsEndpoint);
        }
        return tableName;
    }
    public void showOtherFields()
    {
        next showOtherFields();
        BusinessEventsEndpointType selection =
        any2Enum(EndpointTypeSelection.selection());
        if (selection == BusinessEventsEndpointType::CustomEndpoint)
        {
            this.control(this.controlId(formControlStr(BusinessEventsEndpointConfiguration,
            CustomFields))).visible(true);
        }
    }
}
```

Adding human-readable data fields to the payload

This feature is available in Platform update 30 and later.

The serialization of business events uses FormJsonSerializer to serialize objects in the data contract. FormJsonSerializer can format **UtcDataTime** values in the ISO 8601 date and time format. This format is human-readable when the payload of a business event is viewed. For example, a **UtcDataTime** value can now be formatted as **"2007-12-05T14:30Z"** instead of **"/Date(1196865000000)/"**. In the "/Date(N)" format, N is the number of milliseconds that have passed since January 1, 1970, UTC+0. The ISO format is more often understood by tools that parse JavaScript Object Notation (JSON).

To get the human-readable format, use the extended data type (EDT) that is named **DateTimeIso8601** as the type of the value in the data contract. Alternatively, use an EDT that is derived from the **DateTimeIso8601** EDT.

```
[DataMember("TestIsoEdtUtcDateTime")]
public DateTimeIso8601 testIsoEdtUtcDateTime(DateTimeIso8601 _value = this._testIsoDateTime)
{
    if (!prmIsDefault(_value))
    {
        this._testIsoDateTime = _value;
    }
    return this._testIsoDateTime;
}
```

Troubleshoot business events

2/18/2021 • 4 minutes to read • Edit Online

This topic provides tips for troubleshooting issues that involve business events.

ISSUE	POSSIBLE RESOLUTION
Unable to find navigation to business events in the **System parameters** form	You can access business events by going to **System administration > Set up > Business events**. This change was made when business events was made generally available in Platform update 26.
Error: Unable to construct endpoint. Exception message: Error retrieving secret '[KeyValueSecretName]' from key vault 'https://[KeyVaultName].vault.azure.net/': AADSTS700016: Application with identifier '7e28cb03-dc28-43b5-b129-e13dcfb4b1fb' was not found in the directory 'ee3fe5c6-26af-42b1-9acf-5ee38e6ead6e'.	This can happen if the application has not been installed by the administrator of the tenant or consented to by any user in the tenant. It's likely that you may have sent your authentication request to the wrong tenant.
Error: Trace ID: 19dc9946-45b6-4335-9676-6a133dbf4000 Correlation ID: ecbc8a80-f9d0-41ec-9c8f-d334d050bd64 Timestamp: 2019-02-06 23:27:06Z	This error typically means that the value in the **Azure Active Directory Application ID** field is incorrect. Check the **Azure Active Directory Application ID** value in the customers Azure portal in **Azure Active Directory > App Registration**.
Error: Unable to construct endpoint. Exception message: Error retrieving secret '[KeyValueSecretName]' from key vault 'https://[KeyVaultName].vault.azure.net/': An error occurred while sending the request.	This error is likely due to an incorrect value in the **Key Vault DNS Name** field. To resolve this, go to the customer's Azure portal and open the key vault object. In the **Overview** section, check the **Key Vault DNS Name** value.
Error: Unable to send test event to endpoint. Exception message: 40103: Invalid authorization token signature, Resource:sb://[ServiceBusName].servicebus.windows.net/[QueueName]. TrackingId:cd0eccaa-1717-4f97-b837-4cd7eda99af4_G13, SystemTracker: [ServiceBusName].servicebus.windows.net:[QueueName], Timestamp:2019-02-06T23:36:54	The value in the customer's Key Vault Secret is likely incorrect. Check the **Key Vault Secret** value and make sure that it is correct for the endpoint type.
Error: Unable to construct endpoint. Exception message: Error retrieving secret '[KeyValueSecretName]' from key vault 'https://[KeyVaultName].vault.azure.net/': Access denied	This is likely due to the **Azure Active Directory Application ID** not having the appropriate permissions in the Key Vault. To resolve this, go to the Azure portal and open the **Key Vault** object. Go to **Access Policies** and add the AAD application with Key, Secret, and Certificate Management template.
Error: Unable to send test event to endpoint. Exception message: 40400: Endpoint not found., Resource:sb://[ServiceBusName].servicebus.windows.net/[QueueName].	This issue is likely a result of the queue/topic/hub name being incorrect. Check the **Name** field by going to service bus in the Azure portal and reviewing the **Topic** or **Queue** name. If it is an Event Hub, go to the **Event Hub** object in Azure and validate the **Hub** name.
Error: Unable to send test event to endpoint. Exception message: An error occurred while sending the request.	This is likely due to an incorrect endpoint value specified in the **Endpoint URL** field. Go to the **Event Grid** object in the Azure portal and open the **Event Grid**. In the **Overview** section, this value will be the **Topic Endpoint**.

ISSUE	POSSIBLE RESOLUTION
Business events do not show up in the catalog	The Business event catalog is built during full database sync. As a result, there are some use cases, as noted below, where a manual refresh of the catalog is needed in order to see the new business events. Manual refresh can be invoked from the catalog by going to **Manage > Rebuild business events catalog**. When you are implementing business events in Visual Studio you may not see the newly coded business event in the catalog. When new workflows are configured, such as the workflow elements or steps, might not show up in the business events catalog. In other situations, when you don't see certain business events, doing a manual refresh should resolve the issue.
- 0 is an invalid bundle size - Business events are not getting triggered - Microsoft Flow is not getting triggered by business events - Unable to configure business event because it has reached the limit of 0 configured endpoints	One of the reasons why this issue can occur is if certain parameters are not set as expected in the **BusinessEventsParameters** table. This is due to an update in which some of the business events parameters not being set correctly. In a non-production environment, you must update the parameters in **System administration > Setup** to set retry count = 3; End points allowed per event = 10 and wait time = 1000. After this update, restart the batch service and run IISReset to pick up the latest values. If business events still do not trigger, then the dedicated capacity is not working to process business events. A manual batch job must be scheduled to process business events, which can be enabled from the **Business events parameters** page in **System administration > Set up**.
Platform update 30 compiler warning when creating custom payload context fields by augmenting via Chain of Command (CoC) the addProperties method in the adapter class. **Class 'BusinessEventsServiceBusAdapter' is internal in model 'ApplicationFoundation' and cannot be extended**.	This is a change in the compiler that prevents an internal API from being extended. This is being tracked as a bug to provide alternate ways to add custom properties. For more information, see this Yammer discussion.
Error: Unable to load one or more of the requested types. Retrieve the LoaderExceptions property for more information	This error message on the error tab of active business events can typically be resolved by rebuilding the catalog.
Alert business events don't trigger	One of the reasons why an event is not triggering could be a potential issue with alerts email functionality. Try turning off the send email option in the alert to see if that resolves the issue.
Unable to send test event to endpoint. Exception message: The underlying connection was closed: Could not establish trust relationship for the SSL/TLS secure channel.	Make sure the middleware is using TLS 1.2

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Use cases for business events

2/18/2021 • 9 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

The following are potential uses cases for business events. These use cases aren't an exhaustive list of the potential use cases. Some of these use cases may not have been implemented yet either by Microsoft or other organizations. These use cases are meant to provide ideas and help with understanding business events.

BUSINESS PROCESS	USE CASE	VALUE
Procurement	The procurement process frequently relies on manual intervention, so automating this process should increase procurement manager productivity.	You can make the procurement process more efficient by alerting procurement managers when quotation requests are sent, allowing for prompt follow-up and faster engagement. The goal is to have procurement managers follow up within three days and possibly create a Microsoft Power Automate that automates this follow up.
Procurement	Many organizations use manual processes to communicate internally and externally about production orders. Some organizations have complex approval processes for production orders, and need quick and easy ways to review and approve orders.	By creating business events that are triggered when production orders are updated, you can help improve communications between the back office and the production floor. When integration is required with third-party systems for production, the business events can also be used to help simplify the integration process.
Reporting	Managers are not informed about newly created financial reports. As a result, managers might analyze and make decisions based on outdated data.	You can make the reporting process more efficient by alerting managers when financial reports are sent, allowing for prompt follow-up and faster engagement. The goal is to have managers follow up within three days and possibly create a Power Automate that automates this follow-up.

BUSINESS PROCESS	USE CASE	VALUE
Customer master	When a new customer is created, a credit limit check is needed. If something could automatically trigger an API that subscribes to a credit limit, and then check the website and import-specific credit limit check fields, such as limit amount and rating. At the same time, an approval Power Automate needs to start so that the customer account can be used after approval from management.	This check is required by many companies, especially in the retail area. This use case would be beneficial because partners develop customizations for this purpose.
Month end close	The month-end closing schedule is a simple list that does not allow automatic actions. If functionality could be created to inform a group of people about tasks that have been completed and verified as complete, then a different group of people could start working.	Use real-life scenarios to enhance the currently static and difficult to use month-end closing workspace. One such use case is explained in detail in Business events in Financial Period Close.
Month end close	If functionality could be created to trigger a Power Automate when the status of a period is changed – either opened or closed.	Users are not automatically informed when new periods are opened after the month-end close is completed and they are allowed to start recording transactions in the new period.
Vendor master	If you are using the supplier (vendor) collaboration workspace, there is no automated way to inform the supplier that a new record has been created or existing records have been updated. This is an issue with requests for quotation and purchase orders. This means that the supplier would need to review the supplier collaboration workspace in case there are issues.	Using business events, the supplier portal functionality will provide improved productivity and efficiency by removing the need for additional correspondence between the supplier and the application. This is a true collaboration and will lead to supply chain efficiency. Without this functionality, the organization must follow up on all new requests or updates with phone calls or emails to the suppliers. This improves the supplier collaboration workspace.
Vendor master	Many organizations use manual, offline, and paper-based processes to manage vendors. Additionally, in many organizations, there are "gate keepers" that manage the master data. When updates occur to vendor master data, communication throughout the organization is limited or complex.	By integrating with Power Automate and creating business events for actions and events that regularly occur with vendor master data, you can help improve the overall vendor onboarding process. You can also help improve internal communications within your organization and automate business processes with your trade partners.

BUSINESS PROCESS	USE CASE	VALUE
Sales quotation	A quotation is placed for a personalized product. After the quotation is won this needs to become a real product. Requests are sent to a Data team (using PowerApps or Office) to create the item. After the item has been manually created, the item on the line is updated and the quote triggers the creation of a sales order. Currently, when the item is created, the approach is to wait for the data integrator to copy it to Dataverse. Monitors are set to find new entries on Dataverse and compare the unique references to any open numbers. Ideally, an item created business event exists, which is extended so that it has the unique identifier. This event is then subscribed to (in Power Automate or PowerApps) and immediately updates the quote line.	Using business events, the business would be able to update the won quote with the new product when the new product is created. This removes possible delays, manual updates, and errors.
Sales orders	Shipment for sales orders starts in a different system to enable logistics balancing of third-party haulers, production of customs, and delivery documentation. This pushes the data to a major transport company. Upon picking or shipment of a sales order, the line details are the event that triggers the external system to analyze and determine the details of the utilized shipment. This pushing event depends on the ability for a business action to update the application. This could be that the order is picked and then the shipment information pushed to the application, or it is marked as shipped as it is pushed externally.	This allows businesses with third-party transport companies to send shipping information to their vendors. This approach puts the responsibility of dispatch utilization on the external vendors and simplifies the setup.
Sales orders	Picking for sales orders can be performed in different systems, whether it's a separate internal warehousing system or a third-party location that is not part of the application. Upon confirmation or picking of a sales order, the line details are the event that triggers the external system to analyze and determine the picking priority and utilization.	This allows businesses with third-party warehouses to send the picking information to the system. This can also be used if the warehouse has automated picking machines that determine what is required. This approach puts the responsibility of prioritizing picking and utilization on the external system and simplifies the setup in the application.

BUSINESS PROCESS	USE CASE	VALUE
Sales orders	Organizations are looking for ways to improve customer service, and to streamline and automate sales processes. These processes might involve several people, require many manual steps, or involve several third-party solutions. The email notification profile feature in Finance and Operations has limited email formatting capabilities.	By integrating with Power Automate and creating business events for actions that are taken on sales orders, you can help improve the overall sales process, help increase customer satisfaction, and automate collaboration between sales and operations.
Batch processing	Scheduling batch jobs to process polling logic can constrain resources. To realize near real-time processing, it is compelling to schedule the batch to run as fast as it can in Finance and Operations, which typically ends up being every few minutes. These batch jobs are typically data import/export jobs that look for data to be processed. However, if data is not available the batch job processes empty cycles, which can consume system resources.	Using business events, the polling use case can be redesigned to be asynchronous if it is triggered by the business event. Data will be processed only when it is available. The business logic that makes the data available triggers the business event, which can then be used to start the data processing job/logic. This can save thousands of batch executions from running empty cycles and wasting system resources.
Production orders	The production scheduling process often relies on external systems or optimization engines to finalize the schedule. Automation of the process and an ability to integrate with external systems when a production order is scheduled can help increase productivity and decrease integration costs.	You can improve the integration and automation capabilities by creating a business event when a production order is scheduled. The information can be communicated to downstream systems, such as Microsoft Dynamics 365 Field Service for Internet of Things (IoT) data exchange, third-party manufacturing execution systems (MESs), or a scheduling optimization engine.
Production orders	Many organizations use third-party MESs to control and manage their machinery on the production floor. Integration with these systems is often complex. Additionally, communication between the production floor and the back office can be difficult.	By creating business events that are triggered when production orders are updated, you can help improve communications between the back office and the production floor. When integration is required with third-party systems for production, the business events can also be used to help simplify the integration process.

BUSINESS PROCESS	USE CASE	VALUE
Case management	Case management is a platform feature that lets you track and manage various requests and issues that are internally or externally reported or requested. Typically, each type of case has a different process, and various contributors to the case might be involved throughout the case's lifecycle. The processes can be manual, and communication can often be delayed. Sometimes, contributors to the cases or stakeholders in them might not be users of the system.	By using Power Automate when cases are updated, you can streamline and automate the business processes that are related to various case types. You can also automate communication and implement approval processes when necessary, depending on your organizational requirements.
Transportation management	Many organizations have separate and disconnected systems for managing transportation. Additionally, the staff that is responsible for arranging transportation is typically not the same staff that creates and confirms orders, or the same staff that carries out the work to ship or receive orders. This situation can lead to a breakdown or lack of communication, and can require manual processes to notify various departments when an order is ready for the next step in the process. In many cases, third-party shipping software is used, and data is entered into many systems or tracked in paper logs.	By using business events together with Power Automate, you can streamline the overall transportation management process and help improve communications between departments.

Business events and Microsoft Power Automate

2/18/2021 • 3 minutes to read • Edit Online

This topic provides steps detailing how to configure and consume a business event from a Power Automate endpoint.

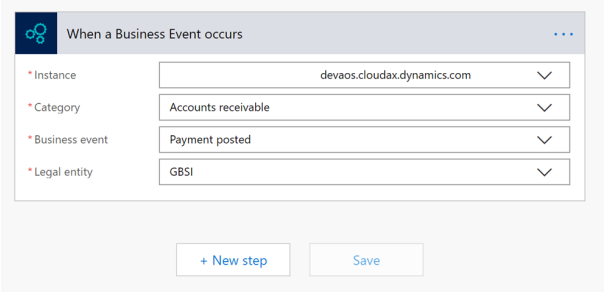
This topic shows how to perform the following tasks:

- Create a new flow in Power Automate.
- Trigger a business event.

Create a new flow in Power Automate

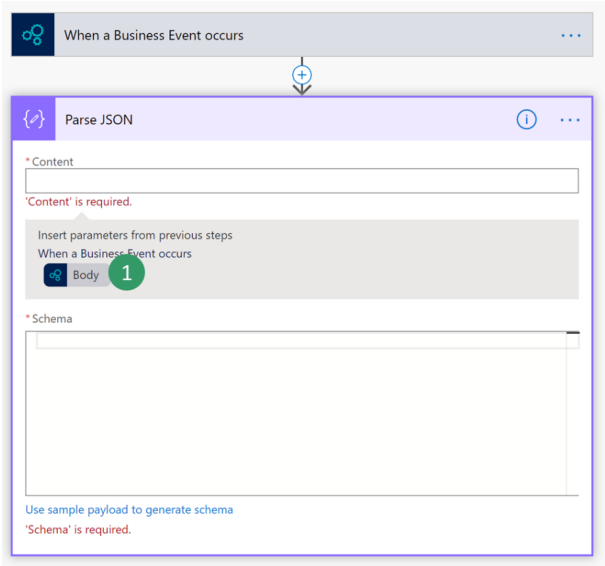
1. Sign in to Power Automate portal.
2. Select an existing environment where you have the permissions needed to create a Power Automate resource. The default environment is open to all companies.
3. Select **New > Create from blank**.
4. Search for **Dynamics 365 for Finance and Operations** and select the connector.
5. You will notice a trigger named **When a Business Event occurs**. Select this trigger.
6. Select your environment instance, category, event name, and legal entity.

> **TIP**
> Take advantage of the auto-complete that Power Automate provides by entering only part of the environment instance URL or part of the event name.

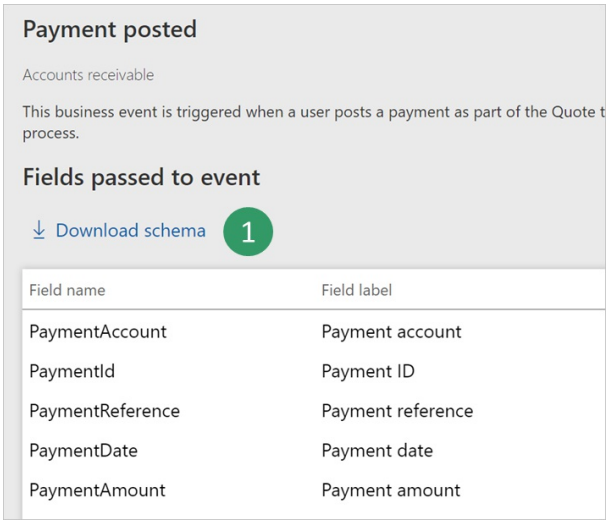


7. Select the **New Step** button to add a new action.
8. Search for the **Parse JSON** data operation. This step is needed to parse the message with the schema of the data contract.

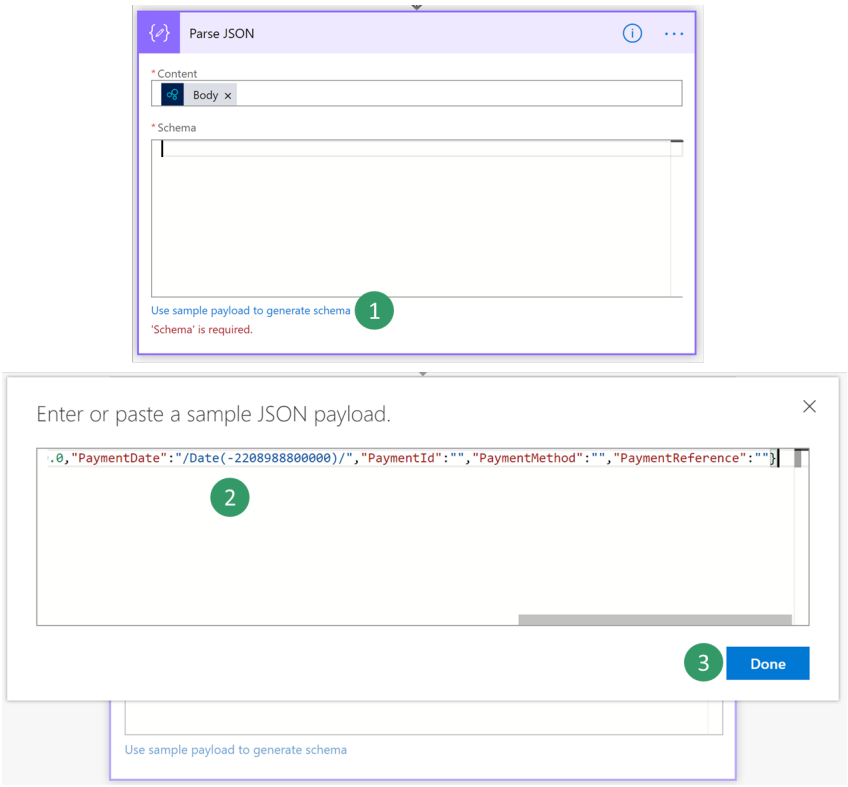
9. Select the content field of **Parse Json** action, then the **Body** output from the previous step should appear as an option. Select **Body**.



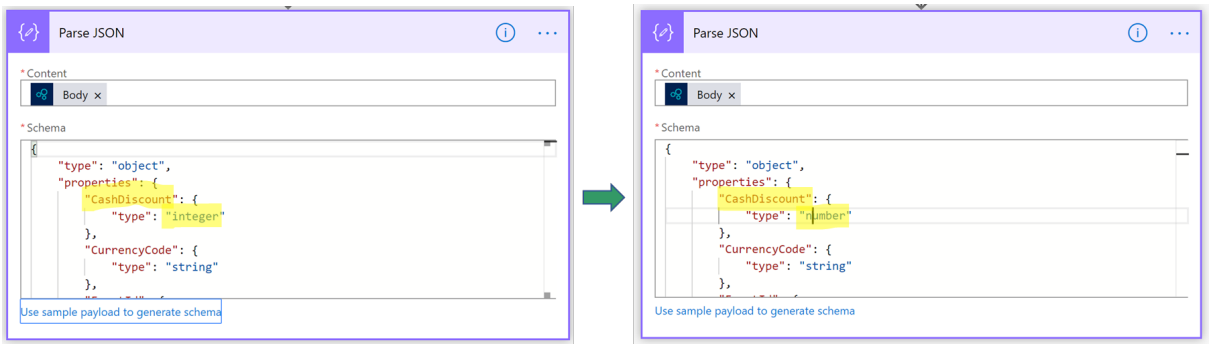
10. Enter the schema of the contract. Because the app provides only a sample payload you can use the Power Automate capability to generate a schema from a payload. Select an event in the catalog (for example, Customer Payment) and select the **Download schema** link. This will download a text file. Open the text file and copy the content.



11. Go Back to Power Automate and select the **Use sample payload to generate schema** link. Paste your text file content and select **Done**.

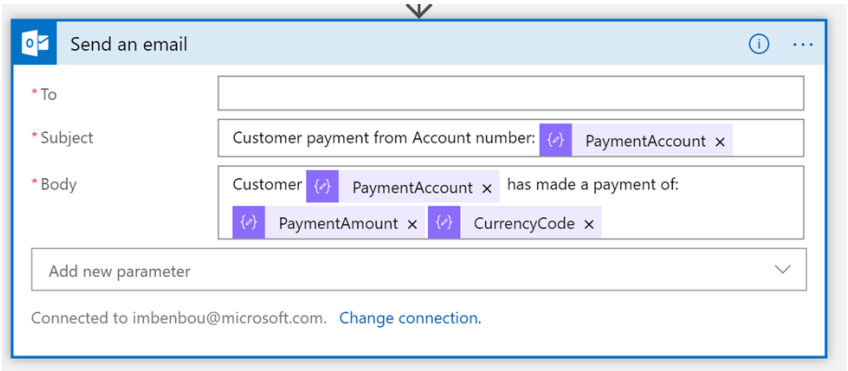


12. Depending on the quality of your sample payload, your generator will not be able to distinguish between an integer and a real number. This is true if the real number is provided as a whole number in the sample payload. Review your generated schema and check if you need to change an "integer" into "number". (In JSON, a "number" data type means real number).



13. Choose another final action to consume the business event content. For instance, you can send an email (or post a text message to Teams) to notify the customer about payment details. Search for the **Send email** action, then sign in to your Microsoft 365 account.

14. Fill in the message with the required fields.

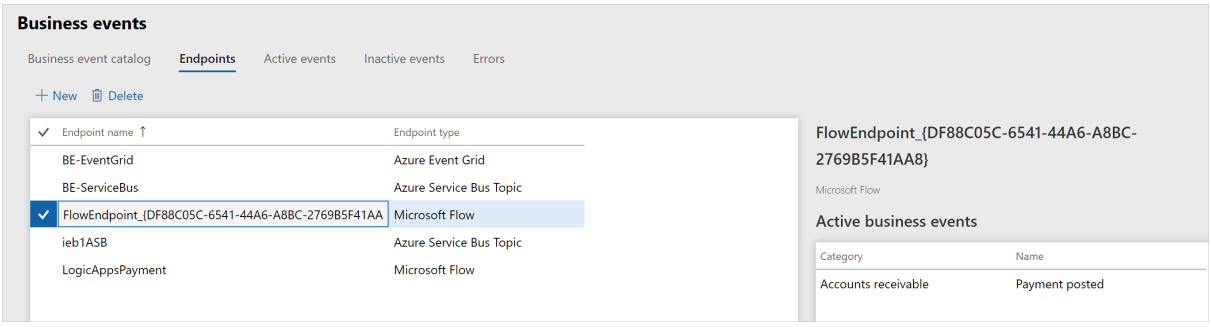


15. Save the flow.

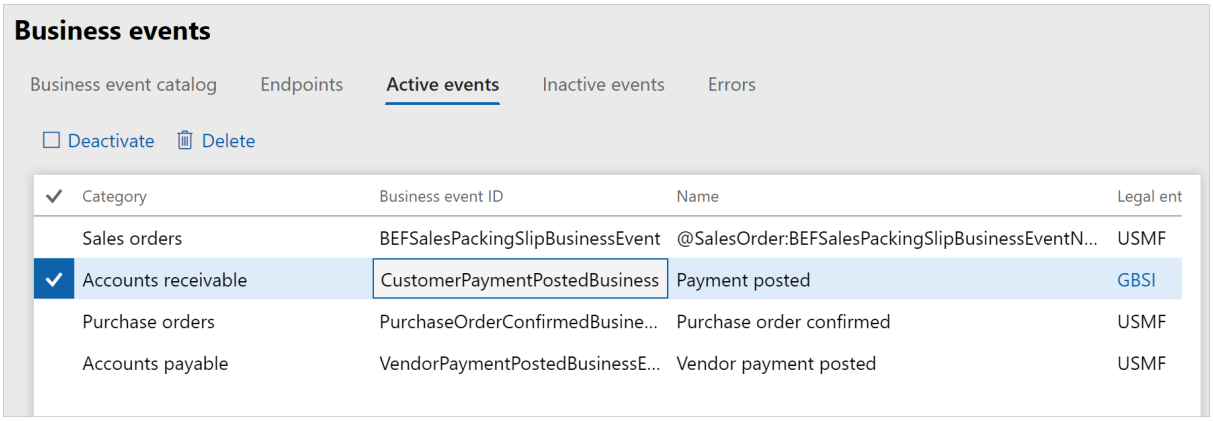
Trigger a Business Event

Power Automate can configure the application automatically for you. After you save your flow, it creates an endpoint, then it activates the business event for you. There is no remaining configuration step apart from verifying that the endpoint has been correctly configured before triggering an event.

1. Sign in to the client.
2. Go to **System Administration > Setup > Business Events**.
3. Select **Endpoints**.
4. Verify that a new endpoint has been created with a GUID appended in the name.



5. If you check the **Active events** tab you can also verify that "**Payment Posted**" is activated for legal entity GBSI.



6. The final step is to trigger the business event of a posted customer payment and check whether the flow runs and you receive an email with customer payment details.

Troubleshooting a flow

Here are some troubleshooting suggestions:

- Power Automate provides a full history of runs to help determine what might be wrong with a failing flow.
- When reviewing a failed run, carefully review the inputs and outputs of trigger and action blocks.
- After changes have been made to the flow, go to the latest run or a particular run, and **Resubmit** the inputs to run the flow again.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Business events and Azure Event Grid

2/18/2021 • 6 minutes to read • Edit Online

This topic explains how to configure a Microsoft Azure Event Grid endpoint, and how to consume a business event from Event Grid.

Scenario overview

Security best practices recommend that you store connection strings outside applications, in an Azure Key Vault drive, and that you give applications the correct access to the key vault keys, secrets, or certificates.

Here are two of the many benefits of this approach:

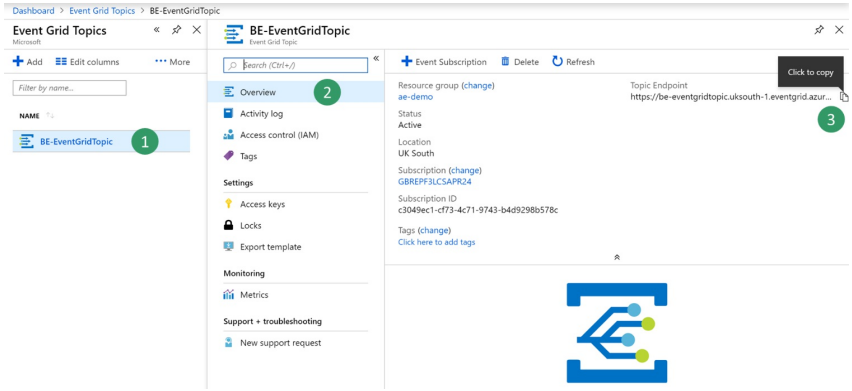
- Someone who gets access to the application database won't be able to get the third-party connection string.
- Maintenance is easier, especially when multiple applications access the same resources, because you must update connection strings in only one place.

Here is an overview of the procedures that you must complete:

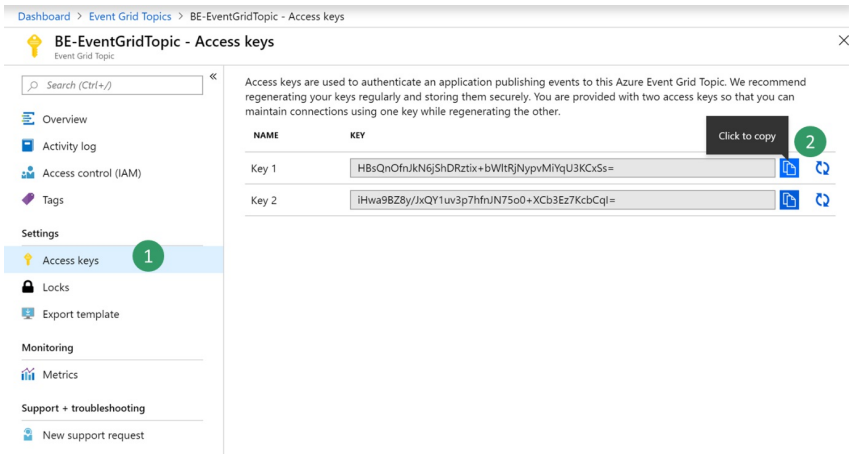
1. Create a new event grid topic.
2. Create a new key vault to store the key for the event grid topic.
3. Register an Azure app that has permission to access the key vault.
4. Configure the parameters of the endpoint.
5. Consume the business event.

Procedure 1: Create a new event grid topic

1. Sign in to the Azure portal.
2. Select **All services** > **Integration** > **Event Grid Topics**.
3. Select **Add** to create a new event grid topic. Set the parameters, and then select **Create**. You can create a new resource group as a container for your lab, or you can use an existing resource group.
4. After deployment is completed, select the new event grid. On the property blade, select **Overview**, and make a note of the **Topic Endpoint** value. You will need this value later.



5. Back on the property blade, select **Access keys**, and copy the **Key 1** value. You will need this value when you configure the key vault in the next procedure.

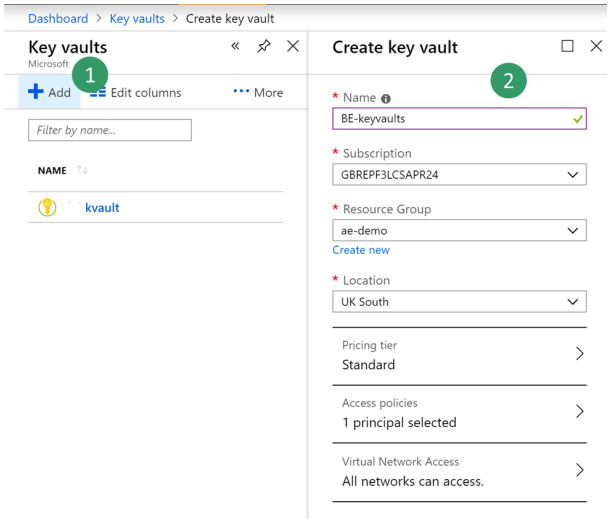


Procedure 2: Create a key vault

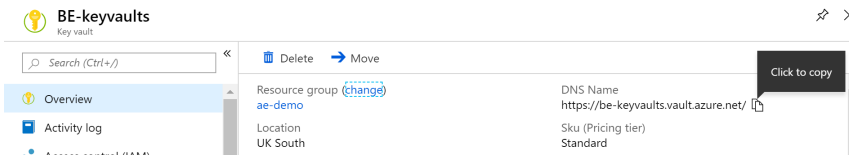
In this procedure, you will create a key vault to store the key that you copied in the previous procedure. A key vault is a secure drive that is used to store keys, secrets, and certificates. Instead of storing the connection string, a more typical and more secure approach is to store it in a key vault. You can then register a new application with Azure Active Directory (Azure AD) and grant it the right to retrieve the secret from the key vault.

1. In the Azure portal, select **All services > Security > Key vaults**.

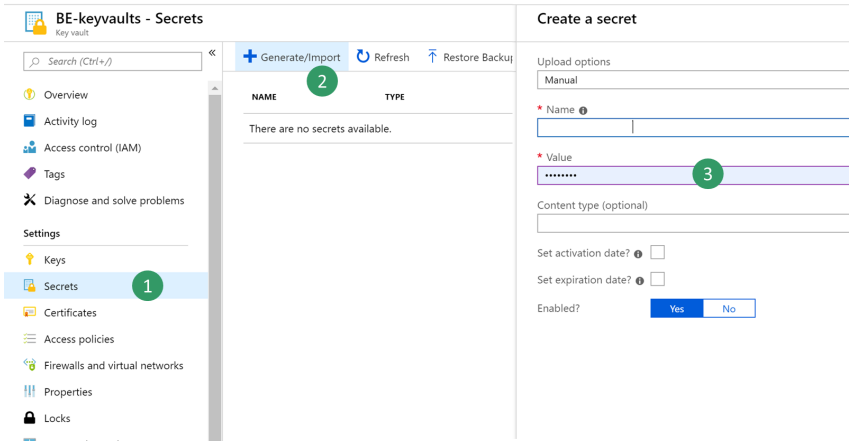
2. Create a new key vault in your resource group and set the default parameters.



3. Select **Overview**, then copy and save the **DNS Name** value for the key vault. You will use this value later.



4. Select **BE-key vault > Secrets > Generate/Import**. Enter a name for your secret, and paste the event grid connection string that you saved earlier.

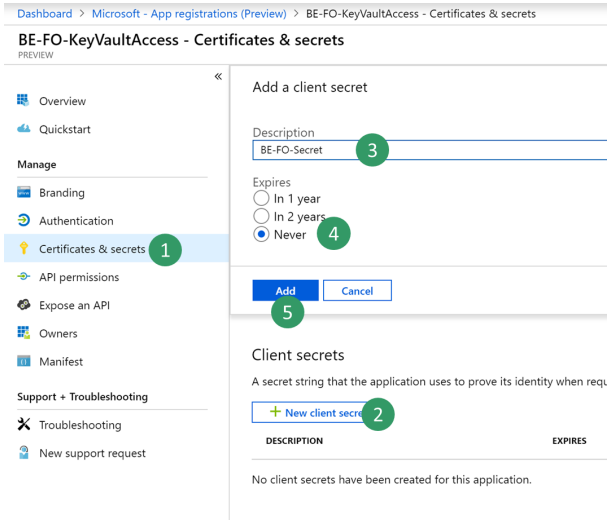


5. Select **Create**.

Procedure 3: Register a new application

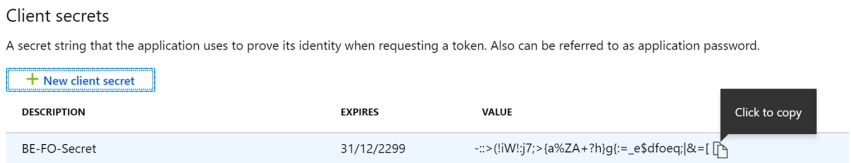
In this procedure, you will register a new application with Azure AD, and give it read and retrieve access to key vault secrets. The application will then use this application to retrieve event grid secrets.

1. In the Azure portal, select **All services > Security > Azure Active Directory**.
2. Select **App registrations (preview) > New registration**, and then enter a name for your application.
3. Select **Register**.
4. Select your new application, and then select **Certificates & secrets > New client secret**. Enter a name for your secret, and set the secret so that it never expires. Then select **Add**.

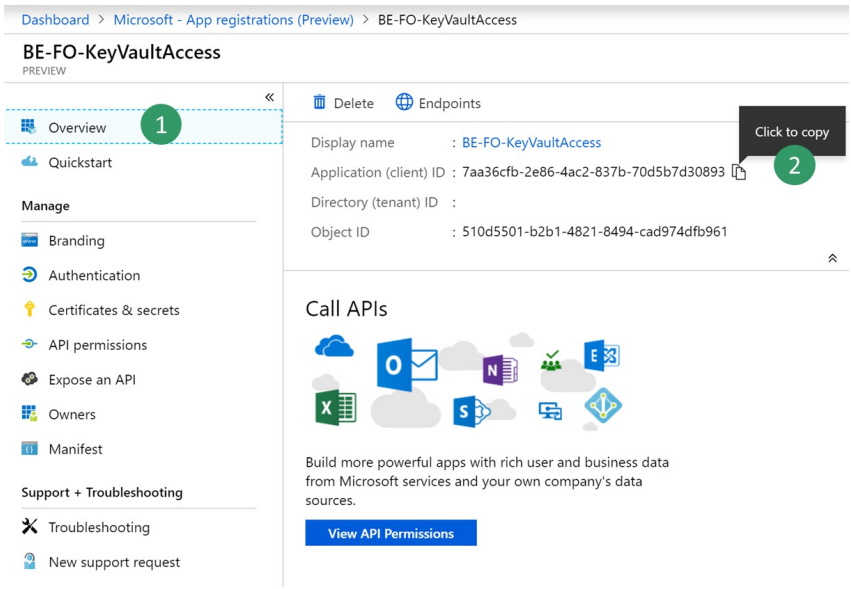


5. Copy and save your new secret. You will use it later.

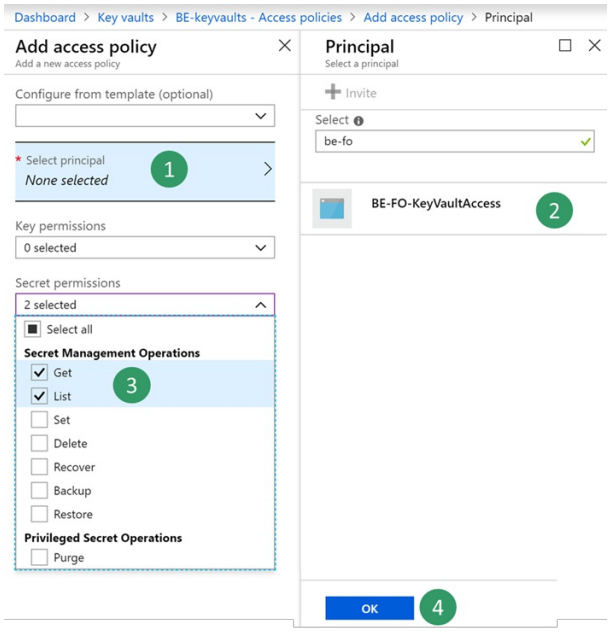
> **IMPORTANT**
>
> Secrets are visible only one time. If you forget to copy the secret, you will have to delete it and create a new secret.



6. Select **Overview**, and copy and save the application ID. You will use this value later.



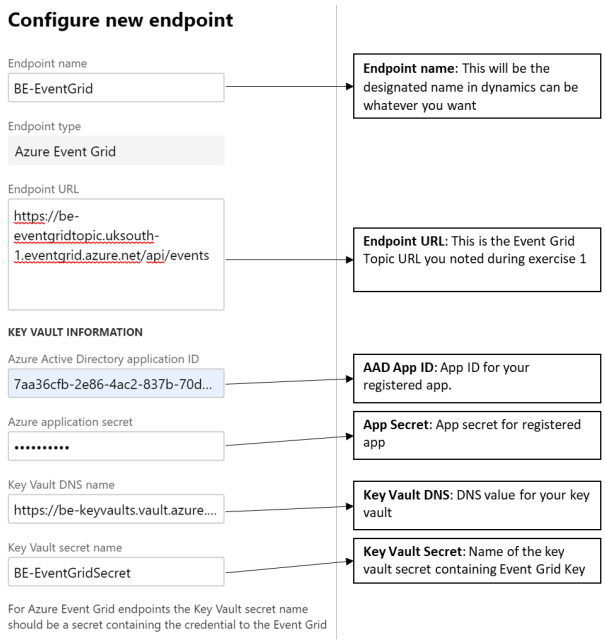
7. Select **All services** > **Security** > **Key vaults**.
8. Select the key vault that you created earlier, and then select **Access policies** > **Add new**.
9. On the **Principal** blade, select your new registered application. Select the check boxes for the **Get** and **List** secret permissions to retrieve key vault secrets.



10. Save your new access policy.

Procedure 4: Configure a Business Events endpoint

1. Sign in to the application and go to **System administration** > **Setup** > **Business events**.
2. Select **Endpoints**.
3. Select **New**.
4. Select **Azure Event Grid**.
5. Select **Next**.
6. Set the required parameter values.

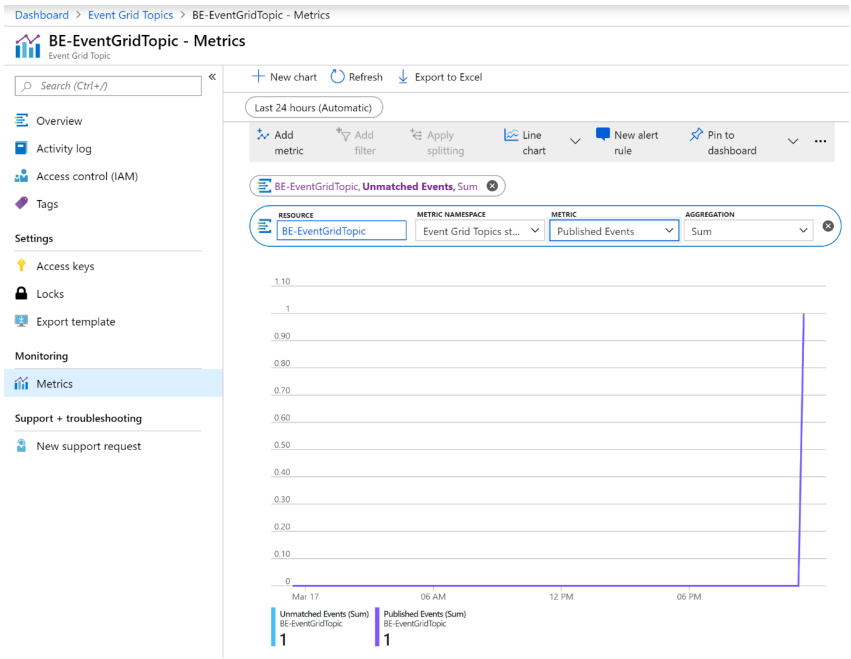


7. Select **OK**.

Procedure 5: Consume a business event

The business scenario involves sending an email message whenever a free text invoice is posted for the USMF company. The message must contain details such as the customer account number, the customer name, and the total amount of the invoice.

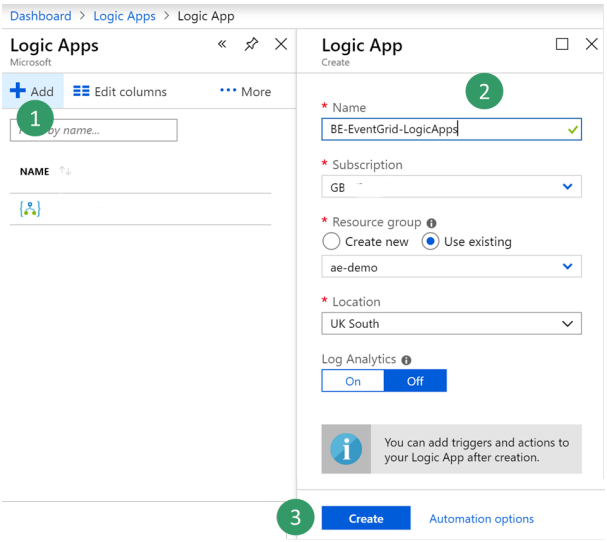
1. Select the business event catalog and look for **free text invoice posted** business event.
2. Then activate the business event for USMF company. Once activated, a test message is sent to validate the configuration and cache the connection.
3. To verify that the test message has been received, in the Azure portal, select your event grid topic, and then select **Metrics**. Verify that both the **Published Events** metric and the **Unmatched Events** metric show a value of at least **1**. If they don't, wait for the batch job to pick up your message.



When both metrics have a value of at least **1**, you will create a new logic app to subscribe to your event grid topic.

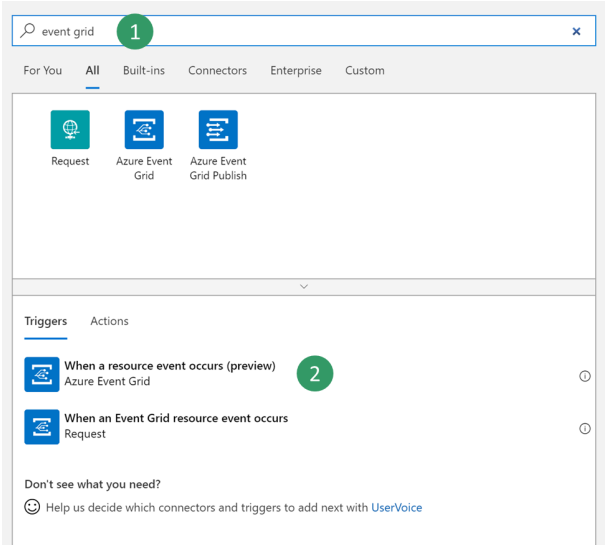
4. Select **All services** > **Integration** > **Logic Apps**.

5. Create a new logic app in your resource group.

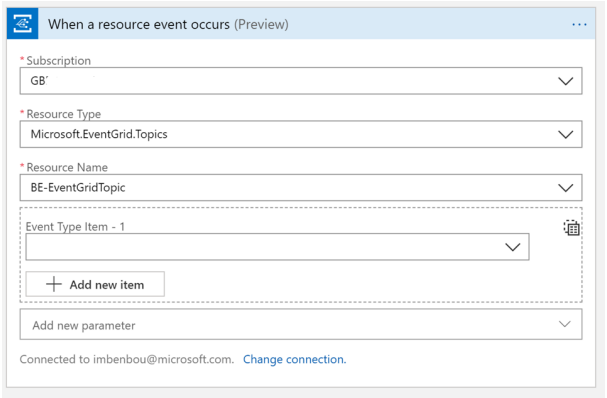


6. After your logic app resource has been created, select the option to create a blank logic app.

7. Search for **Event Grid**, and select the **When a resource event occurs (preview)** trigger.



8. Select your subscription, select **Microsoft.EventGrid.Topics** as the resource type, and select the name of the event grid topic that you created in procedure 1.

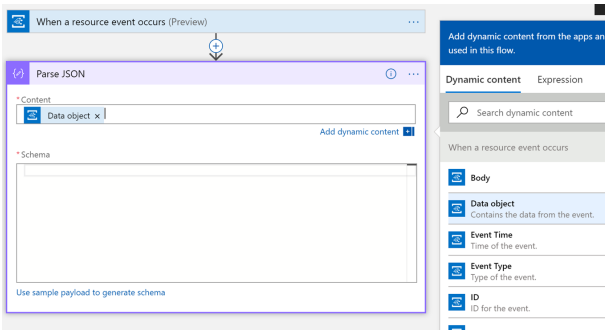


9. Select **New Step** to add a new action.

10. Search for the **Parse Json** data operation. This step is required so that the message can be parsed by using the provided schema for the data contract.

11. Click in the **Content** field of the **Parse Json** action. The pane that appears gives you the option form the previous trigger. You must select the **Data object** field of the event grid message that contains the

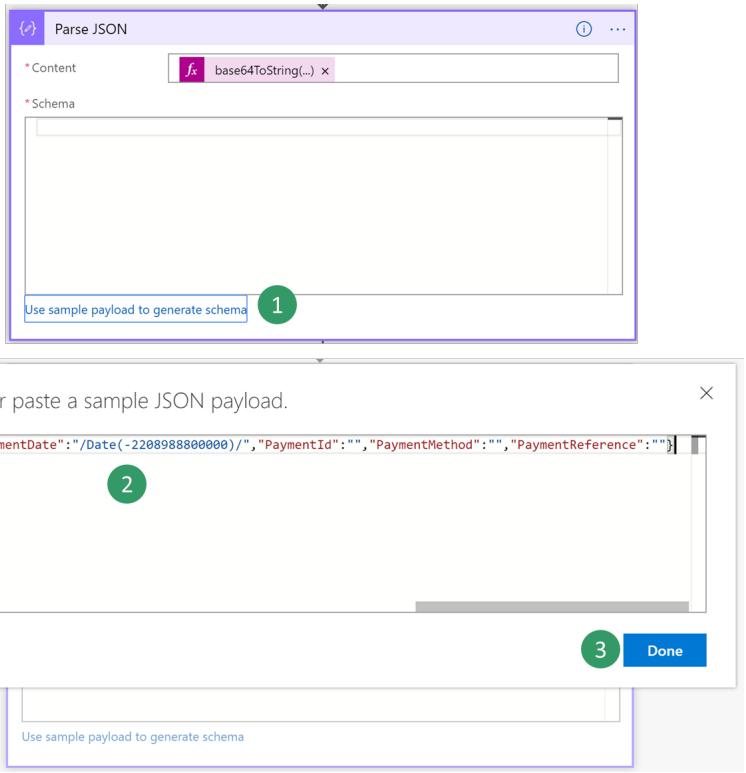
payload that is transmitted by Finance and Operations.



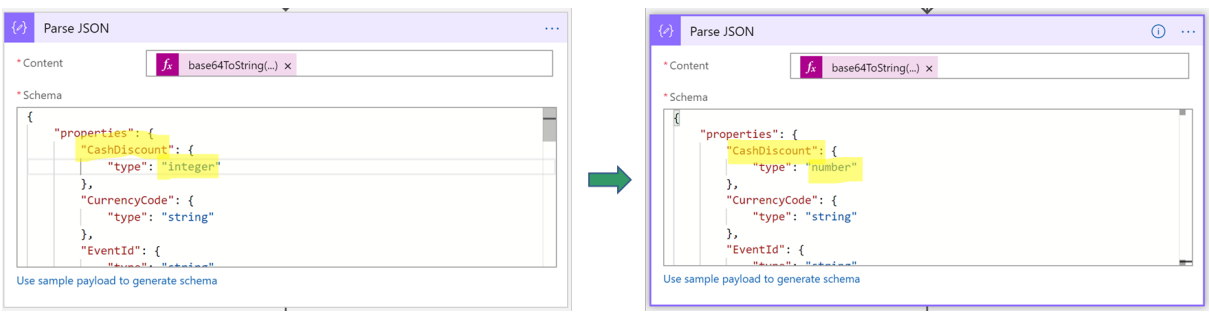
Next, you must enter the provided schema for the contract. This is only a sample payload. However, you can use a capability of Azure Logic Apps to generate a schema from a payload.

12. Select your event in the business event catalog, and then select the **Download schema** link. A text file is downloaded. Open the text file, and copy the contents.

13. Go back to Logic Apps, and select the **Use sample payload to generate schema** link. Paste the contents of the text file, and then select **Done**.



14. Depending on the quality of your sample payload, your generator won't be able to distinguish between an integer and a real value, especially if the real value is provided as a whole number in the sample payload. Review the schema that is generated, and determine whether you must change a field of the **integer** data type to the **number** data type. (In JavaScript Object Notation [JSON], the **number** data type represents real values.)



Next, you will select a final action, such as sending a notification email that includes customer payment

details.

15. Search for the **send email** action, and then sign in to your Microsoft 365 account.
16. Fill in the message with the required fields.
17. Save your logic app.
18. Trigger the business event by posting a customer payment. Then verify that the logic app runs, and that you receive an email that includes customer payment details.

Business events and Azure Service Bus

2/18/2021 • 8 minutes to read • Edit Online

This topic explains how to configure a Microsoft Azure Service Bus endpoint and how to consume a business event from Service Bus.

Scenario overview

Security best practices recommend that you store connection strings outside applications, in an Azure Key Vault drive, and that you give applications the correct access to the key vault keys, secrets, or certificates.

Here are two of the many benefits of this approach:

- Someone who gets access to the application database won't be able to get the third-party connection string.
- Maintenance is easier, especially when multiple applications access the same resources, because you must update connection strings in only one place.

Here is an overview of the procedures that you must complete:

1. Create a new Service Bus namespace.
2. Create a new Service Bus topic and subscription.
3. Create a new key vault to store the Service Bus key.
4. Register an Azure app that has permission to access the key vault.
5. Configure a Business Events endpoint.
6. Consume the business event.

Create a new Service Bus namespace

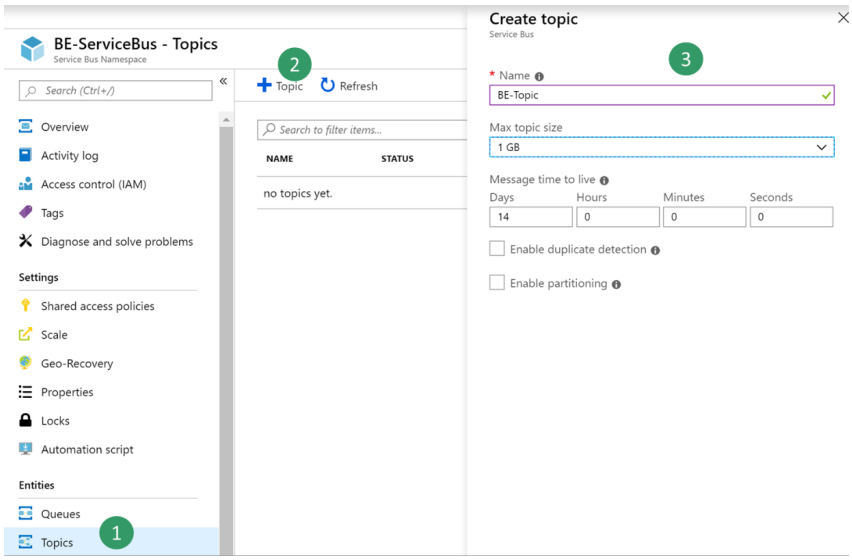
1. Sign in to the Azure portal.
2. Select **All services** > **Integration** > **Service Bus**.
3. Select **Add** to create a new Service Bus namespace, and set the parameters. Select the **Standard** pricing tier. You can create a new resource group as a container for your lab, or you can use an existing resource group.

   > **NOTE**
   >
   > If you select the **Basic** pricing tier, you can create only queues. To create topics, you must select the **Standard** pricing tier.

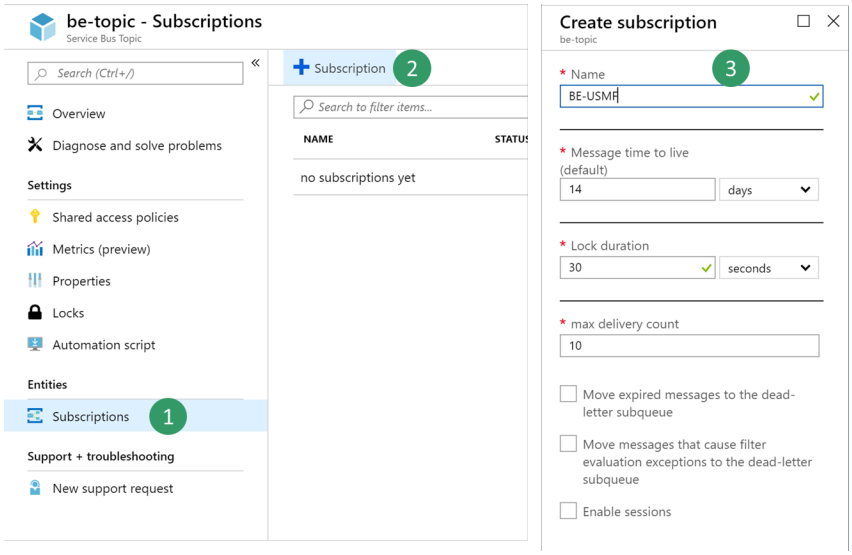
4. When you've finished setting all the parameters, select **Create**.

Create a new Service Bus topic and subscription

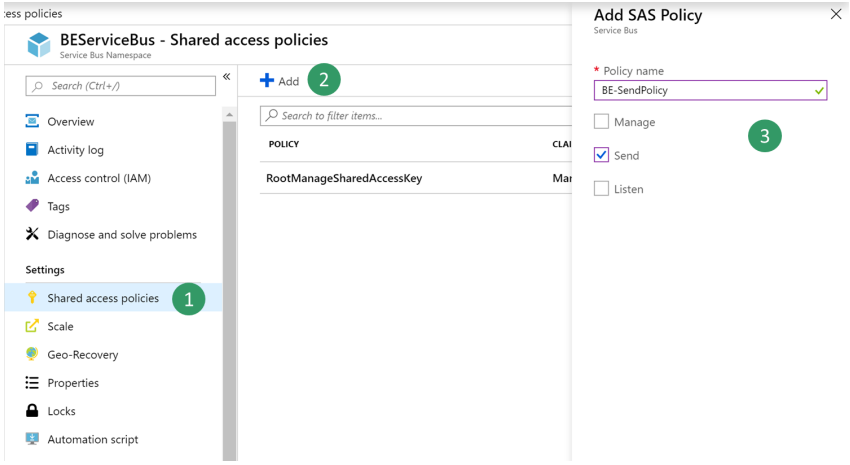
1. In the Azure portal, select the Service Bus that you just created, and then create a new topic.



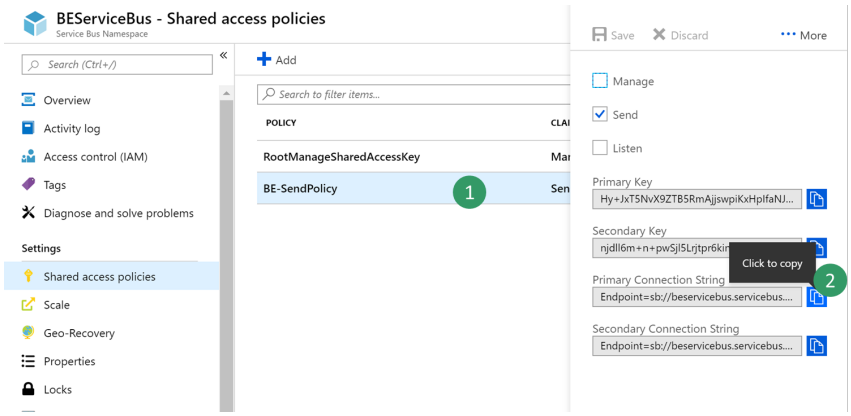
2. Select the new topic, and then create a new subscription that is named **BE-USMF**.



3. Go back to the blade for your Service Bus, and create a new shared access policy to send events. Only the **Send** policy is required to send events to the Service Bus topic.



4. Select the new **Send** policy, and then copy and save the **Primary Connection String** value. You will use this value later.



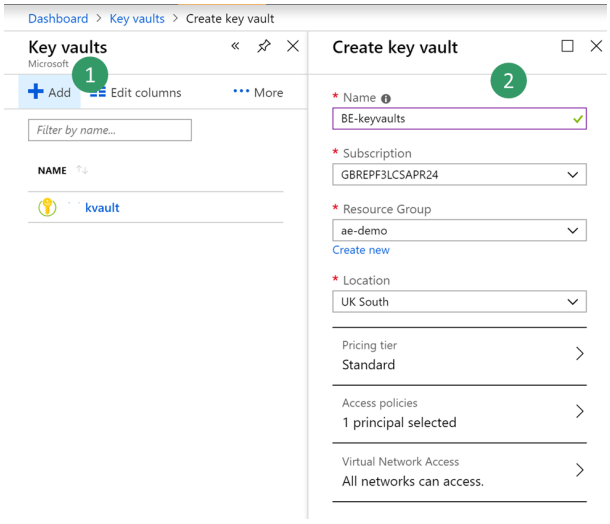
NOTE

The shared access policy must be at the name space level and not at the topic level. If the shared access policy from the topic level is used, the trailing string with semi colon EntityPath= must not be included when configuring the endpoint for business events.

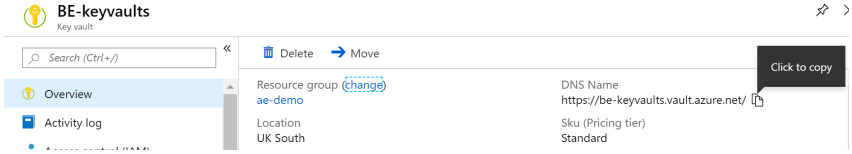
Create a new key vault

In this procedure, you will create a key vault to store the key that you copied in the previous procedure. A key vault is a secure drive that is used to store keys, secrets, and certificates. Instead of storing the connection string, a more typical and more secure approach is to store it in a key vault. You can then register a new application with Azure Active Directory (Azure AD) and grant it the right to retrieve the secret from the key vault.

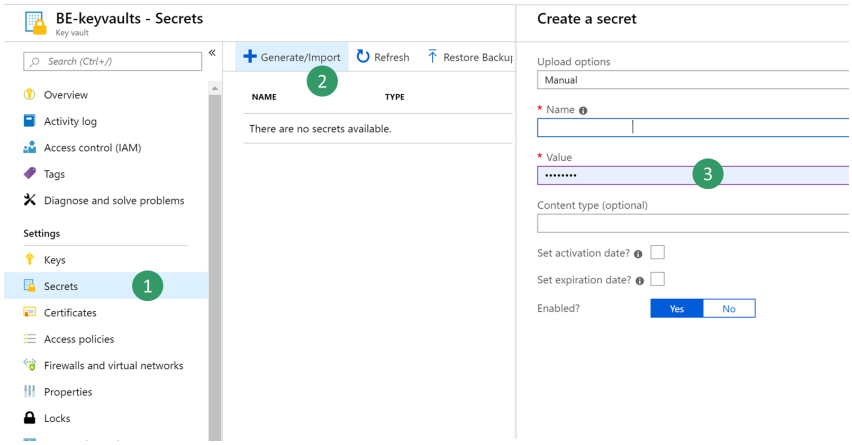
1. In the Azure portal, select **All services** > **Security** > **Key vaults**.
2. Create a new key vault in your resource group and set the default parameters.



3. Select **Overview**, then copy and save the **DNS Name** value for the key vault. You will use this value later.



4. Select **BE-key vault** > **Secrets** > **Generate/Import**. Enter a name for your secret, and paste the Service Bus connection string that you saved earlier.

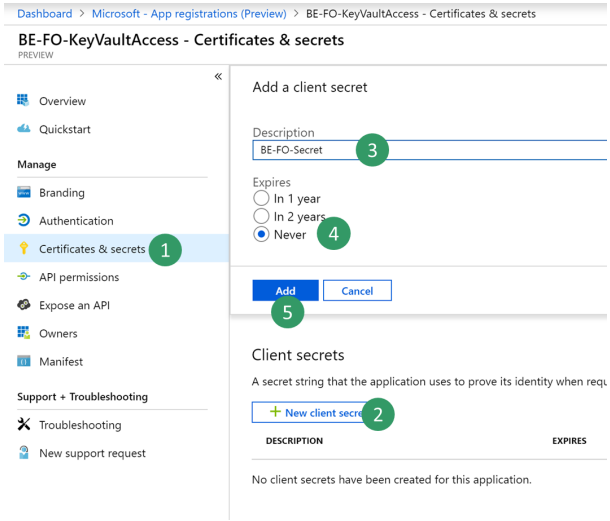


5. Select **Create**.

Register a new application

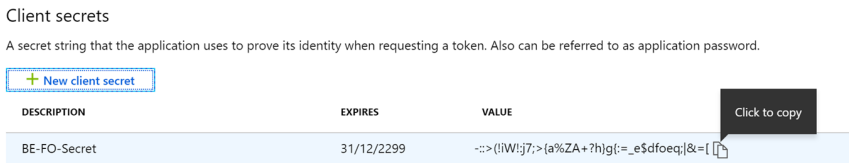
In this procedure, you will register a new application with Azure AD, and give it read and retrieve access to key vault secrets. Finance and Operations will then use this application to retrieve Service Bus secrets.

1. In the Azure portal, select **All services > Security > Azure Active Directory**.
2. Select **App registrations (preview) > New registration**, and enter a name for your application.
3. Select **Register**.
4. Select the new application, and then select **Certificates & secrets > New client secret**. Enter a name for your secret, and set the secret so that it never expires. Then select **Add**.

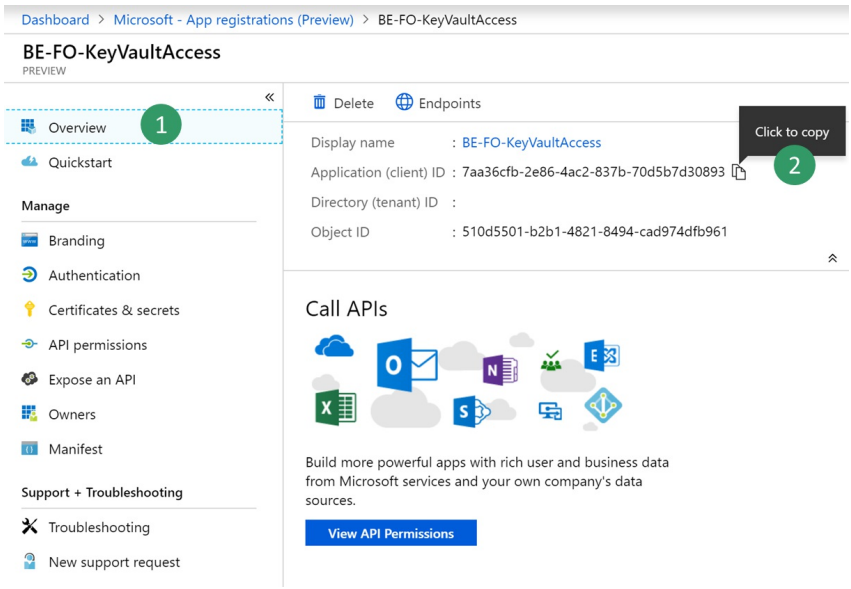


5. Copy and save your new secret. You will use it later.

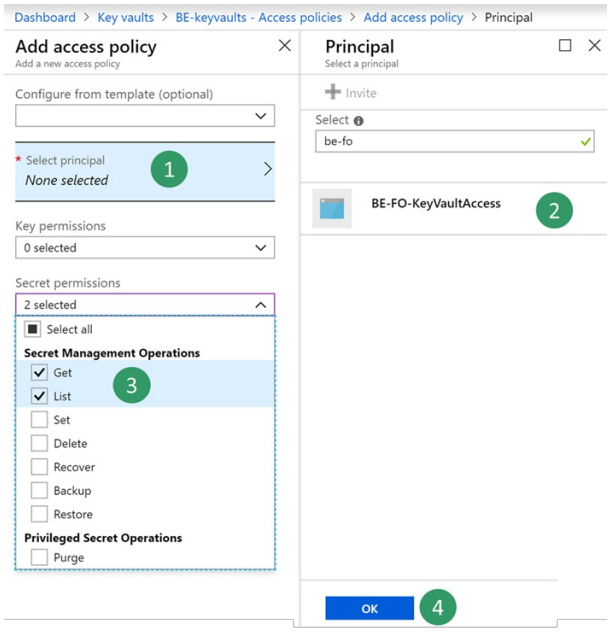
> **IMPORTANT**
>
> Secrets are visible only one time. If you forget to copy the secret, you will have to delete it and create a new secret.



6. Select **Overview**, and copy and save the application ID. You will use this value later.



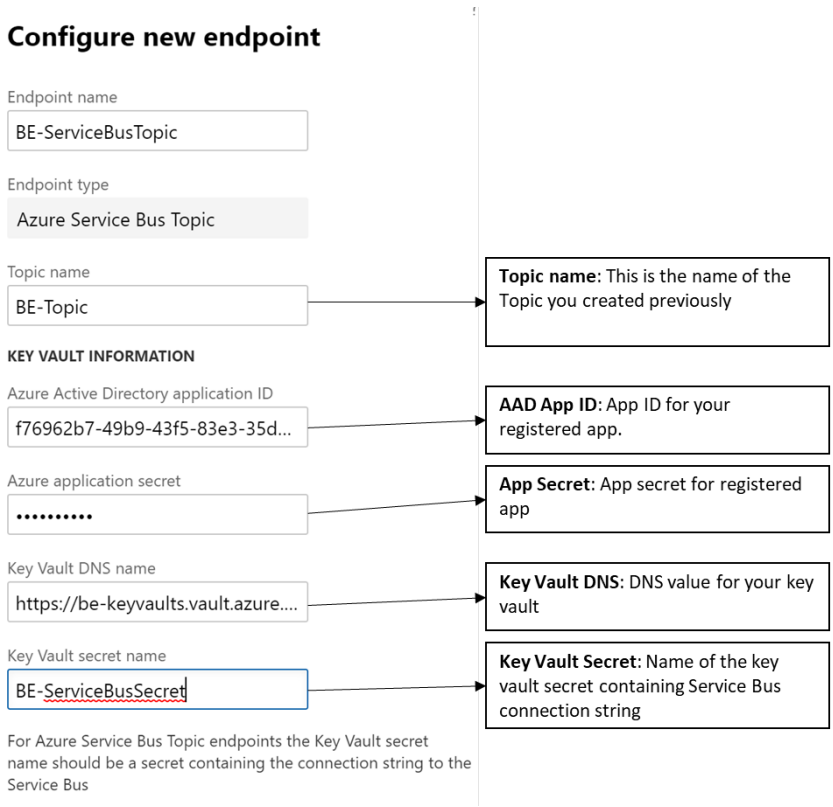
7. Select **All services** > **Security** > **Key vaults**.
8. Select the key vault that you created earlier, and then select **Access policies** > **Add new**.
9. On the **Principal** blade, select your new registered application. Select the check boxes for the **Get** and **List** secret permissions to retrieve key vault secrets.



10. Save your new access policy.

Configure a Business Events endpoint

1. Sign in to the application and go to **System administration** > **Setup** > **Business events**.
2. Select **Endpoints**.
3. Select **New**.
4. Select **Azure Service Bus Topic**.
5. Select **Next**.
6. Set the required parameter values.

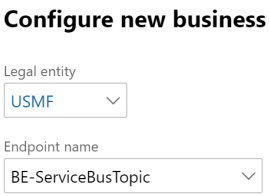


7. Select **OK**.

Consume a business event

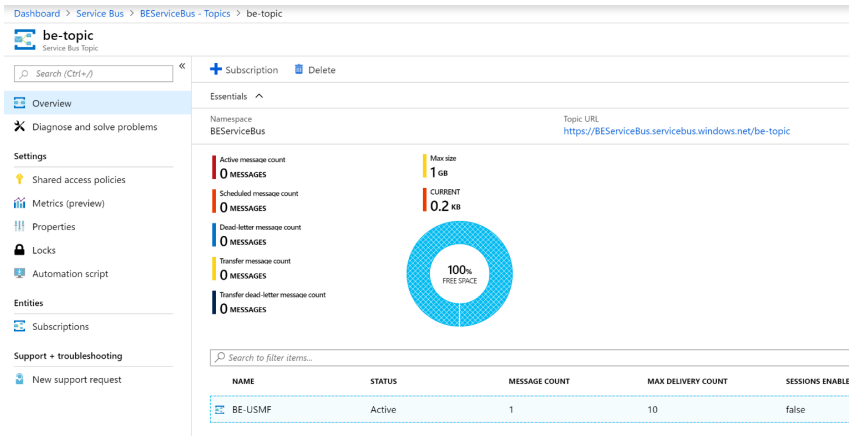
The business scenario involves sending an email or a message to a team channel whenever a customer payment is posted for the USMF company. The message must contain details such as the customer account number, the customer name, and the amount of the payment.

1. Select the business event catalog and look for **customer payment posted** business event
2. Activate the business event for USMF company.

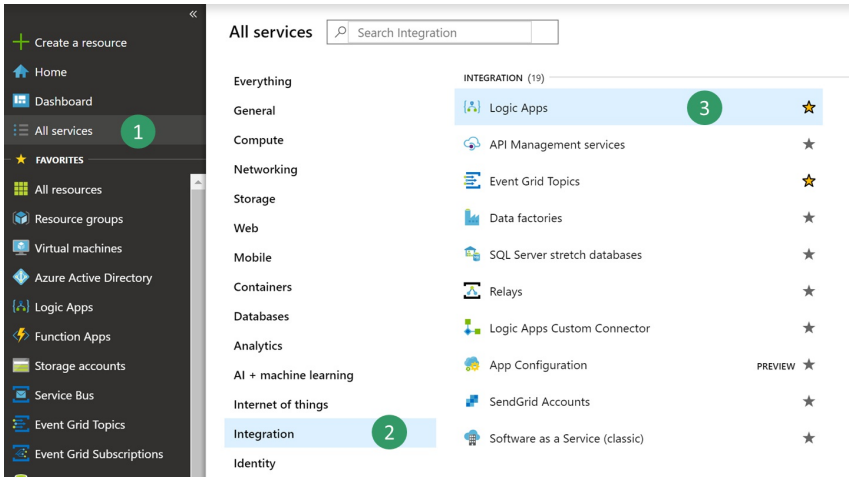


After you activate a business event that uses the new Service Bus endpoint, the application sends a test message to verify that the configuration is accurate and to cache the connection.

3. To verify that the test message has been received, in the Azure portal, select your **BE-Topic** Service Bus topic, and then go into the **BE-USMF** Service Bus subscription that you created earlier. Verify that the message count for the subscription shows a value of at least **1**. If it doesn't, wait for the batch job to pick up your message.



4. Select **All services** > **Integration** > **Logic Apps**.



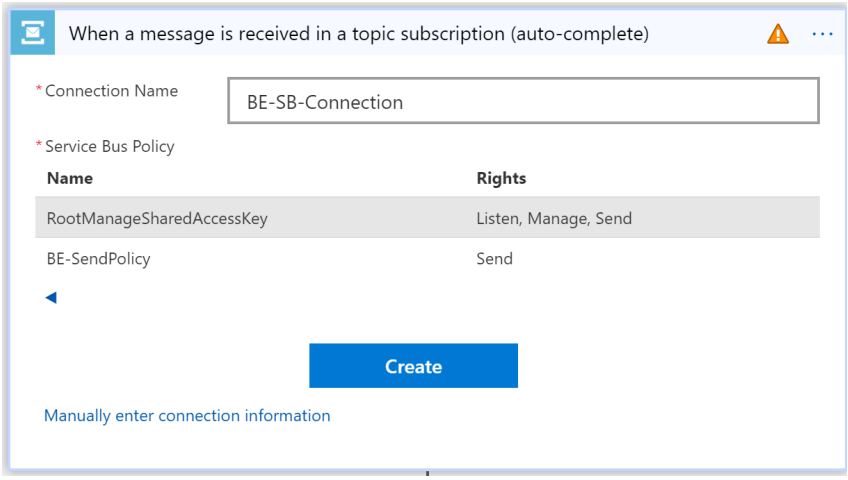
5. Create a new logic app in your resource group.
6. After your Logic Apps resource has been created, select the option to create a blank logic app.
7. Search for **Service Bus**, and select it.
8. Select the trigger that is named **When a message is received in a topic subscription (auto-complete)**.

> **NOTE**
>
> Auto-complete means that the message is deleted from the subscription queue after it's retrieved. Peek-lock authorizes concurrent consumers. It requires a call to the **complete** command of the Service Bus application programming interface (API) in order to delete the message.

Because Logic Apps is accessing your Service Bus for the first time, it asks for a new connection. This connection will cache connection details as a Service Bus namespace URL and credential.

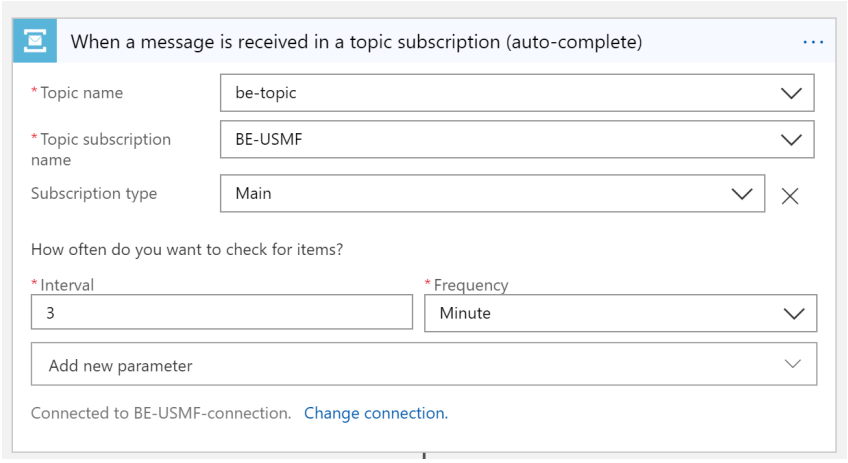
9. Select your Service Bus namespace, and enter a name for the new connection.
10. Select the **RootManageSharedAccessKey** policy for your logic app, and then select **Create**.

> **NOTE**
>
> The **Send** policy can't be used here, because you want to *retrieve* messages, not send them. As a best practice, you could have created a new policy for this use case and given it **Listen** permission only.



11. Select your trigger parameters. Be sure to use the correct names for the topic and subscription that you created.

    This API polls Service Bus for new messages at a configurable recurrence (by default, every three minutes). If the volume of messages is low, the API will have a cost impact for unnecessary triggers, because Logic Apps is priced per trigger call and action run. However, you can implement a push architecture that uses Azure Event Grid in the middle. Service Bus can then push events to Event Grid when there are messages in a queue or a subscription. For more information, see Azure Service Bus to Event Grid integration overview.

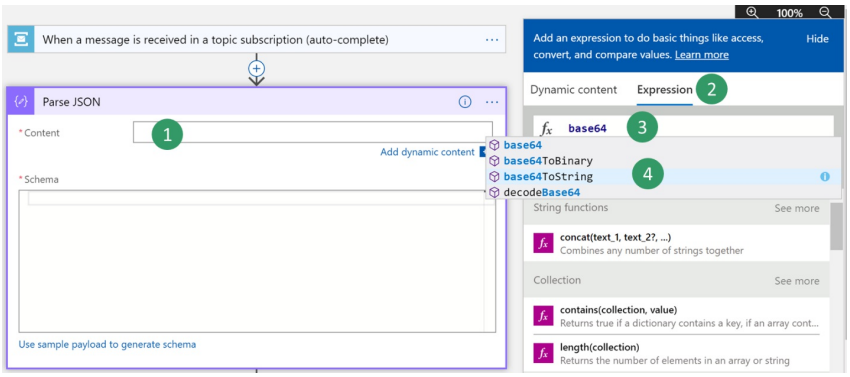


12. Select **New step** to add a new action.

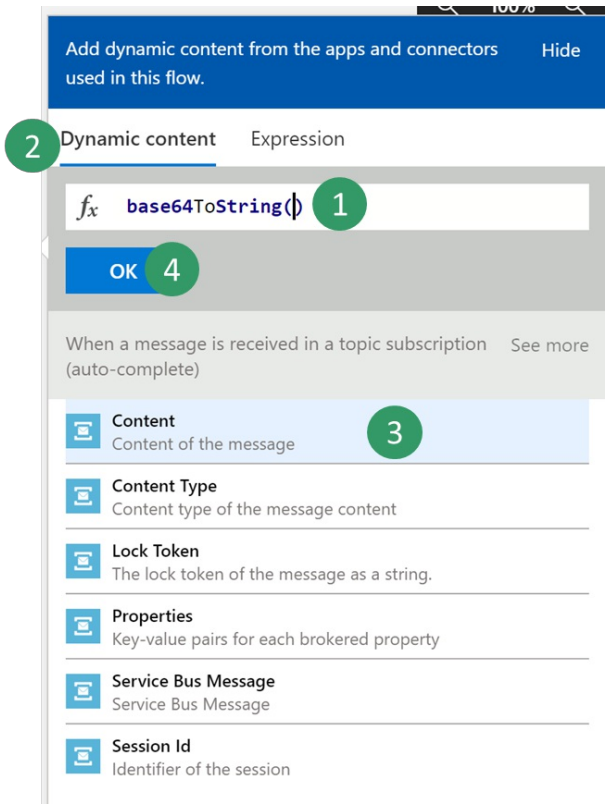
13. Search for the **Parse Json** data operation. This step is required so that the message can be parsed by using the schema of the data contract.

    The body content that is received from the Service Bus is encoded into base64 format. Therefore, you must transform it to string format before the JavaScript Object Notation (JSON) payload can be parsed.

14. Click in the **Content** field, and then, in the pane that appears, on the **Expression** tab, enter the following expression: **Base64ToString()**



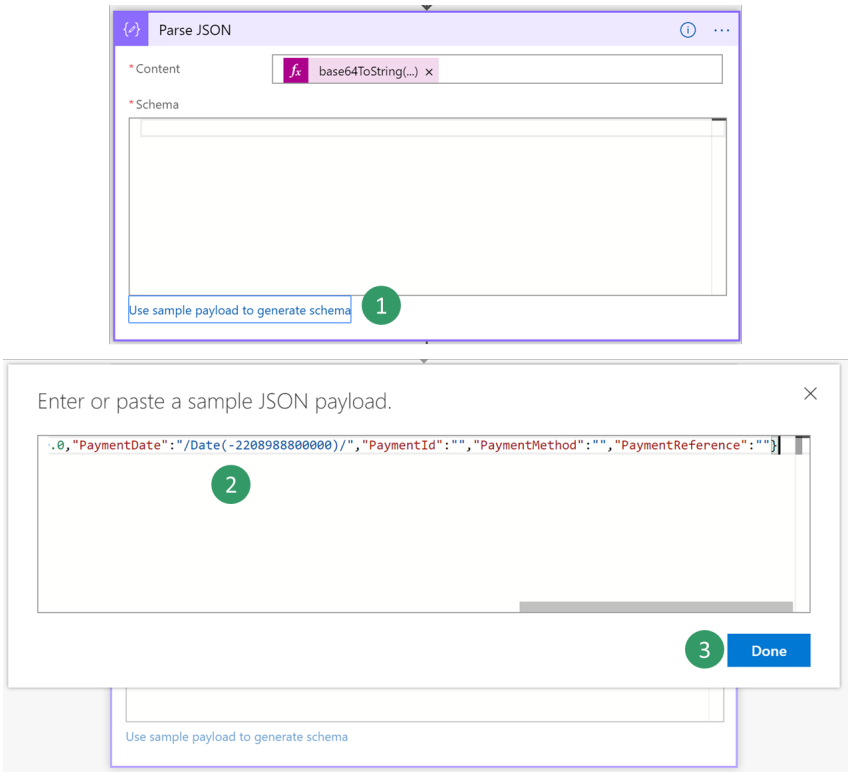
15. Put the cursor between the parentheses in the expression, and then, on the **Dynamic content** tab, find and select the **Content of the message** content from the previous Service Bus trigger. Then select **OK**.



Next, you must enter the schema of the contract that is received from the application. The application only provides a sample payload. However, you can use a capability of Azure Logic Apps to generate a schema from a payload.

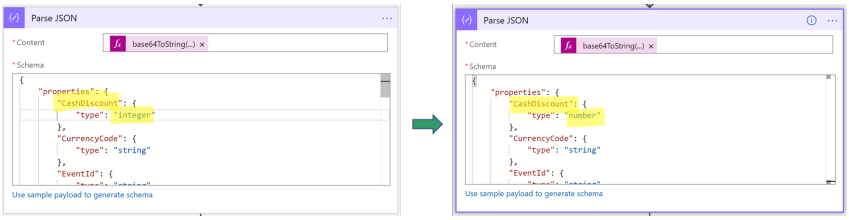
16. Select your event in the business event catalog, and then select the **Download schema** link. Open the text file that is downloaded, and copy the contents.

17. Go back to your Logic Apps, and select the **Use sample payload to generate schema** link. Paste the contents of the text file, and then select **Done**.



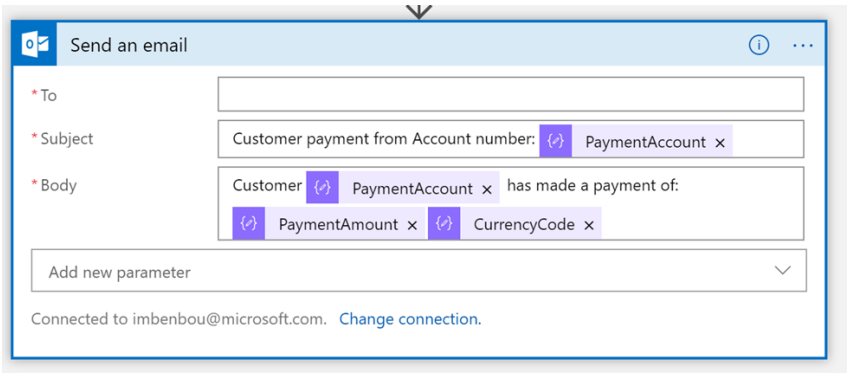
18. Depending on the quality of your sample payload, your generator won't be able to distinguish between

an integer and a real value, especially if the real value is provided as a whole number in the sample payload. Review the schema that is generated, and determine whether you must change a field of the **integer** data type to the **number** data type. (In JSON, the **number** data type represents real values.)



Next, you will select a final action, such as sending a notification email that includes customer payment details.

19. Search for the **send email** action, and then sign in to your Microsoft 365 account.
20. Fill in the message with the required fields.



21. Save your logic app.
22. Trigger the business event by posting a customer payment. Then verify that the logic app runs, and that you receive an email that includes customer payment details.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Business events and workflow approvals

2/18/2021 • 5 minutes to read • Edit Online

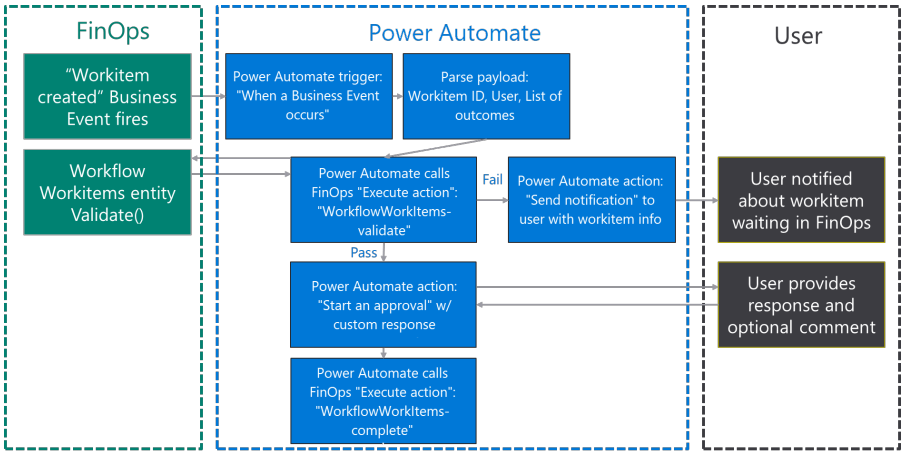
This topic explains how to use Microsoft Power Automate to configure and consume a workflow business event for purchase requisition approval.

To complete this topic, you must be running Microsoft Dynamics 365 for Finance and Operations version 10.0.2 (May 2019) with platform update 26 or later.

Scenario overview

The following illustration shows the high-level process that you must configure by using Power Automate. Note the following points:

1. The application fires a business event whenever a new approval starts.
2. Power Automate trigger starts.
3. After parsing business event payload from F&O, next step is to check wheter the workflow instance ID received from F&O is still alive. This is a security step in case approval has already taken place or workflow has been recalled.
4. If the check is unsuccessful, an email is sent to notify the user about a potential work item in their workspace.
5. If the check is successful, a new Power Automate approval is started.
6. Then workflow is completed by using the outcome of the approval. The outcome can be either **Approve** or **Reject**.



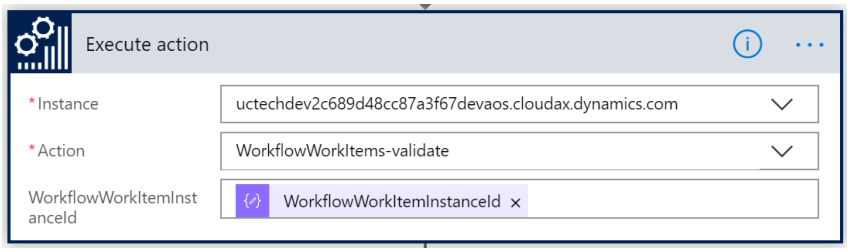
Exercise 1: Create a new flow

1. Sign in to the Power Automate portal.
2. Select an existing environment where you have the right to create a flow resource. The **(default)** environment is available to all companies.
3. Select **New > Create from blank**.
4. Search for **Dynamics 365 for Finance and Operations**, and select the connector.
5. A new trigger is created. This trigger is named **When a Business Event occurs**. Select it.
6. Select the environment instance that has these characteristics:
   - The category is **Workflow workitem**.

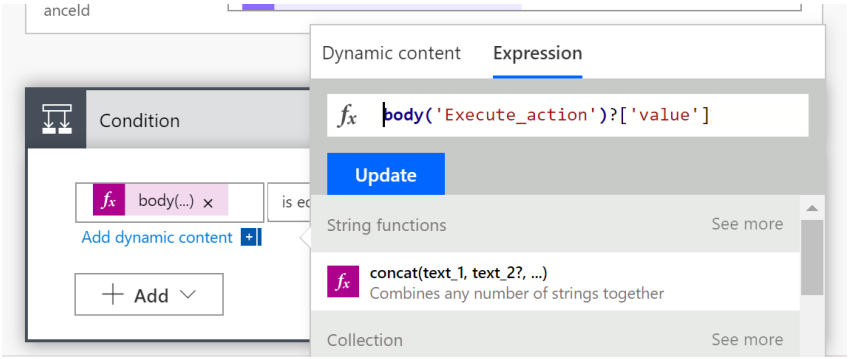
- The event name is **Purchase requisition review (000062) – Approve purchase requisitions**.
- Any legal entity is selected.

7. Select **New Step** to add a new action.
8. Search for the **Parse Json** data operation. This step is required so that the message can be parsed by using the schema of the data contract that the application provides.
9. Select the content field of the **Parse Json** action. The **Body** output from the previous step should appear as an option. Select **Body**.

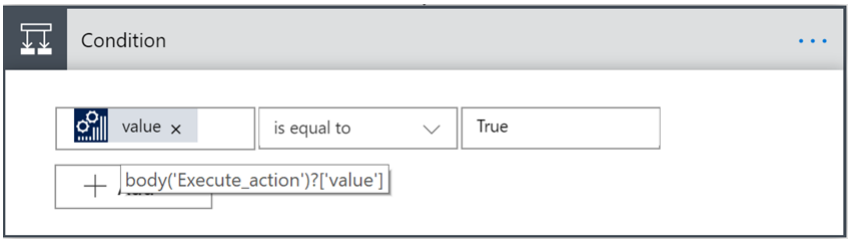
   Next, you must enter the schema of the contract. The application provides only a sample payload. However, you can use a capability of Microsoft Flow to generate a schema from a payload.
10. Select the **000062** workflow event in the catalog, and then select the **Download schema** link. Open the text file that is downloaded, and copy the contents.
11. Go back to Power Automate, and select the **Use sample payload to generate schema** link. Paste the contents of the text file, and then select **Done**.
12. Add a new step to call a workflow action that validates whether a workflow that has the correct instance ID is running and awaiting approval.



13. Add a new condition control step to check the result of the validate action. The dynamic field won't provide the required output. Therefore, you must manually enter the following expression instead: **Body('Execute_action')?['value']**. Then select **OK**.



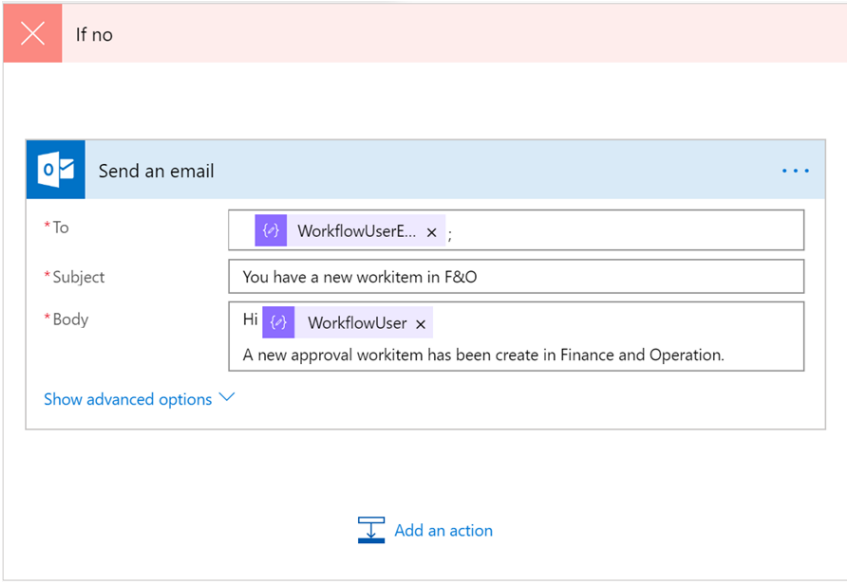
> **NOTE**
>
> The next time that you open the workflow, you will notice that the expression has been updated so that it shows the **value** field. As the following illustration shows, this field will have an application icon.



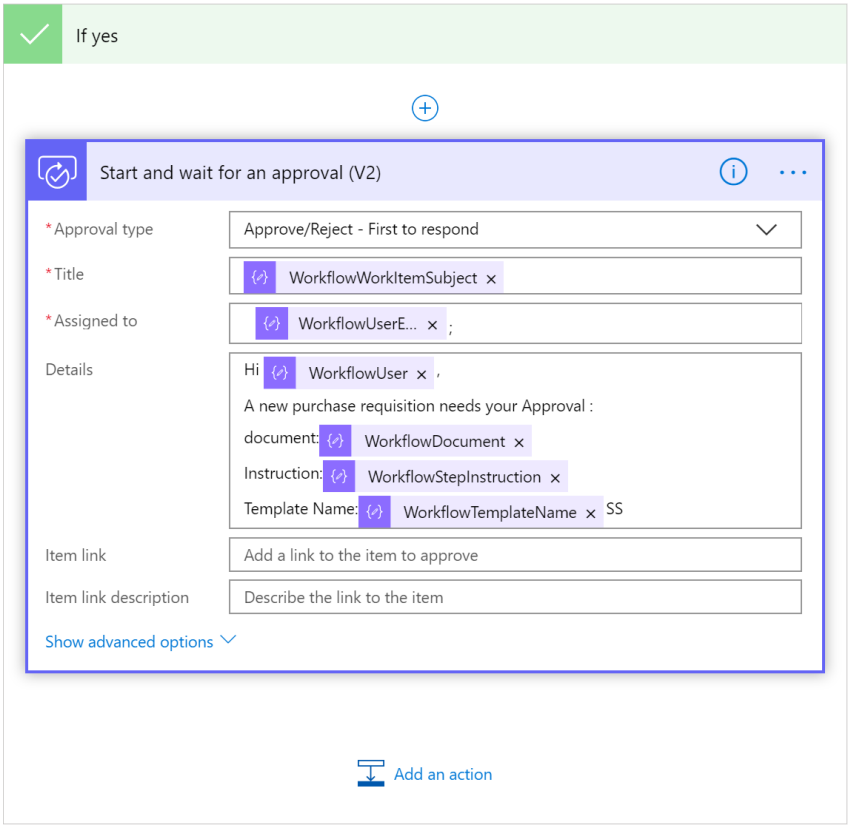
14. The condition control automatically creates two branches for **Yes/No** results. If the result of the validate step is **No**, an email must be sent to the user. This email notifies the user that a new task requires

attention, and that the user must sign in to the client. In order to complete this step create a new send email action within the **No** container and fill in the parameter with the email of the Approver from the previous step **workflowuseremail** and a subject and body of your choice.

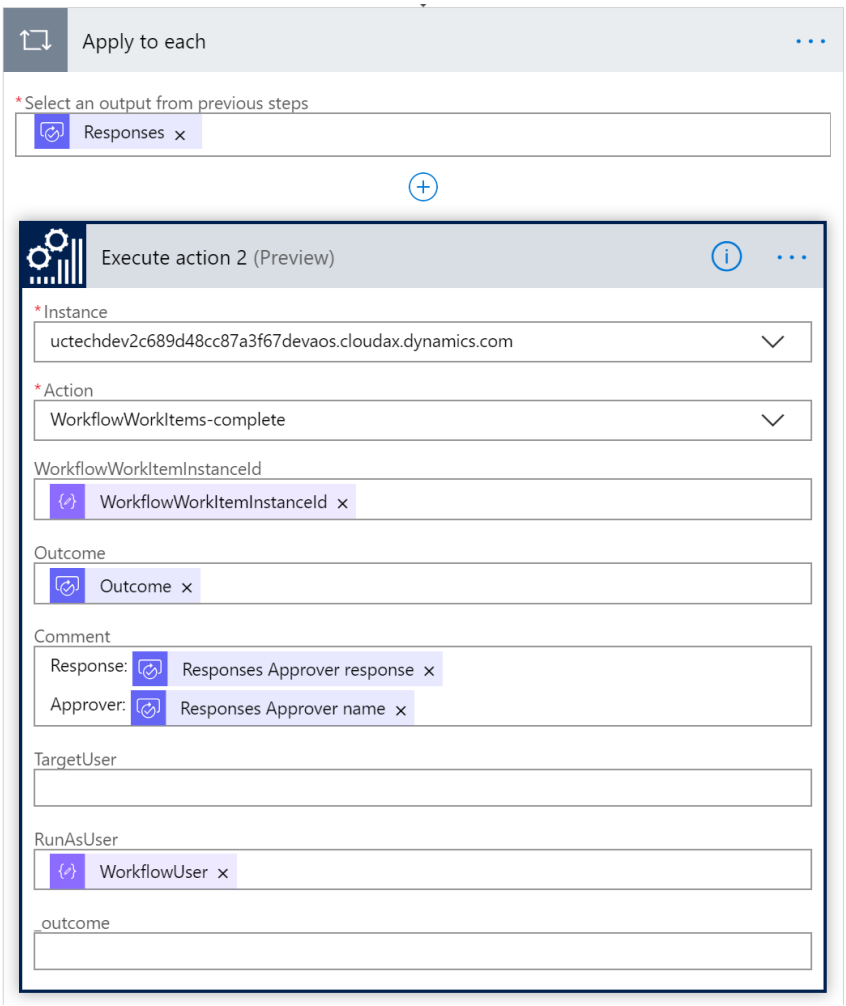
> **NOTE**
>
> The email address that the workflow business event returns is the email address of the workflow approver. If the workflow approver user hasn't been configured in yourdemo environment, you can use your own email address for demo purposes.



15. If the result of the validate step is **Yes**, you must start a new Power Automate approval step. In the **Yes** container, select a new action that is named **Start and wait for an approval (v2)**, and choose inputs as follows: Approval type: Approve/reject: first to respond title: **workflowworkitemsubject** output form Business event payload Assigned to: **workflowuseremail** output Then you can fill in the details section with as much information as needed from previous step such as **workflowdocument** or **workflowstepinstruction**. Again, you can use your own email address in the **Assigned to** field for demo purposes especially if the workflow approver user hasn't been configured in your demo environment.



16. Next, you must complete the workflow approval by using the outcome of the approval step. Still in the **Yes** container, add a new **Finance and Operations Execute Action** step, and choose the **WorkflowWorkitem-complete** action and the **WorkflowWorkitemInstanceID** parameter. Then fill in the rest of the parameters from the approval outputs. As a minimum the outcome section with Approval outcome and the comment section with the approver's responses. Because the approval step can support multiple approvers, the response output is an array. Therefore, as soon as you select the output **Reponses** as an input for the comment section, Power Automate automatically embeds your action in an **Apply to each** container as shown below.



17. Select **Save**.

Exercise 2: Trigger a business event

Power Automate can automatically configure the application for you. After you save your flow, Power Automate creates an endpoint and activates the business event. You don't have to complete any other configurations. You just have to verify that the endpoint has been correctly configured and then trigger an event.

1. Sign in to the client.
2. Go to **System administration** > **Setup** > **Business events**.
3. Select **Business events**.
4. Select **Endpoints**.
5. Verify that a new endpoint has been created, and that a globally unique identifier (GUID) has been appended to the name.
6. On the **Active events** tab, verify that **Workflow workitem** is activated for the USMF company.

Business events and Microsoft Forms Pro

2/18/2021 • 2 minutes to read • Edit Online

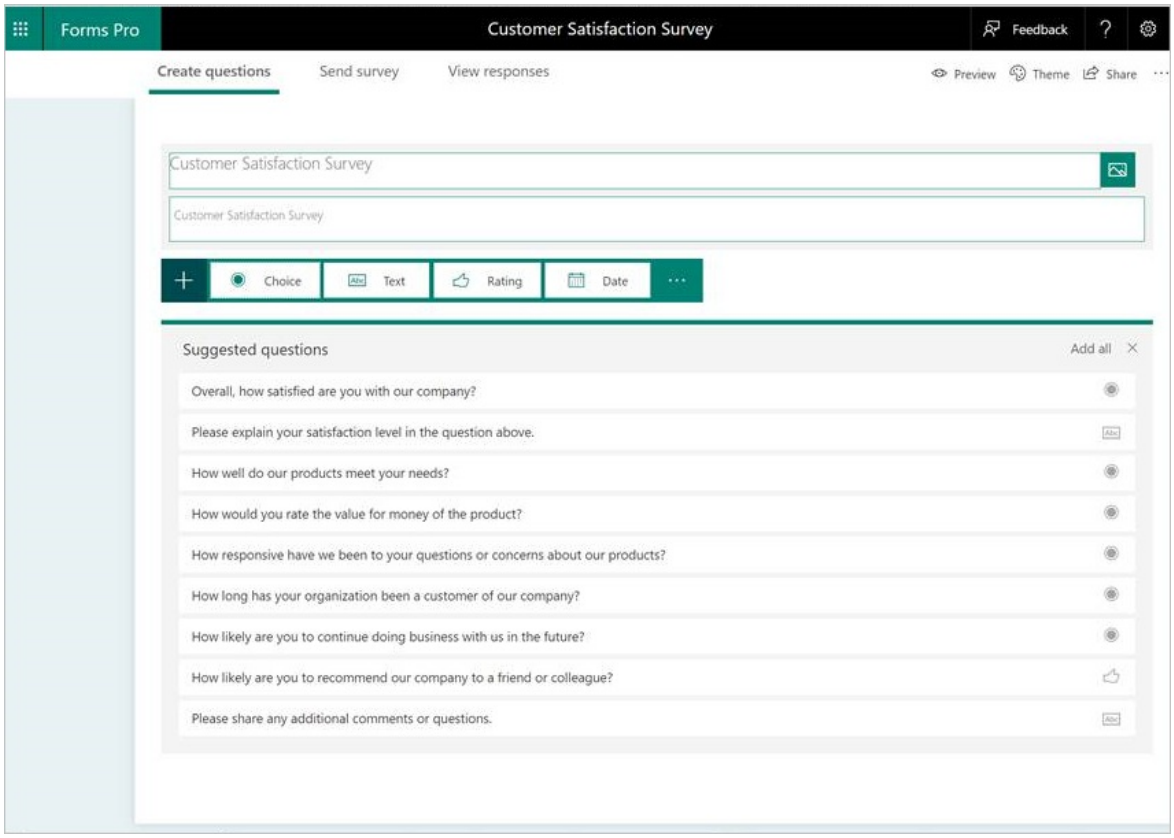
This topic goes through a scenario where Microsoft Forms Pro is used to create a survey that can be used with business events. Specifically, in the scenario that is described here, a survey is sent to customers when a product has been shipped. The survey information is gathered by using Forms Pro.

Prerequisites

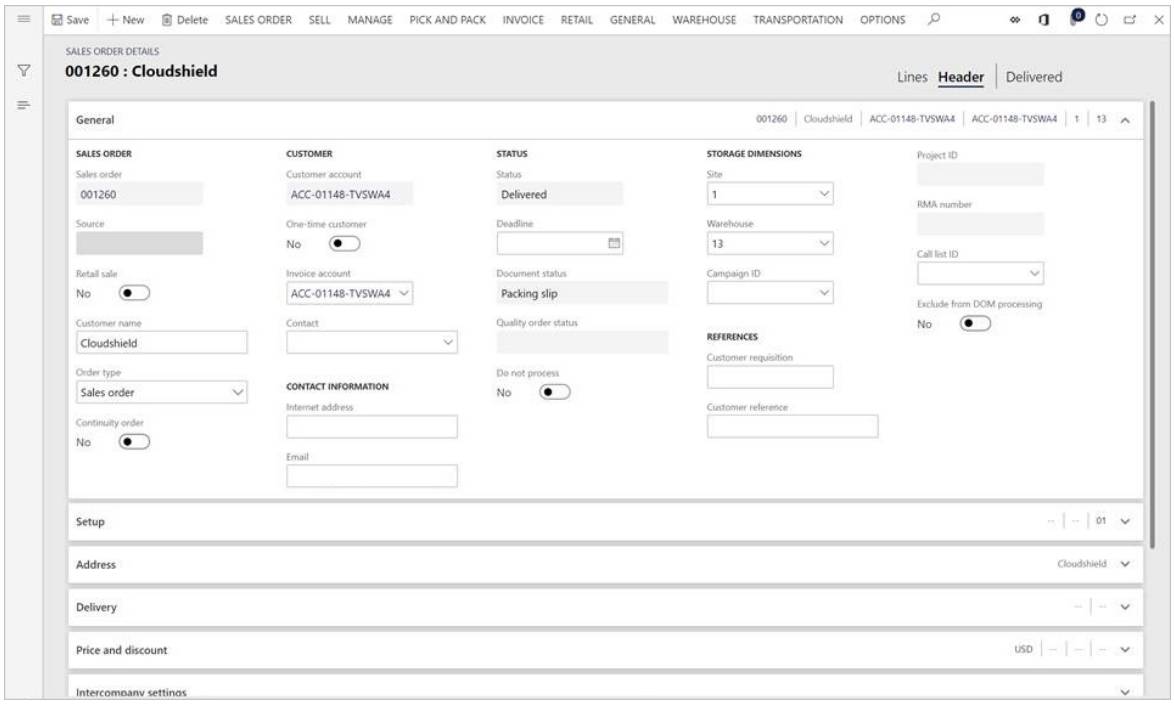
If you haven't used Forms Pro before, you should first read the Forms Pro documentation to learn how to use it.

Scenario

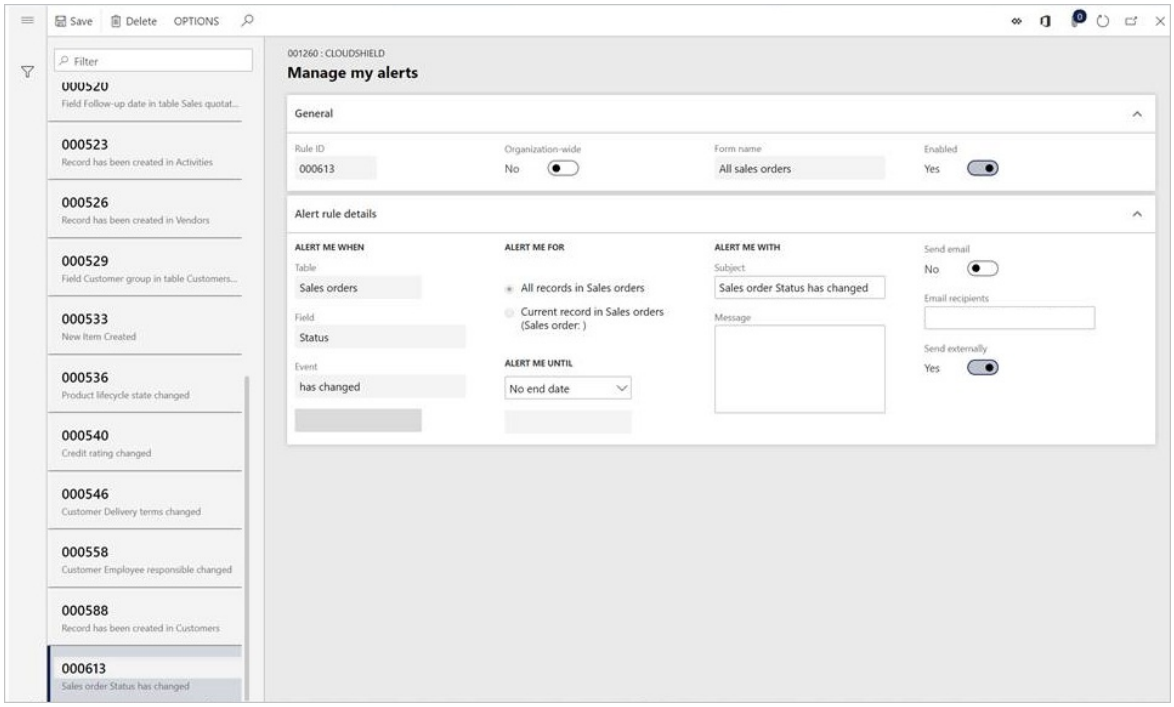
1. Create a survey. Based on the title that you enter for the survey, Forms Pro suggests survey questions.



2. The sales order tracks the shipment. When the product has been shipped, the status of the sales order is changed to **Delivered**.



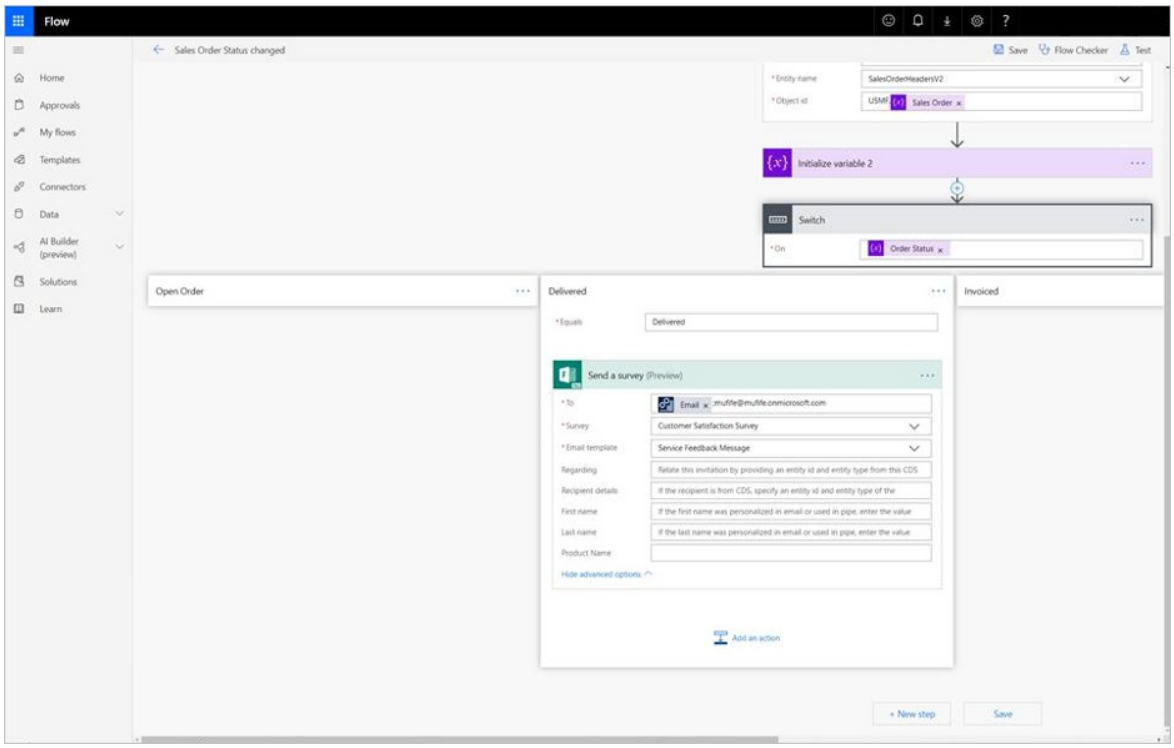
Therefore, configure an alert on the sales order, so that an alert is created whenever the value of the **Status** field is changed. Be sure to set the **Send externally** option to **Yes**, so that the alert will be sent out as a business event.



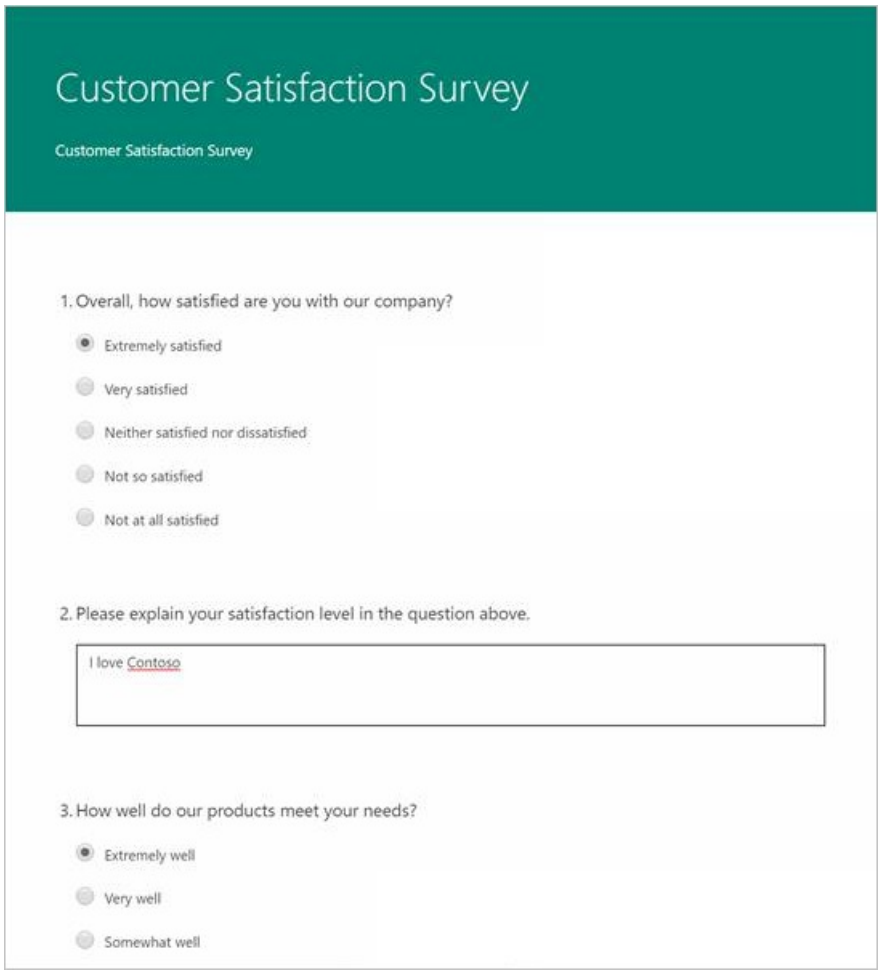
3. Set up the flow that will be triggered by the business event whenever the status of the sales order is updated (see the illustration in the next step). After it's triggered, the flow will use the Forms connector to send the survey to the customer email address that is registered on the sales order.

   The customer email address and other information that is required for the scenario must be in the payload of the business event. If the payload doesn't have this data, it can be extended so that it includes the appropriate fields. For more information, see the Business events developer documentation.

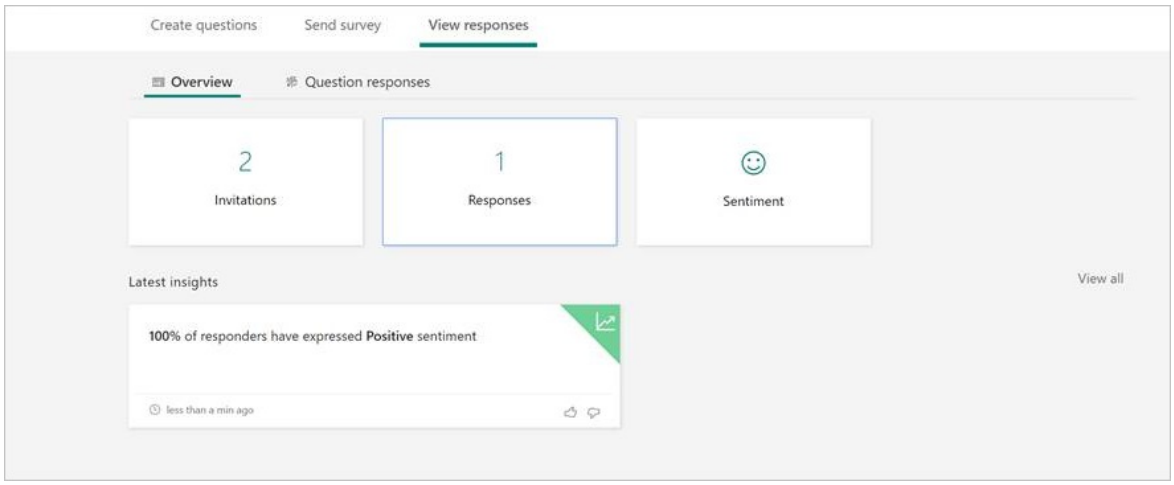
4. Because Microsoft Power Automate is used to orchestrate this scenario, don't activate the **When a change based alert occurs** business event in the application. Instead, set up Power Automate so that it subscribes directly to the business event.



5. After you've finished setting up the flow, it will be triggered and send out the survey whenever the sales order's status is updated.



As users fill in the survey and submit it, Forms Pro shows some analytics.



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

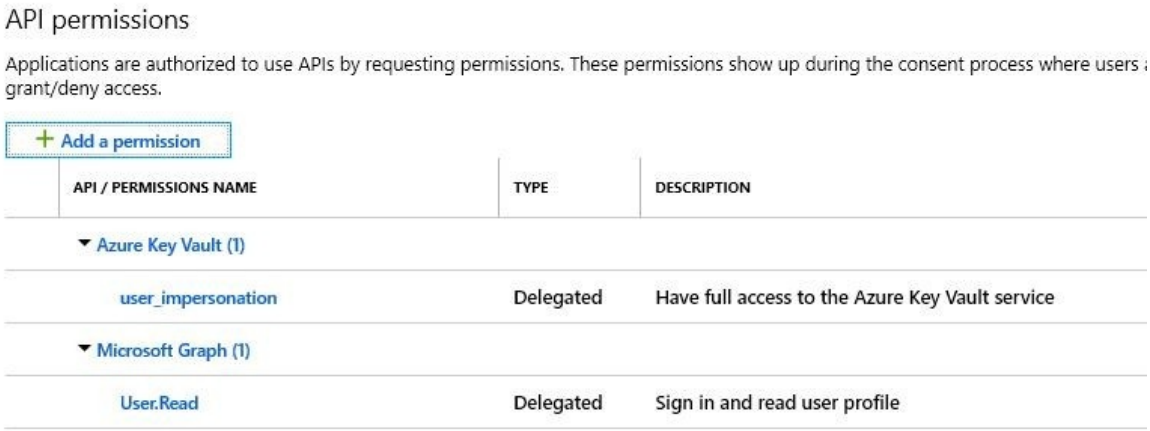
Business events and Azure Event Hubs

2/18/2021 • 2 minutes to read • Edit Online

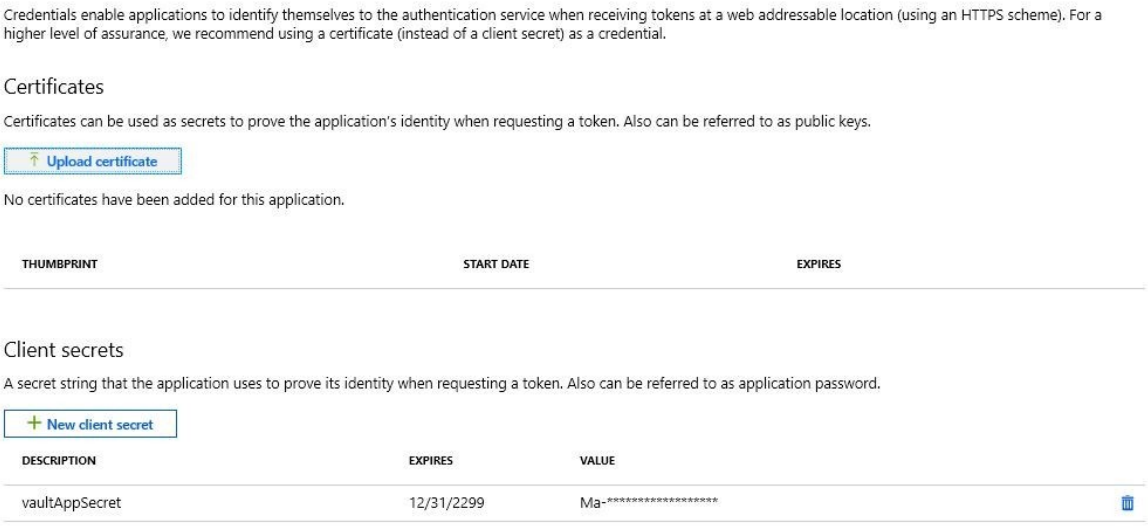
This tutorial describes the steps that you must follow to make business events work with Microsoft Azure Event Hubs.

1. In Azure portal, create an Active Directory application registration. Make a note of the application ID.

2. Give the app, permission to the Azure Key Vault application programming interface (API).

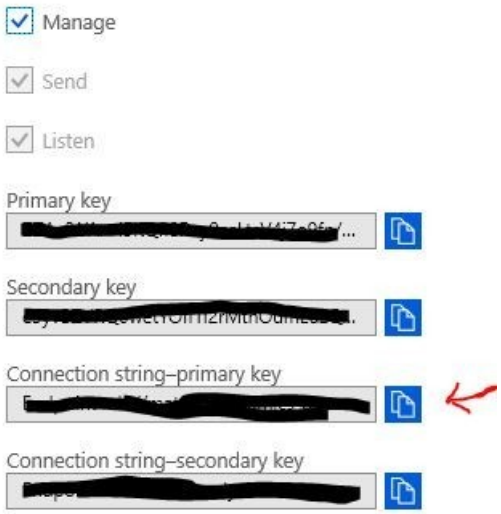


3. In the app registration, create an application secret. Make a note of the value.

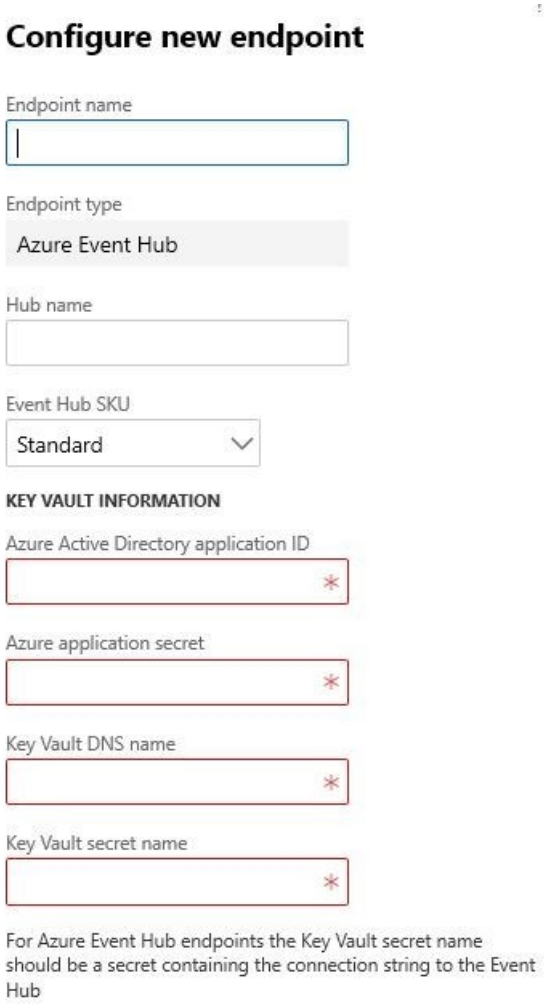


4. In the key vault, give permission to the new app registration.

5. In the key vault, create a new secret. The value of this secret must be the connection string to your event hub. Make a note of the value.



6. Create an endpoint configuration for the event hub. Go to **System administration** > **Setup** > **Business events** > **Business events catalog**, and then, on the **Endpoints** tab, select **New** to open the **Configure new endpoint** wizard.



7. In the **Endpoint type** field, select **Azure Event Hub**.
8. Select **Next**.
9. In the **Endpoint name** field, enter a name for the endpoint.
10. In the **Hub name** field, enter the name of your event hub.
11. In the **Azure Active Directory application ID** field, enter the application ID that was created earlier.
12. In the **Azure application secret** field, enter the value that was created earlier.
13. In the **Key Vault DNS name** field, enter the Domain Name System (DNS) name of your key vault. You can find this value on the **Overview** tab of the key vault configuration in the Azure portal.
14. In the **Key Vault secret name** field, enter the name from the secret that was created earlier.
15. Select **OK**.
16. You can now activate one or more business events that should be sent to this endpoint.

Business events in financial period close

2/18/2021 • 5 minutes to read • Edit Online

This topic explains how to use business events in the financial period close business process to gain insights and provide internal controls.

To complete this topic, you must be running version 10.0.2 (May 2019) with Platform update 26 or later.

Scenario overview

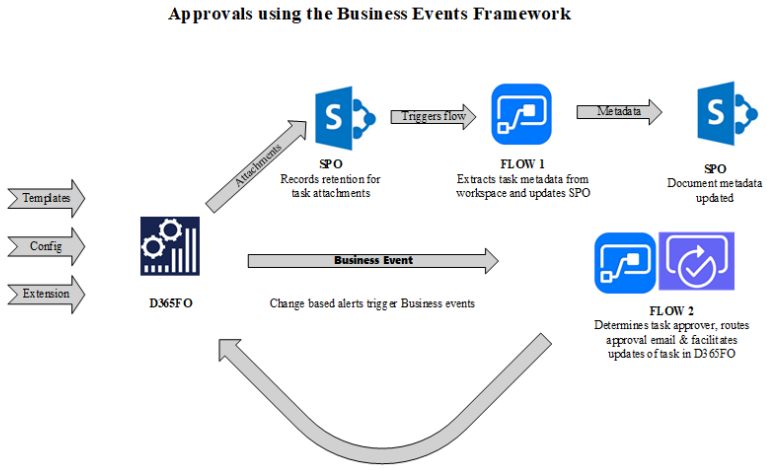
Task management is fundamental to managing business processes across industries. Out-of-box capabilities let users manage business process tasks in a structured manner. The **Financial period close** workspace illustrates these capabilities by offering a central location for managing tasks in a company's accounting period close process.

This topic looks at an organization that recently decided to explore how it can use the **Financial period close** workspace to track and report tasks that are associated with every period close. Performance management and traceability are some of the challenges that this organization faces in the current setup. Therefore, the organization undertook an exercise in business process transformation to identify the capabilities of the **Financial period close** workspace. This exercise revealed the following business requirements:

1. The ability to be notified when tasks must be started
2. The ability to attach documents
3. Record management and disposition capabilities for attachments
4. The ability for multiple approvers to approve tasks, based on predefined logic
5. Task questionnaires for audits
6. Reporting capabilities to track the current status of the period close process and do performance analysis for insights into efficiency

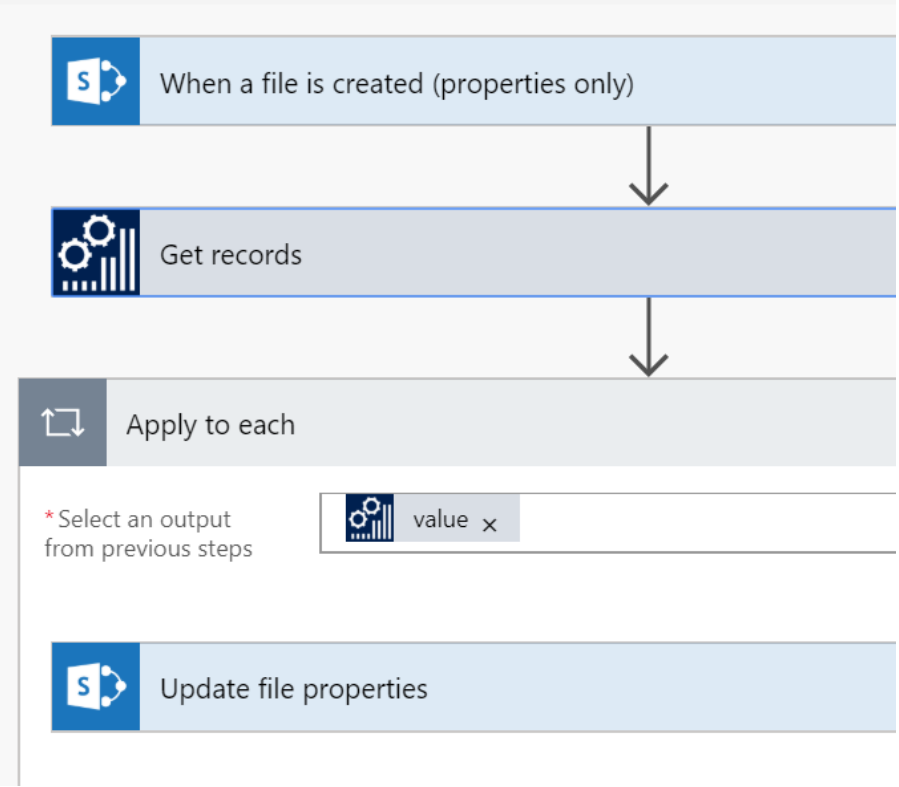
High-level design

To achieve the previously mentioned requirements, the organization used out-of-box capabilities of the **Financial period close** workspace. A gap analysis revealed that, by doing minor extensions to the workspace and the underlying data entities, the organization could achieve requirements 2, 5, and 6, and could partially achieve requirement 4. To achieve requirements 1 and 3, and parts of requirement 4, the organization chose to use Power Automate. The following illustration shows an architectural overview of the solution.



Managing attachments by using Microsoft Power Automate and SharePoint Online

Accountants view their tasks in the **Financial period close** workspace and start to work on them. Attachments are added to the task by using a SharePoint Online document type. SharePoint triggers in Microsoft Power Automate are used to trigger the Power Automate that is shown in the following illustration. This Power Automate updates the SharePoint metadata with metadata from the task in the **Financial period close** workspace. SharePoint columns were created for this purpose in the document library. A separate attachment data entity was created to hold the attachment metadata for every attachment that is added to the **Financial period close** workspace. Fields from the custom entity were mapped to the SharePoint Online columns in the Power Automate. When documents that use the specified document type are created in the predefined SharePoint Online library, Power Automate is triggered, obtains the metadata from the custom data entity, and updates the document's metadata columns in SharePoint Online.



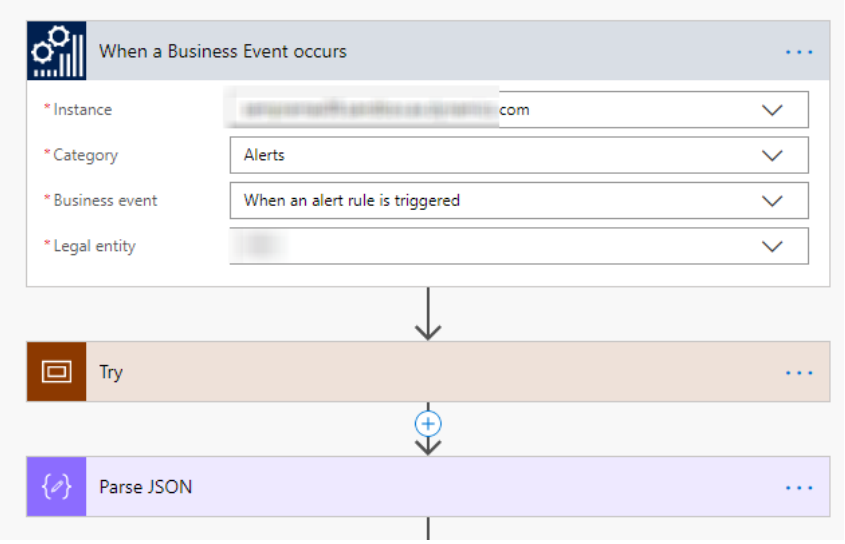
Enabling internal controls by using business events and Power Automate

As accountants complete their tasks, and the tasks become ready for review, the value of the **Review status** custom field is updated to **Ready for review**. The Power Automate gets triggered by the **When the change-based alert is triggered** business event when this update is made. The payload of this business event contains the task name and the area name. The Power Automate uses the combination of the task name and area name, together with the value of the **Review status** field, to route the task through an email-based workflow that is orchestrated by Power Automate. The Power Automate waits for approval, add new comments to the task log, and updates the task in the **Financial period close** workspace , based on both the outcome of the approval process and related metadata. Custom data entities were built in to query and update the **Financial period close** workspace by using Power Automate.

Subscribing to the business event

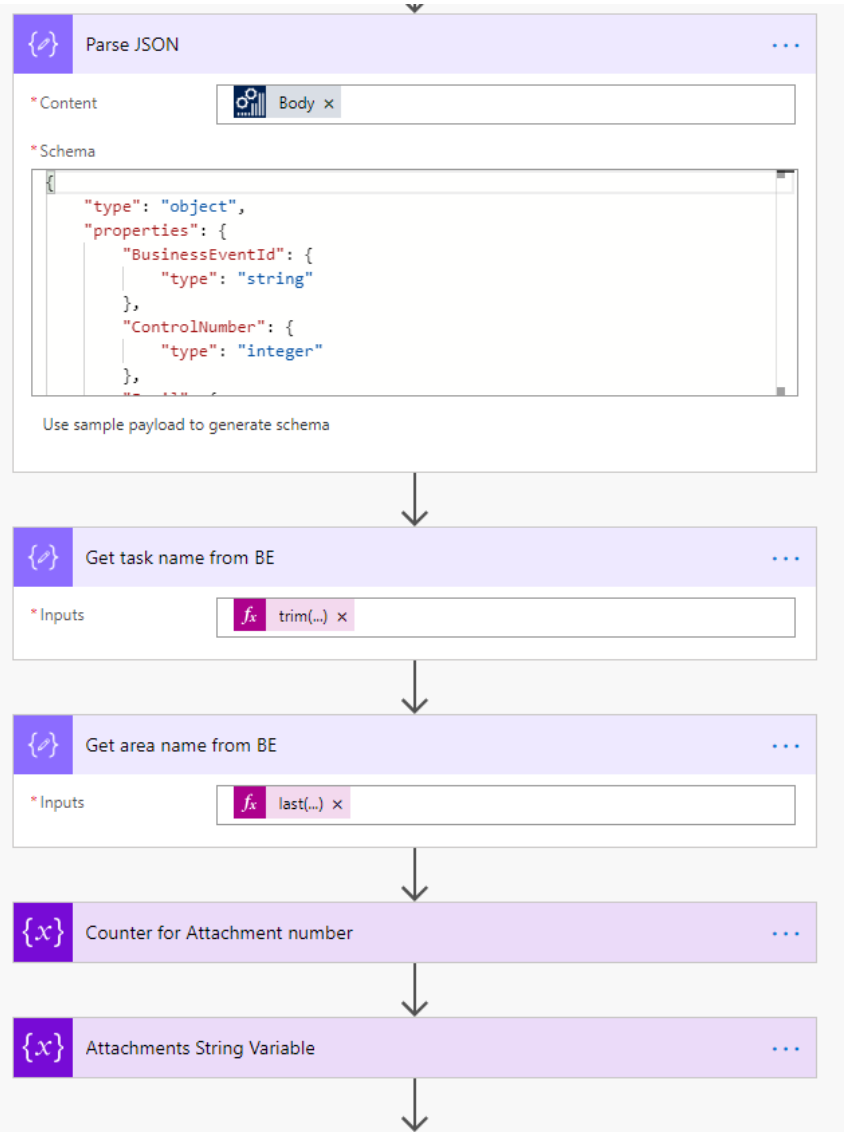
The following example describes the general steps for subscribing to a change-based alert business event.

1. Add the connector trigger to the Power Automate app, and subscribe to the change-based alert business event.



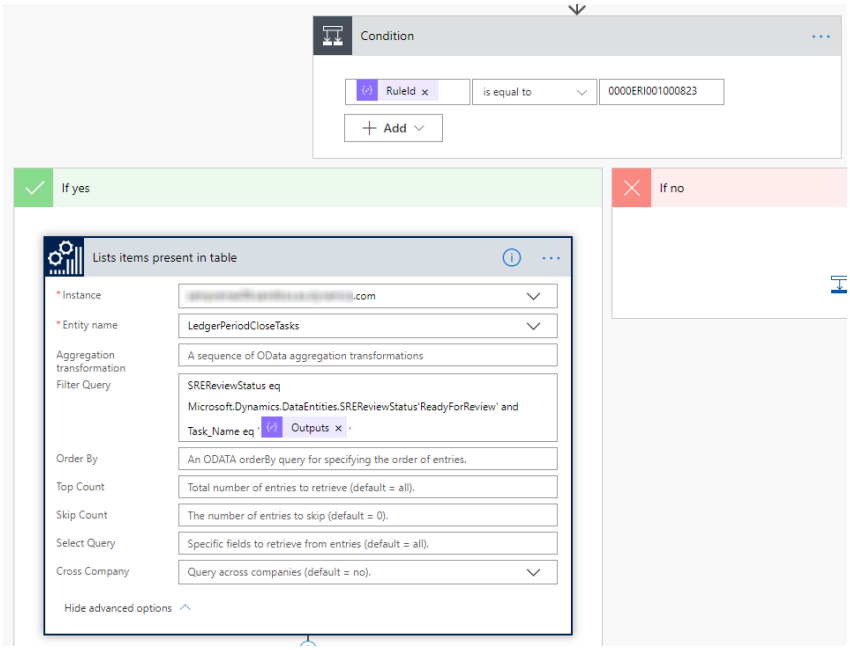
2. Parse the business event payload.

   When the business event is triggered, it triggers Power Automate. This business event contains a payload. In this step, the payload is parsed, and the required variables are initialized.



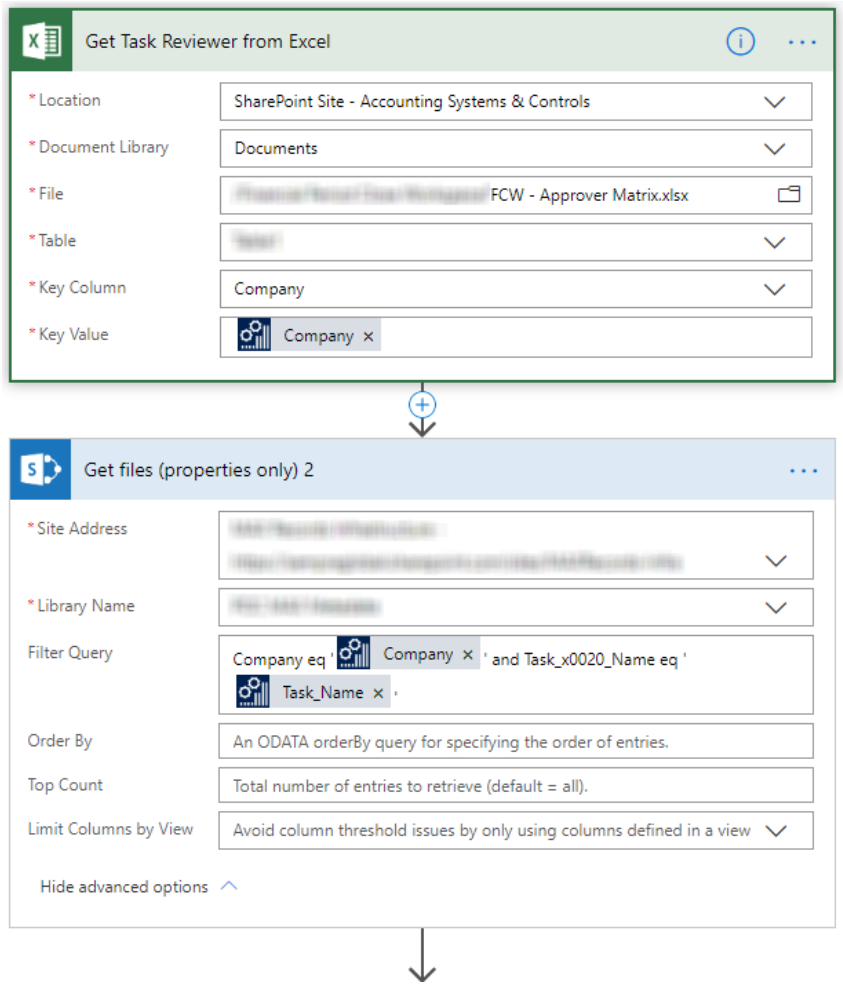
3. Retrieve the task, based on the values from the payload.

   When the task is updated, the business event triggers Power Automate. At that point, after the payload has been parsed, you will know basic information about the task. In this step, the custom data entity is used to retrieve more information about the task.



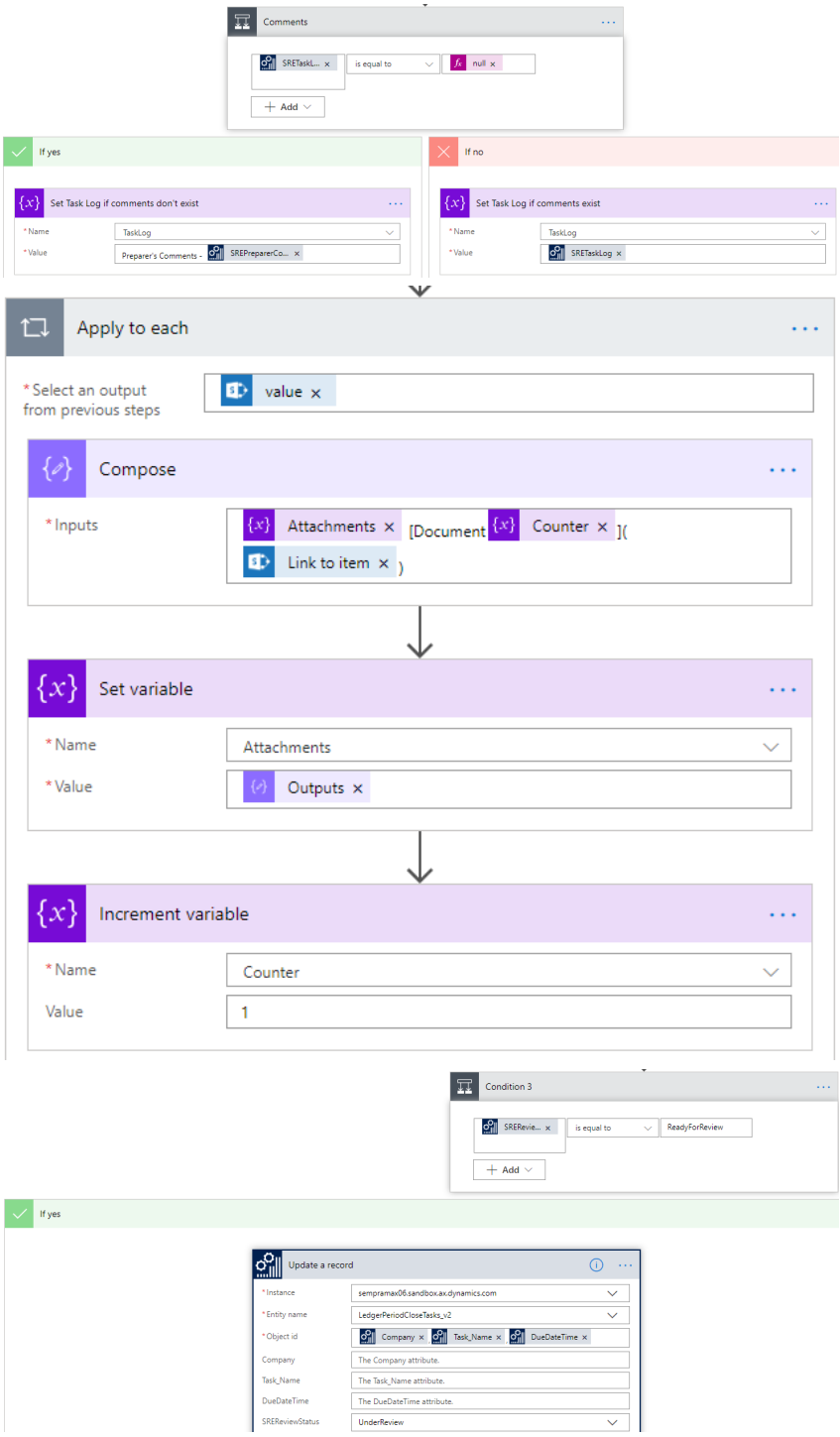
4. Retrieve approvers from the Microsoft Excel file, based on the criteria.

   Next, you must determine the list of approvers, so that you can send the approval request in the appropriate manner. This list is a custom Excel file in a SharePoint Online library. In this step, you query the Excel file to get the list of approvers. You also get the links to the attachments for each task, so that you can send the attachments to the approvers.



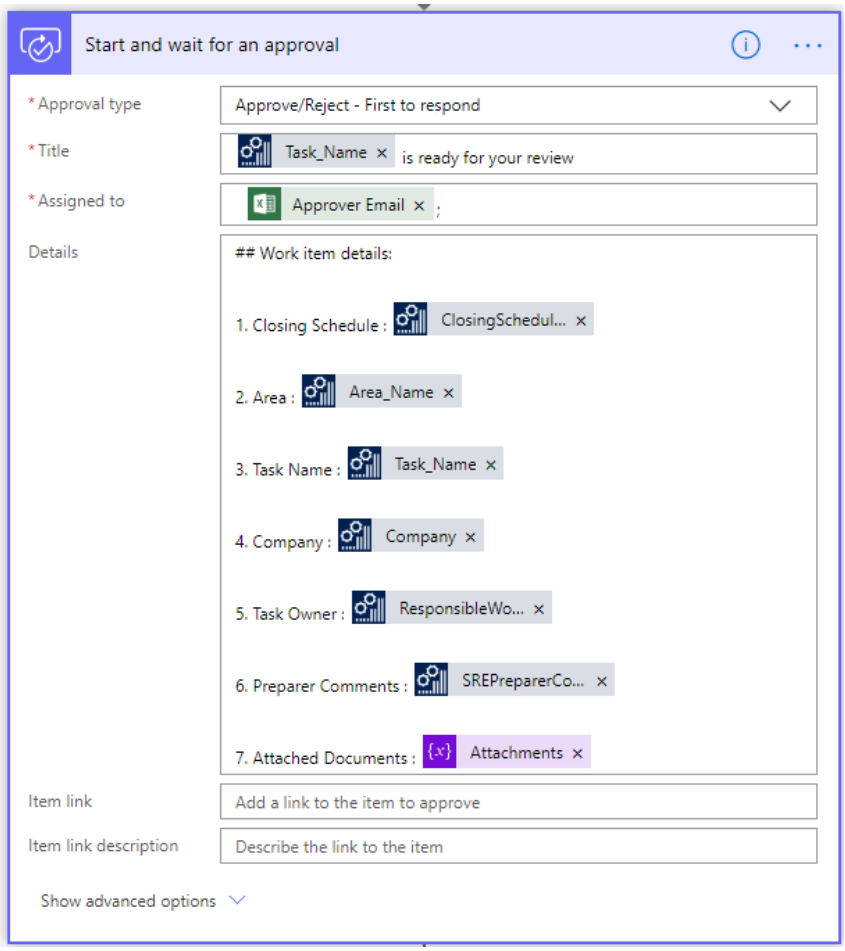
5. Prepare to send the request for approval.

   In this step, you prepare Power Automate to send the approval request by using all the information that was gathered and assembled in the previous step.



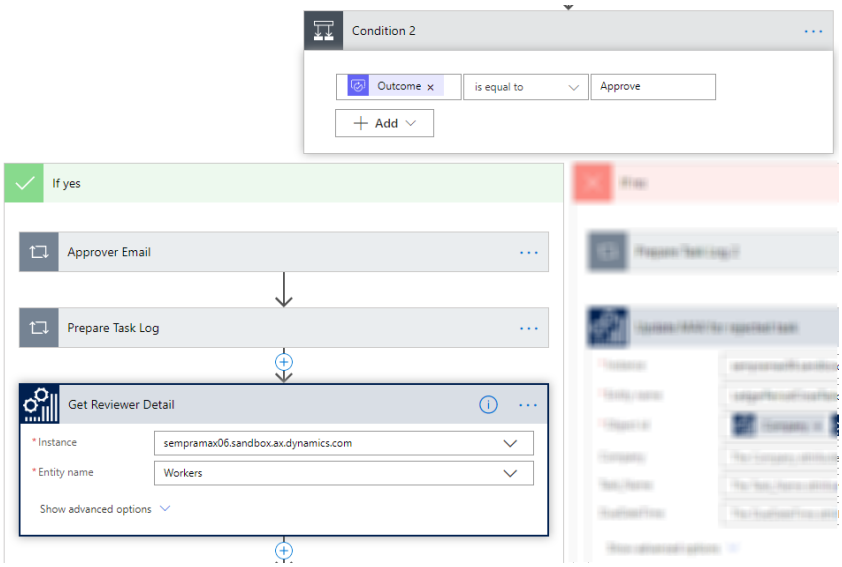
6. Start the approval process.

   In this step, the approval request is sent from Power Automate.



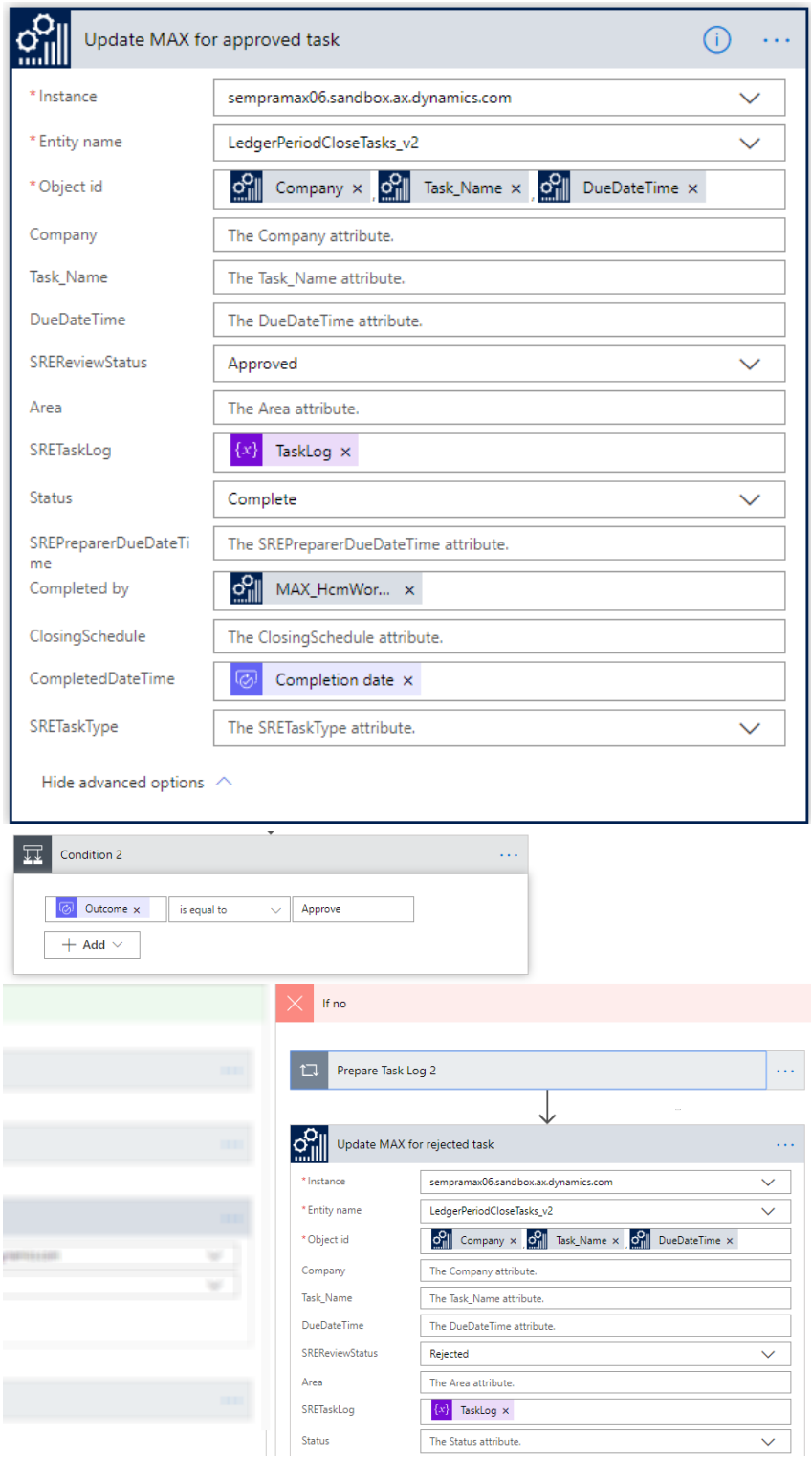
7. Process the approval action that is taken by approvers.

   After the approvers receive the approval request and take action, the Power Automate is notified, and additional processing is done.



8. Update the task with the approval outcome.

   Based on the outcome of the approval process, the task is updated with the result.



Conclusion

For the business requirements of the organization that is described in this topic, this solution involves minimal development and relies mostly on the **Financial period close** workspace, business events, SharePoint Online, and Power Automate to drive functionality. Development is restricted to the addition of fields to pages, the creation of custom data entities, and changes to page labels. Power Automate also provides greater flexibility in the approval process. Because the solution takes advantage of the various applications in the Microsoft 365 suite, internal users can use applications that they are already familiar with. Therefore, the amount of change management that is required is limited.

In conclusion, business events offer unique opportunities for extending functionality but also let you avoid extensive in-app customizations. Here are some things to consider before you start to use business events:

- Establish the security requirements of your solution. Business events honor role-based security. This behavior

can be beneficial in some use cases.

- Business events functionality continues to get enhanced. Be on the lookout for new capabilities.

Business events and Power Automate offer great opportunities for implementing low-code or no-code extensions. The important thing is that you identify opportunities where this framework can help, but that you also understand some of the limitations.

Continuous delivery home page

2/18/2021 • 2 minutes to read • Edit Online

These topics describe tools and guidelines related to continuous delivery of your solution.

FAQ and guidelines

- Development and continuous delivery FAQ

Build and test automation

- Deploy and use an environment that supports continuous build and test automation
- Build automation that uses Microsoft-hosted agents and Azure Pipelines
- Testing and validations
- Merge the build systems for Commerce and Finance
- SysTest filtering using class and method attributes
- Acceptance test library resources

Advanced topics in build automation

- Exclude test packages from build output
- Manage third-party models and runtime packages by using source control
- Update model versions in the automated build

Blogs

- Insider tips on development and customization (blog)

Servicing

- Download updates from Lifecycle Services (LCS)
- Apply updates to cloud environments
- Install metadata hotfixes in development environments
- Patch SQL Server Reporting Services (SSRS) in one-box environments
- Update the Visual Studio development tools

Development and continuous delivery FAQ

2/18/2021 • 9 minutes to read • Edit Online

This topic summarizes answers to questions that are frequently asked by ISVs and partners, especially regarding guidelines about development, testing, delivery, and lifecycle management.

Customization

Do I customize (overlayer) or use extensions?

Extensibility is the only customization framework in Finance, Supply Chain, and Commerce. Overlayering isn't supported.

Dynamics 365 Finance, Supply Chain, and Commerce are extensively customized by partners, value added resellers (VARs), and even some customers. The ability to customize the product is a strength that has historically been supported through overlayering of the application code. The move to the cloud, together with more agile servicing and frequent updates, requires a less intrusive customization model, so that updates are less likely to affect custom solutions. This new model is called *extensibility* and has replaced customization through overlayering.

For more information, see Extensibility home page and the Develop and customize home page.

How do I prevent my models from being customized by customers or other partners?

You can block customizations of your model as described in Turn off model customization and deprecate functionality, or you can distribute deployable packages to your customers instead of distributing model files. See the section titled "How do I distribute my application to customers" later in this topic.

How can I define the scope of my models? How many models or packages should I create?

Designing models and model elements is no different than designing other types of software libraries. You should apply SOLID (object-oriented design) design principles. In addition, we recommend the following tips that are specific to the platform:

- If there are components in your solution that you want to ship and service more frequent than the rest, they are good candidates to place in a separate model and package.
- It is common practice to start with two packages (each with one model) at the initial stage of an implementation, one foundation package that contains extensions to the Microsoft platform packages and one application package that contains extensions to the Microsoft application packages. More models can be introduced on an as-needed basis.
- Existing packages can be subdivided into smaller packages when necessary. If your implementation is already live using one of your packages, avoid renaming a package, to help simplify lifecycle management.

Continuous delivery

Do I need build environments?

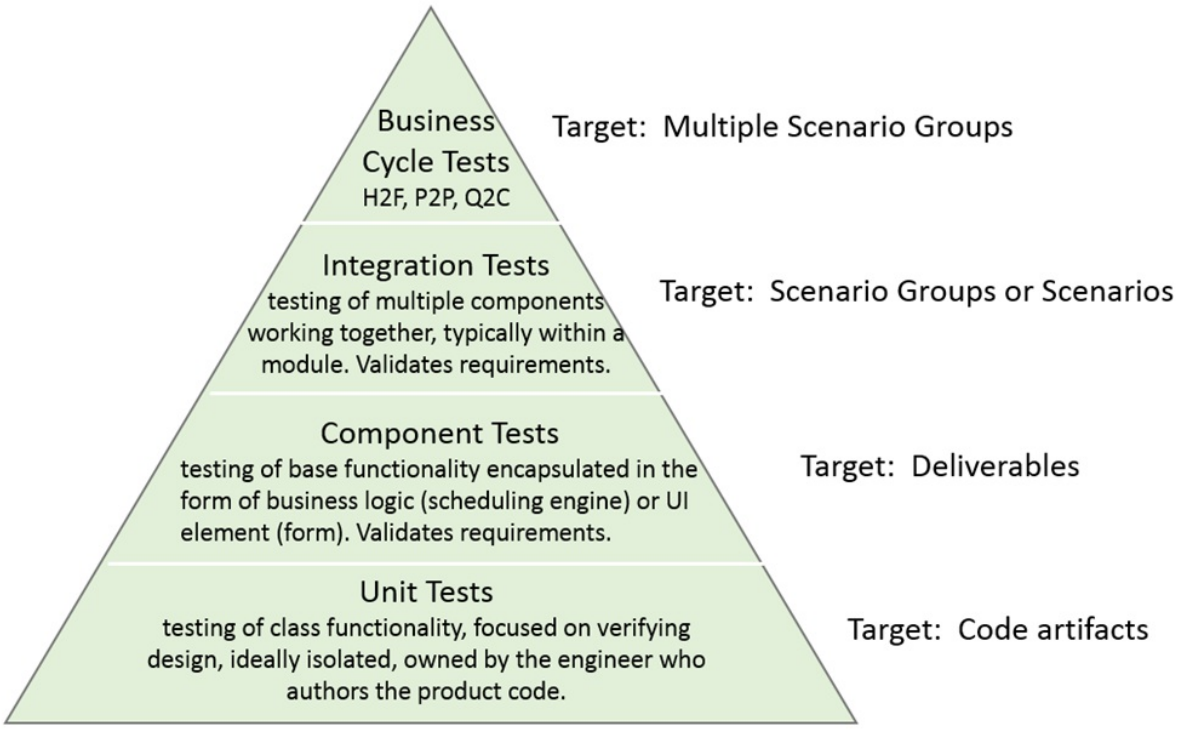
Yes, you should take advantage of the build and test automation tools provided in the build environments. You can deploy build environments from your Lifecycle Services (LCS) project. Creating daily builds and daily regression tests are key tools to enable the continuous delivery and maintain the quality of your application. Refer to Deploy and use an environment that supports continuous build and test automation for more details.

Do not use build environments for development activities. Do not keep a backup of your test database on these build VMs. Build VMs are designed to reset themselves to a known state with every build and whenever they are

updated with a Microsoft binary or platform updates from LCS. For example, if you apply a binary hotfix or platform update to a build VM, the VM prepares itself for the next build as part of the update. This will remove your customizations and also trigger a database synchronization.

What strategy do I use for test automation?

For test automation, concentrate on unit tests (use the SysTest framework) that are data independent or create their own data. Use a smaller number of functional scenario tests (based on Task Recorder) that rely on test data to execute. Scenario tests are more expensive to maintain. Unit tests can then be executed on any development environment easily and quickly. Review the Test Automation Pyramid blog article and refer to Automated testing guidance.



Some key concepts to keep in mind:

- Write tests that run independently and do not assume any kind of ordering.
- Task recorder tests should be limited to functional scenarios tests.
- Write scenario tests after scenarios are complete and after completing unit tests.
- Create test helper classes when possible, so others on your team can leverage that as well.
- SysTest framework supports role-based testing, leverage this feature.

How can I be more agile in my development?

Deliver incremental features every sprint (2 weeks, preferred) or cycle (1 month). Maintain shippable quality of your application at the end of each sprint. Use Azure DevOps for work item tracking and always prioritize bugs over new features. A large bug backlog will quickly become a burden on your efficient delivery of new features and on the quality of your application.

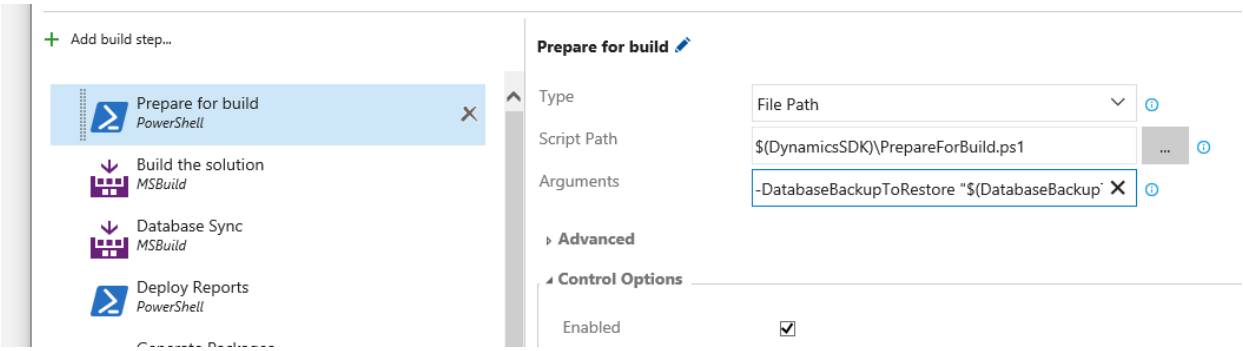
How do I manage test data?

Create and manage your test database as follows:

- Start on a clean environment.
- Create all base data as required. Base data will serve as the starting point for all the tests.
- Take a backup (.bak) of your AxDB database.
- Share this backup with developers.

On a build environment, copy this backup over to the I:\DynamicsBackupDatabases (on some environment it may be a different drive than i:). This database will be restored at the beginning of every build. This step is

executed as part of the first step of the build definition called **Prepare for build**.



How do I distribute my application to customers?

There are two artifacts you can use to distribute your application to your customers or partners: model files or deployable packages. Model files are design time artifacts that contain source code. Use model files if your customer is integrating your application with other third-party models or when you want to allow customization of your models. For more information, see Export and import models. Model files are the most common methods for ISVs to distribute solutions. Deployable packages are final applications. Use deployable packages with customers that will not be customizing or integrating your application with other third-party models. If you use deployable packages, your customer can only use or extend your application. They will not see or have access to your source code. To create a deployable package use the Visual Studio tools (**Dynamics 365** > **Deploy** > **Create Deployment package**) or use a build environment. Build environments generate a deployable package with every successful build.

Development topologies

Should I develop on premises or in the cloud?

There are two modes of development: Cloud VMs and on-premises VMs available via a downloadable VHD. Use a combination of on premise VMs and cloud VMs for development.

- On premise dev VMs are cost effective if you already have the hardware, IT infrastructure, and Windows server licenses to support it.
- Use cloud VMs to scale out when projects require additional resources for a limited period of time. It is more cost effective than planning for worst-case capacity on premise.
- Connect all VMs (on premise and cloud VMs) to Azure DevOps for version control.

Use cloud VMs for build, functional testing, and demos. If you are running on your own Microsoft Azure subscription, turn them off when not in use.

Should I use a customer's dev environment?

If you are a partner, use your own VMs for development of your own intellectual property (IP), this is code and configuration data packages that are reusable across different customer implementations. For customer-specific implementations, you can use the customer's dev VM. All customer subscriptions come with at least one development VM. Customers can pay for add-on dev VMs or run local dev VMs.

What are the benefits of MSDN subscriptions with respect to development?

The following is a summary of a Visual Studio (VS) with MSDN subscriptions benefits:

- Includes a Microsoft Azure subscription with a $50 monthly credit for Visual Studio Professional with MSDN and $150 for VS Enterprise with MSDN.
- Subscriptions come with lower dev/test rate, you will pay the Linux rates instead of the Windows rates.
- For more details, visit https://azure.microsoft.com/pricing/member-offers/msdn-benefits-details/

As a Microsoft partner, acquire Microsoft core competencies to earn free VS Enterprise with MSDN

subscriptions. For example, an application development competency for a gold partner will earn 25 free MSDN Enterprise licenses in addition to the 10 licenses that come with the core benefits. For more details, visit Monthly Azure credit for Visual Studio subscribers. These benefits make cloud development very economical, for example:

- D12v2 VM list price = $470/month (4 core, 28 Gigs)
- D12v2 VM price if running on MSDN Azure subscription or any other dev/test subscription = $276/month
- Turn off 12 hours per day: 276/2 => $138/month
- Monthly credit (VS Professional with MSDN) => 138 - 50 = **$88/month**
- Monthly credit (VS Enterprise with MSDN) => 138 – 150 = **Free**

Here is another example:

- D13v2 machine list price = $843/month (8 core, 56 Gigs)
- D13v2 machine price if running on MSDN Azure subscription = $551/month
- Turn off 12 hours per day: 551/2 = $275.5/month
- Monthly credit (VS Professional with MSDN): 275.5 - 50 = **$225.5/month**
- Monthly credit (VS Enterprise with MSDN) => 275.5 – 150 = **$125.5/month**

Add an average of $15 monthly for storage (non premium) per VM.

Can more than one developer develop concurrently on the same VM?

This is not supported. However, you can provision more than one developer account on the same VM, they just cannot develop concurrently. For details, see Create new users on development machines.

If you are a Microsoft partner developing code for more than one customer, we recommend having at least one development VM per customer. You will need one additional VM for every additional developer working on a customer project. Development VMs can be thought of as disposable assets as long as your source code is checked into version control (Azure DevOps) and you keep a backup of test databases.

Customer implementation LCS projects

How many sandbox environments do I need within an LCS customer implementation project?

A customer subscription comes with three environments by default: a dev or build environment, a tier-2 sandbox environment, and a production environment. You can use the tier-2 sandbox environment as a configuration and a UAT environment before the application goes live in production. After configuring the sandbox with the code and data that you need to go live (also known as your *gold configuration*), you can run your validation on the same environment. When your validation passes, restore your sandbox database to the point in time of its gold configuration. You can then deploy your code to production and copy the sandbox database to your production environment. You can also choose to have more than one sandbox environment that is tier-2 or higher, especially after your application is live. One sandbox can be used as a pre-production UAT environment, and the other sandboxes can be used for configuration, upgrade or other scenarios. You can purchase additional tier-2 or higher sandboxes.

The following servicing requests and tools are supported by LCS, which may help you decide whether one tier-2 sandbox is sufficient for your implementation.

1. Restore a sandbox database to a point in time.
2. Copy a sandbox database to a production environment (only allowed before the application is live in production).
3. Apply configuration data packages on a sandbox environment.
4. Apply configuration data packages on a production environment.
5. Refresh a sandbox database from production. Copy the production environment's database to a tier-2 sandbox environment. This is typical after the application is live and you want to debug an issue or validate

upcoming updates.

6. Apply updates (Hotfixes, customizations) to a sandbox environments for validation before applying them to a production environment.

For more information about planning an environment, see Environment planning.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Exclude test packages from build output

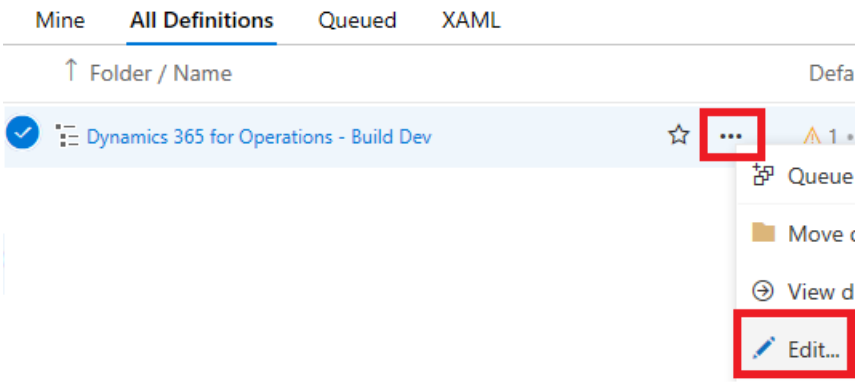
2/18/2021 • 2 minutes to read • Edit Online

In Platform update 4, the automated build process lets you prevent specific packages from being included in the deployable package in the build output. This capability can be important for customers that use automated testing. These customers might want to build and run their tests, but prevent them from being added to the deployable package that the build generates as output.

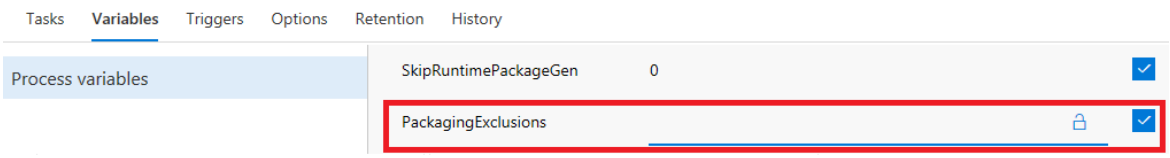
When customers that have an existing build definition from Platform update 3 or earlier upgrade, they won't see the build definition automatically updated. To use the new feature, these customers must make a few manual edits to the build definition (see below for details).

The new feature exposes a new optional parameter for the package creation step in the build process. Because this parameter is managed by a build variable, you can easily adjust it.

1. In Microsoft Azure DevOps, on the **Build & Release** page, under **Builds**, on the **All Definitions** tab, find your build definition. Click the ellipsis (...), and then click **Edit**.



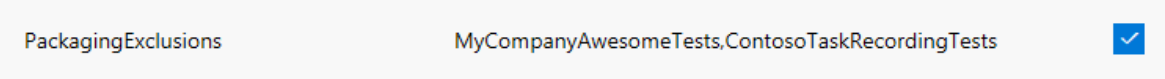
2. On the **Variables** tab, notice that the new build definition has a variable that is named **PackagingExclusions**.



3. In the **PackagingExclusions** variable, specify a comma-separated list of the names of packages that should not be packaged into the deployable package.

> **NOTE**
> The name of a package isn't necessarily the name of the model. Instead, the package name is typically the name of the folder where the model resides. Alternatively, you can copy and paste the package name from the descriptor file of one of the package's models. (In the XML, you can find the package name in the **ModelModule** field.)

   For example, you have one package that is named MyCompanysAwesomeTests and another package that is named ContosoTaskRecordingTests, and you want to exclude both these packages from the deployable package. In this case, the value for the **PackagingExclusions** variable will look like this.



After you complete this setup, the build process will still build the code and run any tests that the packages contain. However, the deployable package that the build creates won't include those packages.

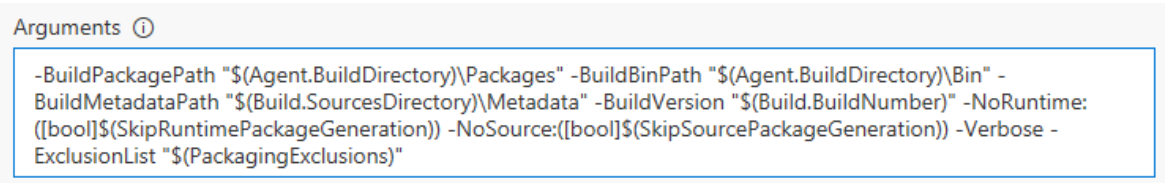
Update an existing build definition after upgrade to Platform update 4 or later

To use the new feature, you must manually update any existing build definitions that you deployed before Platform update 4.

> **NOTE**
> The feature can be added to a build definition only after you update the build virtual machine (VM) to Platform update 4 or later.

1. On the **Variables** tab, click **+ Add** at the bottom of the page.
2. In the **Name** column, enter **PackagingExclusions**. In the last column, select the **Settable at queue time** check box.
3. On the **Tasks** tab, find the **Generate Packages** task. Click to select it.
4. On the right side of the page, find the **Arguments** parameter. Click in the text box, and then press the End key or scroll to the end of the text box. The new build definition will have a new argument that passes the **PackagingExclusions** variable that you defined earlier. However, for an existing build definition, add a space and then the following text to the end of the parameter: **-ExclusionList "$(PackagingExclusions)"**

   The **Arguments** text box should now look like this.

5. Click **Save**.

You can now use the new feature as described.

> **NOTE**
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Manage third-party models and runtime packages by using source control

2/18/2021 • 4 minutes to read • Edit Online

Customers that work with solutions from third parties might receive different solution artifacts to use in their solution. Typically, these artifacts are distributed as code (in the form of models) or binaries (in the form of deployable packages). In some cases, third parties might provide some parts of their solution as code and other parts as a binary.

This topic outlines a recommended strategy for managing, distributing, and deploying these third-party solutions.

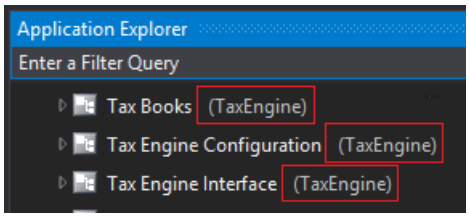
Models from third parties

Any source code that is received from third parties must be compiled into a binary and included in a deployable package. Models should be installed on a development virtual machine (VM) and added to source control. From there, the build VM can pick up the source code, build it, and include it in a deployable package. Other developers can just synchronize the model from Microsoft Azure DevOps to their development VMs. They don't have to manually install it.

For information about how to install a model on a development VM, see Export and import models.

After you install the model, follow these steps to add the new model to source control.

1. In Microsoft Visual Studio, on the **Dynamics 365** menu, click **Model Management** > **Refresh Models**.
2. Open Application Explorer by clicking **View** > **Application Explorer**.
3. Right-click the **AOT** root node, and then click **Model view**.
4. In the list of models, find the new model that you installed. Make a note of the name of the package that contains the model. The package name appears in parentheses after the model name. For example, in the following illustration, the **Tax Books**, **Tax Engine Configuration**, and **Tax Engine Interface** models all belong to the package that is named **TaxEngine**.



5. Open Source Control Explorer by clicking **View** > **Other Windows** > **Source Control Explorer**.
6. Navigate to the metadata folder that is mapped on this development VM, such as **MyProject/Trunk/Main/Metadata**.
7. In the metadata folder, find the folder for the package that contains the new model. Right-click the package folder, and then click **Add Items to Folder**.
8. In the **Add to Source Control** dialog box, select the **Descriptor** folder and the folder that has the name of the model. Some models may also contain referenced DLLs in the **bin** folder. If these exist you'll need to also include the approriate DLL files from the **bin** folder. Once all files have been selected, click **Next**.

9. Review the items that will be added, and then, when you're ready, click **Finish**.
10. Open the **Pending Changes** window from the **Team Explorer** pane or by clicking **View** > **Other Windows** > **Pending Changes**.
11. Review the changes, enter a check-in comment, and then click **Check In**.

Deployable packages from third parties

Deployable packages from third parties can be manually installed on a development VM, and the installed artifacts can then be added to source control. Then, by synchronizing their local workspace, other developers can receive the runtime package on their VMs without having to install the deployable package. The build process on the build VM will help guarantee that the runtime packages for any extensions or other dependencies are available on the build VM. In Platform update 6 and later, by default, these runtime packages will be included in the final deployable package that is created from the build VM. For more information, see the Deploying third-party code section later in this topic.

For information about how to install a deployable package on a development VM, see Install deployable packages from the command line.

> **NOTE**
> Don't install a software deployable package directly on the build VM. Use source control as described in this topic. Only binary updates should be installed on build VMs.

After you install the deployable package on a development VM, follow these steps to add the runtime package to source control.

1. Open Source Control Explorer by clicking **View** > **Other Windows** > **Source Control Explorer**.
2. Navigate to the metadata folder that is mapped on this development VM, such as **MyProject/Trunk/Main/Metadata**.
3. Right-click the **Metadata** folder, and then click **Add Items to Folder**.
4. In the **Add to Source Control** dialog box, double-click the folder that has the package name that you want to add to source control.
5. Select all the folders except **XppMetadata** and **Descriptor**, if they exist, and then click **Next**.
6. On the next page, on the **Excluded items** tab, select all files by clicking one of the files and then pressing Ctrl+A. At the bottom of the selection window, click **Include item(s)**. When you're ready, click **Finish**.
7. Open the **Pending Changes** window from the **Team Explorer** pane or by clicking **View** > **Other Windows** > **Pending Changes**.
8. Review the changes, enter a check-in comment, and then click **Check In**.

Deploying third-party code

Because the models and runtime packages are in source control, other developers who use other development environments can just synchronize the models and packages to their workspace by using the **Get latest** feature of source control.

As of Platform update 4, the automated build process will also pick up the runtime packages. Therefore, dependencies in packages that are built will be resolved correctly. This feature is also available for Platform update 3 and Platform update 2 through a hotfix.

In Platform update 6, the build process will include this runtime package in the final deployable package. This allows customers to take the deployable package from the build and have one package to deploy to their environments. The one package includes both custom solutions and all the third party solutions.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

All-in-one deployable packages

2/18/2021 • 3 minutes to read • Edit Online

Customers can update the software in their environments by applying software deployable packages. These packages can originate from the customers themselves in the form of customizations. They can also be provided by partners and independent software vendors (ISVs). Microsoft recommends that customers combine all these various packages into a single package before they apply them to an environment. For customers who have self-service environments, this approach is a hard requirement.

This topic outlines the best practices for creating and managing an all-in-one deployable package.

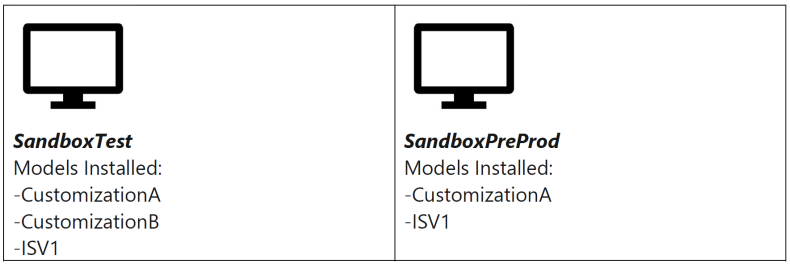
IMPORTANT

- The enforcement of all-in-one packages will be done in phases. Request to extend the support for deployable packages that are **not** all-in-one deployable packages will end October 31, 2020. The extension approval will be subject to valid justification.
- If a payment connector is currently deployed in your environment, you will have to create a payment connector package and include it in the all-in-one deployable package.
- If you currently use Microsoft Dynamics 365 Commerce functionality for the retail point of sale, you will also have to synchronize self-service installers.

What is an all-in-one deployable package?

An all-in-one deployable is a software deployable package that contains all the models and binaries that you currently have in an environment. Think of it as a single package that represents all the non-Microsoft software in an environment.

For example, you have two environments: **SandboxTest** and **SandboxPreProd**.



If your software deployable package contains CustomizationA, CustomizationB, and ISV1, it's a fully deployable package for the SandboxTest environment. This is because it exactly matches the model list. It's also a fully deployable package for the SandboxPreProd environment because it has all the models that are installed, plus CustomizationB.

However, if your software deployable package contains only CustomizationA, it isn't fully deployable for either environment. The package is missing some of the models that are already installed.

How do I create an all-in-one deployable package?

There are two primary methods for creating an all-in-one deployable package:

- If you're using the continuous integration/continuous deployment model, you're already creating all-in-one deployable packages.
- If you don't have a build environment, you can create a package in Microsoft Visual Studio. For more

information, see Create deployable packages of models.

What if my ISV packages don't contain source code?

ISVs can choose whether to share their source code with you. If they don't share it, they will provide a binary-only package. This package can easily be managed into an all-in-one deployable package. For instructions, see Manage third-party models and runtime packages by using source control.

How can I deploy ISV licenses?

ISVs can send license deployable packages to provide or update a license. However, for self-service environments licenses should also be included in an all-in-one deployable package. You can add a task to your build or release pipelines to add any licenses you have to a deployable package. For more information, see Add license files to a deployable package in Azure Pipelines.

Why are these packages important?

The best practice of using fully deployable packages helps reduce the complexity and number of packages that are applied to a given environment. In some circumstances, installation of different packages can change the behavior of your environment. For example, if you install ModelA and then ModelB instead of ModelB and then ModelA.

In addition, this approach is a hard requirement for self-service environments. This is because those environments use containerization technology and build a brand-new environment every time that you apply a package. If you apply ModelA today and then apply only ModelB tomorrow, you will effectively uninstall ModelA.

Update model versions in the automated build

2/18/2021 • 4 minutes to read • Edit Online

In Platform update 6, a new task in the automated build definition updates the models in the source package and deployable package of the build output with the version of the build that produced them.

Build definitions that were created before Platform update 6 must be manually updated to include this task. See the Updating an existing build definition section later in this topic.

Version numbers

Even though models are compiled into one package, the metadata information of all models is retained inside the binary package. This information can be reviewed from Microsoft Dynamics Lifecycle Services (LCS) or from the client.

In LCS, follow these steps to find the version numbers of models that are installed in an environment.

1. On the **Full details** page for the environment, under **Environment Version Information**, click the **View detailed version information** link.
2. On the **Installed updates** page, in the **Machine name** field, select an Application Object Server (AOS) computer.
3. In the table list, find the **Publisher name** field of the model, and expand the list by clicking the arrow icon. A full list of all models from that publisher is shown. The version number is shown in the **Version** column.

In the client, follow these steps to find the version numbers of models that are installed in an environment.

1. Open the URL for the environment, and sign in.
2. After the dashboard is loaded, click the gear symbol at the upper right of the page, and then click **About**.
3. In the dialog box that appears, expand **Loaded Packages and their Models**. Find the package where the model resides, and expand the list by clicking the arrow icon. The list of models for the package is shown, together with the version number.

All version numbers are in .NET assembly format. They consist of four numbers that are separated by a dot (.), such as **1.2.3.4**.

The purpose of model versioning

As code is updated, the build is used to produce new packages that can be deployed to environments. Microsoft Azure DevOps tracks the changes that have been included in each build since the previous build. When the version number of the build is included in the models that are produced, it provides end-to-end traceability of the code changes that are available in a specific environment. You can find the build number and then review the changes that are included in that build in Azure DevOps. For customers and partners that use builds on different branches, or that use different build definitions for nightly builds, gated check-in builds, or deployment builds, each build can have a different versioning scheme. This approach helps differentiate the model metadata in the deployable packages and tie them back to their originating build definition.

Setting up versioning

For build definitions that are created by Platform update 6 or newer deployments, the task to include build version in models is automatically added and active. The default build number of a new build definition in Azure DevOps consists of the year, month, and day, and the incremental number of the build for that day. For more

information about build numbers in Azure DevOps, and the options that are available, see Build definition options on the Microsoft Visual Studio docs site.

The automated build will apply the build version number to the models that are built.

Preventing models from being updated

By default, the build task assigns versions only to models that are in layers above the ISP layer. Therefore, customers can consume code models from third-party vendors without overwriting the version numbers that are supplied in their models. However, you can also prevent other models from having their version numbers overwritten during the build, regardless of layer. When you edit the build definition, on the **Variables** tab, in the **ModelVersionExclusions** variable, supply a comma-separated list of model names to exclude.

Updating models in lower layers

For third parties that develop solutions in the ISV or ISP layer, a manual change must be made to the build definition to automatically set model versions in those layers.

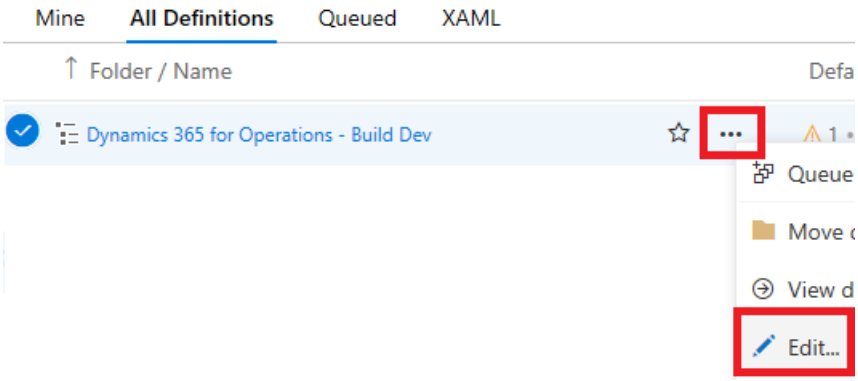
1. Edit the build definition. On the **Tasks** tab, click the **Set Model Versions** task.
2. In the **Arguments** field, add the following option at the end of the existing list of arguments: **-UpdateLayersAbove 7**

Updating an existing build definition

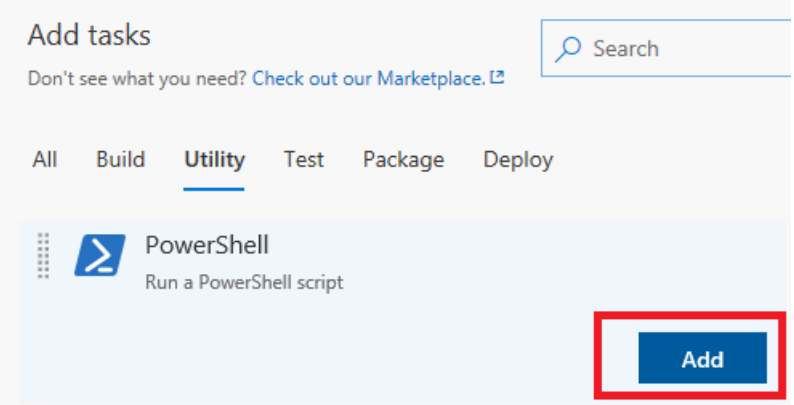
For build definitions that were created before Platform update 6, a new task must be manually added to the build definition.

> **NOTE**
> This feature can be added to a build definition only after the build virtual machine (VM) has been updated to Platform update 6 or later.

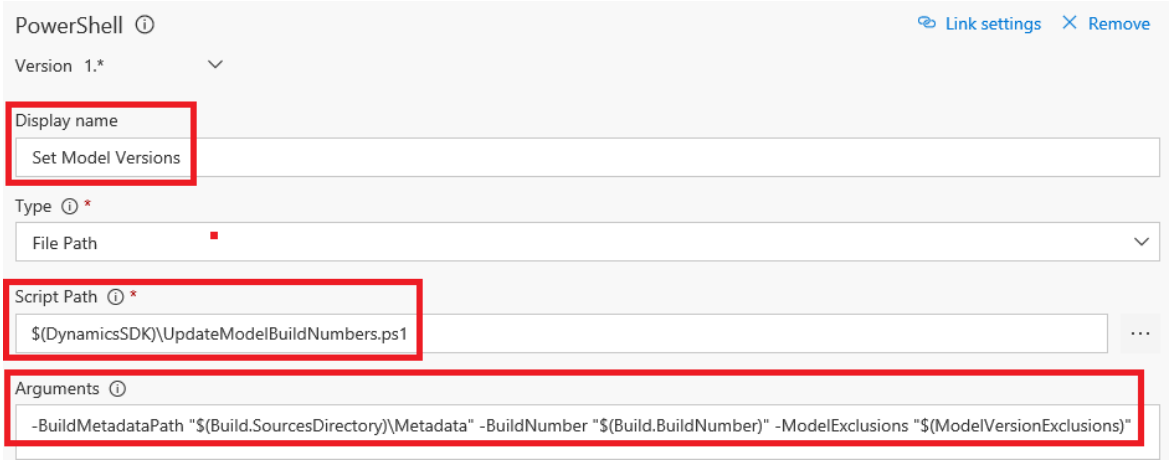
1. In Azure DevOps, on the **Build & Release** page, under **Builds**, on the **All Definitions** tab, find your build definition.
2. Click the ellipsis (...), and then click **Edit**.



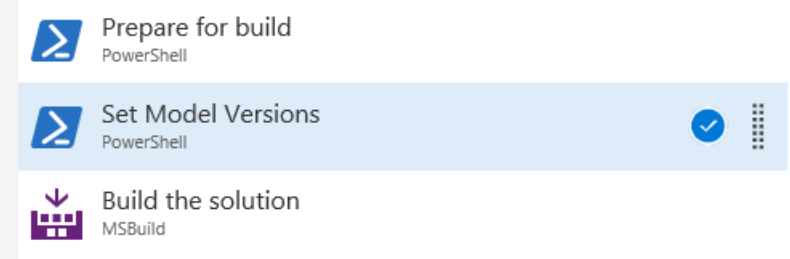
3. On the **Tasks** tab, click **+ Add Task** at the bottom of the page.
4. In the **Add tasks** pane on the right side of the page, on the **Utility** tab, scroll down to find the **PowerShell** task.
5. Hover the mouse pointer over the task, and click the **Add** button that appears.



6. In the list of tasks on the left side of the page, click to select the **PowerShell Script** task that is added.
7. On the right side of the page, change the **Display name**, **Script Path**, and **Arguments** properties to reflect the required settings.



8. In the list of tasks on the left side of the page, drag the **Set Model Versions** task so that it's between the **Prepare for build** and **Build the solution** tasks.



9. On the **Variables** tab, click **+ Add** at the bottom of the list of variables. In the first column, for the **Name** variable, enter **ModelVersionExclusions**.
10. Click **Save** to save the new task.

Choose a data integration strategy

2/18/2021 • 13 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

This topic is intended to help architects and developers make sound design decisions when they implement integration scenarios.

The topic describes integration patterns, integration scenarios, and integration solutions and best practices. However, it doesn't include technical details about how to use or set up every integration pattern. It also doesn't include sample integration code.

NOTE

When providing guidance and discussing scenarios for choosing a pattern, data volume numbers are mentioned. These numbers must be used only to gauge the pattern and must not be considered as hard system limits. The absolute numbers will vary in real production environments due to various factors, configurations are only one aspect of this scenario.

The following table lists the integration patterns that are available.

PATTERN	DOCUMENTATION
Power Platform integration	Microsoft Power Platform integration with Finance and Operations apps
Dual-write	Dual-write overview
Classic data integration	Classic data integration overview
OData	Open Data Protocol (OData)
Batch data API	Recurring integrations Data management package REST API
Custom service	Custom service development
Consume external web services	Consume external web services
Excel integration	Office integration overview

NOTE

For on premise deployments, the only supported API is the Data management package REST API. This is currently available on 7.2, platform update 12 build 7.0.4709.41184.

Power platform integration

Finance and Operations is a virtual data source in Dataverse, and enables full create, read, update, delete (CRUD) operations from Dataverse and Microsoft Power Platform. By definition, the data for virtual entities doesn't reside in Dataverse. Instead, it continues to reside in Finance and Operations. To enable CRUD operations on Finance and Operations entities from Dataverse, entities must be made available as virtual entities in Dataverse. This allows CRUD operations to be performed, from Dataverse and Microsoft Power Platform, on data that resides in Finance and Operations apps. For detailed information, see Microsoft Power Platform integration.

Dual-write vs. classic data integration patterns vs. virtual entities

Dual-write provides synchronous, bi-directional, near-real time experience between model-driven applications in Dynamics 365 and Finance and Operations applications. Data synchronization happens with little or no intervention and is triggered by create, update and delete actions on an entity. Dual-write is suitable for interactive business scenarios that span across Dynamics 365 applications.

Classic data integration provides asynchronous and uni-directional data synchronization experience between customer engagement apps and Finance and Operations apps. It's an IT-administrator led experience and you must schedule the data sync jobs to run on a specific cadence. Classic data integration is suitable for business scenarios that involves bulk ingress/egress of data across Dynamics 365 applications.

PATTERN	TIMING	BATCH	TECHNOLOGY	FINANCE AND OPERATIONS APP	MODEL-DRIVEN APPS IN DYNAMICS 365
Dual-write	Synchronous Bi-directional	No	OData	Finance Supply Chain Commerce Service Industry CoreHR	Sales Marketing Customer Service Field Service Project Service Automation Talent
Classic data integration	Asynchronous, uni-directional	Yes	DIXF	Finance Supply Chain Commerce Service Industry CoreHR	Sales Marketing Customer Service Field Service Project Service Automation Talent

Virtual entities provide a mechanism to use Microsoft Power Platform with Finance and Operations without having to physically copy data to Dataverse. This guidance must be used to determine if the requirements will need dual-write or data integrator or virtual entities. Virtual entities and dual-write/data integrator are complementary technologies such that, they can be used together if required.

Synchronous vs. asynchronous integration patterns

Processing can be either synchronous or asynchronous. Often, the type of processing that you must use

determines the integration pattern that you choose.

A *synchronous* pattern is a blocking request and response pattern, where the caller is blocked until the callee has finished running and gives a response. An *asynchronous* pattern is a non-blocking pattern, where the caller submits the request and then continues without waiting for a response.

The following table lists the inbound integration patterns that are available.

PATTERN	TIMING	BATCH
OData	Synchronous	No
Batch data API	Asynchronous	Yes

Before you compare synchronous and asynchronous patterns, you should be aware that all the REST and SOAP integration application programming interfaces (APIs) can be invoked either synchronously or asynchronously.

The following examples illustrate this point. You can't assume that the caller will be blocked when the Open Data Protocol (OData) is used for integration. The caller might not be blocked, depending on how a call is made.

PATTERN	SYNCHRONOUS PROGRAMMING PARADIGM	ASYNCHRONOUS PROGRAMMING PARADIGM
OData	DbResourceContextSaveChanges	DbResourceContextSaveChangesAsync
Custom service	httpRequestGetResponse	httpRequestBeginGetResponse
SOAP	UserSessionServiceGetUserSessionInfo	UserSessionServiceGetUserSessionInfo Async
Batch data API	ImportFromPackage	BeginInvoke

Both OData and custom services are synchronous integration patterns, because when these APIs are called, business logic is immediately run. Here are some examples:

- If OData is used to insert product records, the records are immediately inserted as part of the OData call.
- If a custom service is used to look up on-hand inventory, business logic is immediately run as part of the JSON/SOAP call, and an inventory sum number is immediately returned.

Batch data APIs are considered asynchronous integration patterns, because when these APIs are called, data is imported or exported in batch mode. For example, calls to the ImportFromPackage API can be synchronous. However, the API schedules a batch job to import only a specific data package. The scheduling job is quickly returned, and the work is done later in a batch job. Therefore, batch data APIs are categorized as asynchronous.

Batch data APIs are designed to handle large-volume data imports and exports. It's difficult to define what exactly qualifies as a large volume. The answer depends on the entity, and on the amount of business logic that is run during import or export. However, here is a rule of thumb: If the volume is more than a few hundred thousand records, you should use the batch data API for integrations.

In general, when you're trying to choose an integration pattern, we recommend that you consider the following questions:

- Is there a business requirement that the integration should be in real time?
- What is the requirement for the peak data volume?
- What is the frequency?

Error handling

When you use a synchronous pattern, success or failure responses are returned to the caller. For example, when an OData call is used to insert sales orders, if a sales order line has a bad reference to a product that doesn't exist, the response that the caller receives contains an error message. The caller is responsible for handling any errors in the response.

When you use an asynchronous pattern, the caller receives an immediate response that indicates whether the scheduling call was successful. The caller is responsible for handling any errors in the response. After scheduling is done, the status of the data import or export isn't pushed to the caller. The caller must poll for the result of the corresponding import or export process, and must handle any errors accordingly.

Typical scenarios and patterns that use dual-write

Here are some typical scenarios that use dual-write.

Enable customer service representative to facilitate change of address for Finance and Operations customers

A customer relocates and wishes to change their billing and shipping address information. This customer contacts the customer support representative and requests a change of address. The customer support representative takes the call and changes the billing and shipping address information of the customer.

DECISION	INFORMATION
Is real-time data required?	Yes
Peak data volume	
Frequency	Ad hoc

Recommended solution

This scenario of near-real time data synchronization is best implemented by dual-write.

- The customer's information is sourced in a Finance and Operations app.
- A customer calls customer support and asks to change their billing and shipping address information.
- A customer support representative retrieves the customer’s record in Dynamics 365 Customer Service.
- The customer support representative updates the billing and shipping address and saves the data.
- The new billing and shipping address syncs back to the Finance and Operations app in real-time.

Sales representatives can change customer credit limits without logging into a Finance and Operations app

A customer has a credit limit of $2,000 and wants to increase it to $5,000. This customer calls the customer support and requests the increase. The ticket is assigned to the sales department. The head of sales reviews the request, checks the customer's payment history, and determines that the customer is eligible for an increased credit limit. The head of sales approves the request and responds to the ticket. The customer receives an email informing the approval of $5,000 credit limit.

DECISION	INFORMATION
Is real-time data required?	Yes
Peak data volume	
Frequency	Ad hoc

Recommended solution

This scenario is best implemented by dual-write.

- A customer calls customer support and wants to increase their credit limit from $2,000 to $5,000.
- A customer support representative creates a ticket in Dynamics 365 Customer Service.
- This ticket is assigned to the sales unit.
- A sales representative from the sales unit reviews and approves the request.
- This result is the increase of credit limit of the customer to $5,000 in Dynamics 365 Sales.
- The credit limit in the Finance and operations app is updated to $5,000.
- The sales representative responds to the ticket and resolves it.
- The customer receives an email about the increased credit limit.

Typical scenarios and patterns that use OData integrations

Here are some typical scenarios that use OData integrations.

> **NOTE**
> Use of OData for Power BI reports is discouraged. Using entity store for such scenarios is encouraged.

Create and update product information

A manufacturer defines and configures its product by using a third-party application that is hosted on-premises. This manufacturer wants to move its production information from the on-premises application to Finance and Operations. When a product is defined, or when it's changed in the on-premises application, the user should see the same change, in real time.

DECISION	INFORMATION
Is real-time data required?	Yes
Peak data volume	1,000 records per hour*
Frequency	Ad hoc

* Occasionally, many new or modified production configurations will occur in a short time.

Recommended solution

This scenario is best implemented by using the OData service endpoints to create and update product information in Finance and Operations.

In Finance and Operations:

- Determine all the entities that are required for the integration.
- Make sure that the OData service endpoints are available for the same set of entities.

In the third-party application:

- When product information is created or modified in the third-party application, an OData call is made to Finance and Operations to make the same change.

Read the status of customer orders

A company has a self-hosted customer portal where customers can check the status of their orders. Order status information is maintained in the application.

DECISION	INFORMATION
Is real-time data required?	Yes
Peak data volume	5,000 records per hour
Frequency	Ad hoc

Recommended solution

This scenario is best implemented by using the OData service endpoints to read order status information.

In Finance and Operations:

- Determine the entity that is required in order to read order status information.
- Make sure that the OData service endpoint is available for the entity.

On the customer portal site:

- When a customer checks the status of an order, make a real-time OData call to Finance and Operations to read the corresponding order and retrieve its status.

Approve BOMs

A company uses a product lifecycle management (PLM) system that is hosted on-premises. The PLM system has a workflow that sends the finished bill of materials (BOM) information to the application for approval.

DECISION	INFORMATION
Is real-time data required?	Yes
Peak data volume	1,000 records per hour
Frequency	Ad hoc

Recommended solution

This scenario can be implemented by using an OData action.

In Finance and Operations:

- Determine the entity that is required for the integration.
- Make sure that the OData service endpoints are available for the entity.
- On the entity, create an action to run the required business logic.

In the PLM solution:

- Make the PLM system invoke the OData action to approve the BOM.

> **NOTE**
> You can find an example of this type of OData action in **BOMBillOfMaterialsHeaderEntity::approve**.

Typical scenarios and patterns that use a custom service

Here are some typical scenarios that use a custom service.

Look up on-hand inventory

An energy company has field workers who schedule installation jobs for heaters. This company uses the application for the back office and third-party software as a service (SaaS) to schedule appointments. When field workers schedule appointments, they must look up inventory availability to make sure that installation parts are available for the job.

DECISION	INFORMATION
Is real-time data required?	Yes
Peak data volume	1,000 records per hour
Frequency	Ad hoc

Recommended solution

This scenario can be implemented by using a custom service.

In Finance and Operations:

- Create a custom service to calculate the physical on-hand inventory for a given item.

In the scheduling application:

- Make a real-time call to a custom service endpoint, through either SOAP or REST, to retrieve inventory information for the selected item.

> **NOTE**
> You can find an example of this type of custom service in the Retail Real Time Services implementation: **RetailTransactionServiceInventory::inventoryLookup**.

You can also use the inventorySiteOnHand entity to achieve the same result. Sometimes, you can use multiple methods to expose the same data and business logic, and all the methods are equally valid and effective. In this case, choose the method that works best for a given scenario and that a developer is most comfortable with.

Typical scenarios and patterns that use batch data integrations

Here are some typical scenarios that use batch data APIs.

Import large volumes of sales orders

A company receives a large volume of sales orders from a front-end system that runs on-premises. These orders must periodically be sent to the application for processing and management.

DECISION	INFORMATION
Is real-time data required?	No
Peak data volume	200,000 records per hour
Frequency	One time every five minutes

Recommended solution

This scenario is best implemented by using batch data APIs.

In Finance and Operations:

- Determine all the entities that are required for the integration.

- Make sure that data management is enabled for the entities.

In the on-premises system:

- Use the REST batch data API to import files.

Export large volumes of purchase orders

A company generates a large volume of purchase orders in Finance and Operations and uses an on-premises inventory management system to receive products. Purchase orders must be moved from Finance and Operations to the on-premises inventory system.

DECISION	INFORMATION
Is real-time data required?	No
Peak data volume	300,000 records per hour
Frequency	One time per hour

Recommended solution

This scenario is best implemented by using batch data APIs.

In Finance and Operations:

- Determine all the entities that are required for the integration.
- Make sure that data management is enabled for the entities.
- If incremental push is required, make sure that change tracking can be enabled on the entities.

In the on-premises inventory system:

- Use the REST batch data API to export the file from Finance and Operations and import it into the inventory system.

Typical scenarios and patterns that call external web services

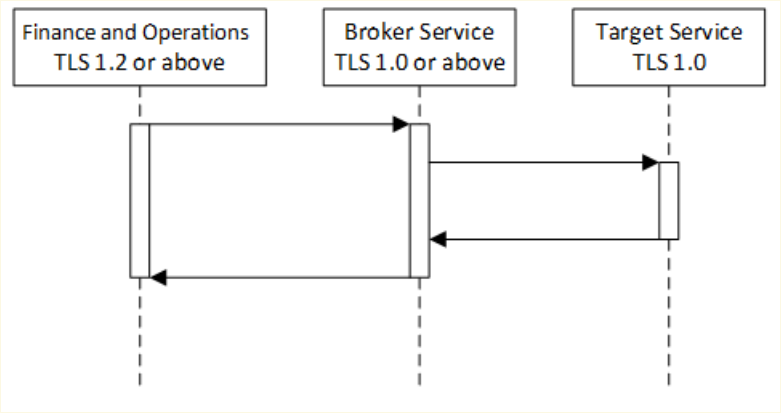
It's typical that the application calls out to an external web service that is hosted either on-premises or by another SaaS provider. In this case, the application acts as the integration client. When you write an integration client, you should follow the same set of best practices and guidelines that you follow when you write an integration client for any other application. For a simple example, see Consume external web services.

IMPORTANT

Because of security requirements, production and sandbox environments support only secured communication that uses Transport Layer Security (TLS) 1.2 or later. In other words, the target web service endpoint that the application calls out to must support TLS 1.2 or later. If the target service endpoint doesn't meet this requirement, calls fail. The exception error message resembles the following message:

Unable to read data from the transport connection: An existing connection was forcibly closed by the remote host.

If you can't modify the target service so that it uses TLS 1.2 or later, you can work around this issue by introducing a broker service and making a two-hop call, as shown in the following illustration.



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Priority-based throttling

2/18/2021 • 3 minutes to read • Edit Online

NOTE

The functionality noted in this topic is available as part of a preview release. The content and the functionality are subject to change. To test this capability, configure integration priorities on the **Throttling priority mapping** page.

Priority-based throttling prevents the over-utilization of resources to preserve the system's responsiveness and ensure consistent availability and performance for environments running Dynamics 365 Finance and Operations apps.

- Integrations can be prioritized based on business-critical needs. Throttling honors these priorities.
- For OData and custom service requests, a429 error "Too many requests", will occur.
- You can query throttling events on the **Lifecycle Services Monitoring** page.

Priority-based throttling provides the ability to set priorities for OData and custom service-based integrations, depending on the business-critical need of integrations.

The **Integration priority** page is used to assign priorities for integrations so that priorities can be honored when requests are throttled.

Setting appropriate priorities ensures that low-priority integrations will be throttled before high-priority integrations, based on the integration. For more information about how to set up integration, see Enable connectivity with external services.

There are two kinds of applications are supported in Microsoft Azure Active Directory (Azure AD):

- User based - This flow uses a username and password for authentication and authorization.
- Azure AD app based - A confidential client is an application that can keep a client password confidential. The authorization server assigned this client password to the client application.

For more information, see Authentication.

Configure priorities for integrations

After you have registered your service in Azure AD and in your Finance and Operations apps, you can set up priorities for integrations.

NOTE

You must be assigned the System administrator or Integration priority manager role to complete the set up.

1. In Finance and Operations apps, go to**System administration** > **Setup** > **Throttling priority mapping**.
2. Select**New**.
3. In the**Authentication type**field, select **User** or **Azure AD application** based on your integration scenario.
4. If **Azure AD application type** is selected, in the**Client ID**field select the application that you registered in the Azure Active Directory application.
5. If **User type** is selected, in the**User ID**field select an appropriate service account user ID.
6. Assign the appropriate priority and then select **Save**.

Retry operations

When a request is throttled, the system provides a value indicating the duration before any new requests from the user can be processed. When a request is throttled and a 429 error occurs, the response header will include a**Retry-After** interval, which can be used to retry the request after a specific number of seconds. The following example shows this operation.

```
if (!response.IsSuccessStatusCode)
        {
            if ((int)response.StatusCode == 429)
            {
                int seconds = 30;
                //Try to use the Retry-After header value if it is returned.
                if (response.Headers.Contains("Retry-After"))
                {
                    seconds = int.Parse(response.Headers.GetValues("Retry-After").FirstOrDefault());
                }
                Thread.Sleep(TimeSpan.FromSeconds(seconds));



                // Retry sending the request.
            }
        }
```

Monitoring

To have a successful onboarding experience with the throttling capability, you must also be able to monitor your Odata and custom service integration patterns. Microsoft Dynamics Lifecycle Services (LCS), which is the administration center, contains a collection of monitoring and diagnostics tools that can help ensure that you have an accurate view of the environments you manage. For more details, see Monitoring and diagnostics tools in Lifecycle Services (LCS).

You can use a set of predefined queries to get raw logs for an issue. You can then export the logs for a more advanced analysis. The following types of queries are available:

- All throttling events
- Requests throttled

Access the Monitoring and diagnostics portal

1. In LCS, open the appropriate project.
2. In the **Environments** section, select the environment to view, and then select **Full details**.
3. On the **Environment details** page, select **Environment monitoring** to open the Monitoring and diagnostics portal.
4. On the **Environment monitoring** page, select the **Activity** tab to view the **Raw logs** page.
5. Select the **Query name**, and then select **All throttling events** for all Odata and custom services activities.
6. Select the **Query name**, and then select **Requests throttled** for all Odata and custom services requests that have been throttled.

Priority-based throttling FAQ

2/18/2021 • 2 minutes to read • Edit Online

IMPORTANT

Some or all of the functionality noted in this topic is available as part of a preview release. The content and the functionality are subject to change. For more information about preview releases, see Service update availability.

This topic provides answers to some frequently asked questions (FAQ) about priority-based throttling for Open Data Protocol (OData) and custom service-based integrations.

How do I access the Data management Yammer group?

Follow this link: Data management Yammer group.

Will a retry request receive preferential treatment over a new request?

No.

Is there a report that determines when throttling might occur?

Yes. A report will be provided and can be accessed through the **Raw logs** within environment monitoring page in Microsoft Dynamics Lifecycle Services (LCS).

Will throttling affect the Data Import/Export Framework (DIXF) and batch?

No. Throttling is only for OData and custom service integrations.

In Preview, will my requests be throttled if priorities aren't configured?

No, because only the telemetry is available. The actual throttling occurs if you configure priorities. We recommend that you use this approach in **non-production** environments during the Preview period.

What happens to requests if the user didn't retry a throttled request?

Currently, if a request isn't retried when a 429 error is received, the request won't be processed.

Will historic throttling information be used to advise me when I resize environments?

Yes. For one month, you can export the information to Excel for more analysis and archiving.

Is throttling functionality version-specific? If it is, which version is it available in?

Priority-based throttling will be available in Preview starting with the **Platform updates for version 10.0.13**

of Finance and Operations apps release.

Are there plans to provide an option for the Priority mapping grid entry?

Microsoft will consider this request in a future release.

Will the requests of my interactive users be throttled?

No. There will be no impact on the requests of interactive (online) users.

If I experience a performance issue in Dynamics 365 Finance while a page is being loaded or a business document is being processed, how does that performance issue differ from throttling?

Throttling helps maintain a healthy system when there is a resource constraint. It won't affect any page actions.

How can I determine the wait time before I retry a throttled request?

When a request is throttled, the response header includes a time that will be used in retry logic.

Do you recommend that I use a dedicated integration account instead of just the generic admin user account?

Yes, we recommend this approach. However, it isn't mandatory.

Does throttling depend on the tier that my environment is running on?

In the initial release, no. Throttling will calculate its threshold based on the resources that are available for each environment.

Is throttling functionality legal entity–specific?

No.

Does throttling affect bring your own database (BYOD) export?

No.

Will throttling monitoring be available in Application Insights?

Monitoring will be onboarded to any available tool in the future.

If a production environment regularly runs out of resources, will Microsoft have to resize it?

Yes. Sizing estimate will also have to be revalidated and uploaded.

After April 2021, will the system be able to override priority-based throttling?

The system will use default values if no priorities are configured after April 2021.

Can the throttling engine be configured (thresholds)?

No.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Data management package REST API

2/18/2021 • 10 minutes to read • Edit Online

This topic describes the Data management framework's package representational state transfer (REST) application programming interface (API). The package API lets you integrate by using data packages. The REST API can be used with both cloud deployments and on-premises deployments. For on-premises deployments, this functionality is currently available for Microsoft Dynamics 365 for Finance and Operations, Enterprise edition with platform update 12 (November 2017) (version 7.2), build 7.0.4709.41184, and later.

Although on-premises support has been added, API names haven't been changed. Therefore, Microsoft can keep a single API set for both cloud deployments and on-premises deployments.

Choosing an integration API

Two APIs support file-based integration scenarios: the Data management framework's package API and the recurring integrations API. Both APIs support both data import scenarios and data export scenarios. The following table describes the main decision points that you should consider when you're trying to decide which API to use.

DECISION POINT	RECURRING INTEGRATIONS API	DATA MANAGEMENT FRAMEWORK'S PACKAGE API
Scheduling	Scheduling in Finance and Operations apps	Scheduling outside Finance and Operations apps
Format	Files and data packages	Only data packages
Transformation	Support for Extensible Stylesheet Language Transformations (XSLT) if the data file is in XML format	Transformations that are external to the system
Supported protocols	SOAP and REST	REST
Service type	Custom service	Open Data Protocol (OData) action
Availability	Microsoft Dynamics Finance and Operations (February 2016) and later. Note: This is not supported with the on-premises version of Dynamics 365 Finance and Operations.	Microsoft Dynamics 365 for Finance and Operations with platform update 5 (March 2017) and later

If you decide that the recurring integrations API meets your requirement better than the Data management framework's package API, see Recurring integrations. The rest of this topic discusses the Data management framework's package API.

Authorization

The Data management framework's package API uses OAuth 2.0 to authorize access. The API must be called by using a valid OAuth access token. For more information about OAuth 2.0 and Microsoft Azure Active Directory (Azure AD), see Authorize access to web applications using OAuth 2.0 and Azure Active Directory. For on-premises deployments, Active Directory Federation Services (AD FS) is used for authorization.

NOTE

When you use the Client Credentials Grant flow, the application maintains an access control list. You can find the access control list at **System administration** > **Setup** > **Azure Active Directory applications**. The **Azure Active Directory applications** page shows the approved client IDs and the user security mapping that should be enforced when the API is called by using the Client Credentials Grant flow.

For on-premises deployments, this list must have a valid client ID from AD FS. Additionally, for on-premises use, **<baseurl>** in the following examples must append **/namespaces/AXSF** when a connection is made.

Import APIs

The following APIs are used to import files (data packages).

GetImportStagingErrorFileUrl

The GetImportStagingErrorFileUrl API is used to get the URL of the error file containing the input records that failed at the source to the staging step of import for a single entity. An empty string is returned if no error file is generated.

POST /data/DataManagementDefinitionGroups/Microsoft.Dynamics.DataEntities.GetImportStagingErrorFileUrl

```
Body
{
 "executionId":"<string>",
 "entityName":"<string>"
}

Successful Response:

HTTP/1.1 200 OK
{
 "@odata.context":"https://<baseurl>/data/$metadata#Edm.String",
 "value":"<errorfileurl>"
}
```

Input parameters

PARAMETER	DESCRIPTION
string executionId	Execution ID of import. This is called as Job ID in the UI.
string entityName	Name of the entity for which to get the error file.

Output parameters

PARAMETER	DESCRIPTION
string errorkeysfileurl	The URL of the error file. The return value is empty if an error file was not generated.

GenerateImportTargetErrorKeysFile

The GenerateImportTargetErrorKeysFile API is used to generate an error file containing the keys of the import records that failed at the staging to target step of import for a single entity.

If this API returns true, then use the GetImportTargetErrorKeysFileUrl API to get the URL of the generated error keys file.

POST
/data/DataManagementDefinitionGroups/Microsoft.Dynamics.DataEntities.GenerateImportTargetErrorKeysFile

```
Body

{
 "executionId":"<string>",
 "entityName":"<string>"
}

Successful Response:

HTTP/1.1 200 OK
{
 "@odata.context":"https://<baseurl>/data/$metadata#Edm.Boolean",
 "value": <errorsExist>
}
```

Input parameters

PARAMETER	DESCRIPTION
string executionId	Execution ID of import
string entityName	Name of the entity for which to get the error file

Output parameters

PARAMETER	DESCRIPTION
Boolean errorsExist	true if there are import errors; false if there are no errors

GetImportTargetErrorKeysFileUrl

The **GetImportTargetErrorKeysFileUrl** API is used to get the URL of the error file that contains the keys of the import records that failed at the staging-to-target step of import for a single entity.

If the error file is available, this API returns the URL. If the error file is still being generated, or if there is no error file, the API returns an empty string.

> **IMPORTANT**
> Before you call this API, call the **GenerateImportTargetErrorKeysFile** API to generate the error file. If the **GenerateImportTargetErrorKeysFile** API returns **true**, call this API in a loop until it returns a non-empty string. If the **GenerateImportTargetErrorKeysFile** API returns **false**, this API will always return an empty string, because there are no errors.

Pseudocode example

```
errorsExist = GenerateImportTargetErrorKeysFile(executionId, entityName)

if (errorsExist)
{
    errorFileUrl = null

    while (errorFileUrl is not a non-empty string)
    {
        errorFileUrl = GetImportTargetErrorKeysFileUrl(executionId, entityName)
        if (errorFileUrl is not a non-empty string)
        {
            wait for some time before retrying
        }
    }
}
```

```
POST
/data/DataManagementDefinitionGroups/Microsoft.Dynamics.DataEntities.GetImportTargetErrorKeysFileUrl

Body
{
    "executionId":"<string>",
    "entityName":"<string>"
}
```

Here is an example of a successful response.

```
HTTP/1.1 200 OK
{
    "@odata.context":"https://<baseurl>/data/$metadata#Edm.String",
    "value":"<errorkeysfileurl>"
}
```

Input parameters

PARAMETER	DESCRIPTION
string executionId	The execution ID of the import. This is called as Job ID in the UI.
string entityName	The name of the entity to get the error file for.

Output parameters

PARAMETER	DESCRIPTION
string errorkeysfileurl	The URL of the error keys file, if the file is available. If the error file is still being generated, or if no errors exist, the method returns an empty string.

GetAzureWritableUrl

The **GetAzureWritableUrl** API is used to get a writable blob URL. The method includes a shared access signature (SAS) token that is embedded in the URL. You can use this method to upload a data package to the Azure Blob storage container. For on-premises deployments, this API will still return the URL that has been abstracted to local storage.

NOTE

An SAS is valid only during an expiry time window. Any request that is issued after the window has passed returns an error. For more information, see Using shared access signatures (SAS).

```
POST /data/DataManagementDefinitionGroups/Microsoft.Dynamics.DataEntities.GetAzureWriteUrl
BODY
{
    "uniqueFileName":"<string>"
}
```

Here is an example of a successful response.

```
HTTP/1.1 200 OK
{
    "@odata.context":"https://<baseurl>/data/$metadata#Edm.String",
    "value": "{\"BlobId\":\"
{<GUID>}\",\"BlobUrl\":\"https://<baseurl_id>.blob.core.windows.net/dmf/<uniqueFileName>?<SAS Token>\"}"
    }
}
```

Input parameters

PARAMETER	DESCRIPTION
string uniqueFileName	A unique file name that is used to track blob IDs. You can include a globally unique identifier (GUID) to help guarantee a unique file name.

Output parameters

PARAMETER	DESCRIPTION
string BlobId	The blob ID of the allocated blob container.
string BlobUrl	A URL that has an embedded SAS token. The URL can be used to write to Blob storage.

ImportFromPackage

The **ImportFromPackage** API is used to initiate an import from the data package that is uploaded to the Blob storage that is associated with your implementation. For on-premises deployments, the import will be initiated from the local storage that the file was uploaded previously to.

NOTE

Starting in Platform update 12, the **ImportFromPackage** API supports composite entities. However, the limitation is that there can be only one composite entity in a package.

```
POST /data/DataManagementDefinitionGroups/Microsoft.Dynamics.DataEntities.ImportFromPackage
BODY
{
    "packageUrl":"<string>",
    "definitionGroupId":"<string>",
    "executionId":"<string>",
    "execute":<bool>,
    "overwrite":<bool>,
    "legalEntityId":"<string>"
}
```

Here is an example of a successful response.

```
HTTP/1.1 200 OK
{
    "@odata.context":"https://<baseurl>/data/$metadata#Edm.String",
    "value":"<executionId>"
}
```

Input parameters

PARAMETER	DESCRIPTION
string packageUrl	The URL of the data package in the Blob storage that is associated with a Finance and Operations app.
string definitionGroupId	The name of the data project for import.
string executionId	The ID to use for the job. This is called as Job ID in the UI. If an empty ID is assigned, a new execution ID will be created.
bool execute	Set this parameter to **True** to run the target step. Otherwise, set it to **False**.
bool overwrite	This parameter must always be set to **False** when a composite entity is used in a package. Otherwise, set it to **True**.
string legalEntityId	The legal entity for the data import.

Output parameters

PARAMETER	DESCRIPTION
string executionId	The execution ID of the data import. This is called as Job ID in the UI.

> **NOTE**
>
> **ImportFromPackage()** uses a batch to perform the import. Therefore, parallel processing rules must be used in Data management to perform parallel imports. **ImportFromPackage()** must not be called in parallel threads. Otherwise, it will fail.

Export APIs

The following APIs are used to export files (data packages).

ExportToPackage

The **ExportToPackage** API is used to initiate an export of a data package. This API is applicable to both cloud deployments and on-premises deployments.

- The export data project must be created before you call this API. If the project doesn't exist, a call to the API returns an error.
- If change tracking has been turned on, only records that have been created or updated since the last run are exported. (In other words, only the delta is returned.)

```
POST /data/DataManagementDefinitionGroups/Microsoft.Dynamics.DataEntities.ExportToPackage
BODY
{
    "definitionGroupId":"<Data project name>",
    "packageName":"<Name to use for downloaded file.>",
    "executionId":"<Execution Id if it is a rerun>",
    "reExecute":<bool>,
    "legalEntityId":"<Legal entity Id>"
}
```

Here is an example of a successful response.

```
HTTP/1.1 200 OK
{
    "@odata.context":"https://<baseurl>/data/$metadata#Edm.String",
    "value":{
        "value":"<executionId>"
    }
}
```

Input parameters

PARAMETER	DESCRIPTION
string definitionGroupId	The name of the data project for export.
string packageName	The name of the exported data package.
string executionId	The ID to use for the job. This is called as Job ID in the UI. If an empty ID is assigned, a new execution ID will be created.
bool reExecute	Set this parameter to **True** to run the target step. Otherwise, set it to **False**.
string legalEntityId	The legal entity for the data import.

Output parameters

PARAMETER	DESCRIPTION
string executionId	The execution ID of the data export. This is called as Job ID in the UI.

GetExportedPackageUrl

The **GetExportedPackageUrl** API is used to get the URL of the data package that was exported by a call to

ExportToPackage. This API is applicable to both cloud deployments and on-premises deployments.

```
POST /data/DataManagementDefinitionGroups/Microsoft.Dynamics.DataEntities.GetExportedPackageUrl
BODY
{"executionId":"<Execution Id>"}
```

Here is an example of a successful response.

```
HTTP/1.1 200 OK
{
    "@odata.context":"https://<baseurl>/data/$metadata#Edm.String",
    "value":{
        "value":"https://<baseurl_id>.blob.core.windows.net/dmf/<uniqueFileName>?<SAS Token>"
    }
}
```

Input parameters

PARAMETER	DESCRIPTION
string executionId	The execution ID of the data project run. This is called as Job ID in the UI.

Output parameters

PARAMETER	DESCRIPTION
string BlobUrl	A blob URL that has an embedded SAS token. The URL can be used to download the exported data package.

Status check API

The following APIs are used to check status. They are used during both import flows and export flows.

GetExecutionSummaryStatus

The **GetExecutionSummaryStatus** API is used for both import jobs and export jobs. It's used to check the status of a data project execution job. This API is applicable to both cloud deployments and on-premises deployments.

```
POST /data/DataManagementDefinitionGroups/Microsoft.Dynamics.DataEntities.GetExecutionSummaryStatus
BODY
{"executionId":"<executionId>"}
```

Here is an example of a successful response.

```
HTTP/1.1 200 OK
{
    "@odata.context":"https://<baseurl>/data/$metadata#Edm.String",
    "value":"<executionStatus>"
}
```

Input parameters

PARAMETER	DESCRIPTION
string executionId	The execution ID of the data project run. This is called as Job ID in the UI.

Output parameters

PARAMETER	DESCRIPTION
DMFExecutionSummaryStatus executionStatus	The execution status. Here are the possible values: • Unknown • NotRun • Executing • Succeeded • PartiallySucceeded • Failed • Canceled

> **NOTE**
> The file in Blob storage will remain there for seven days. It will then be automatically deleted.

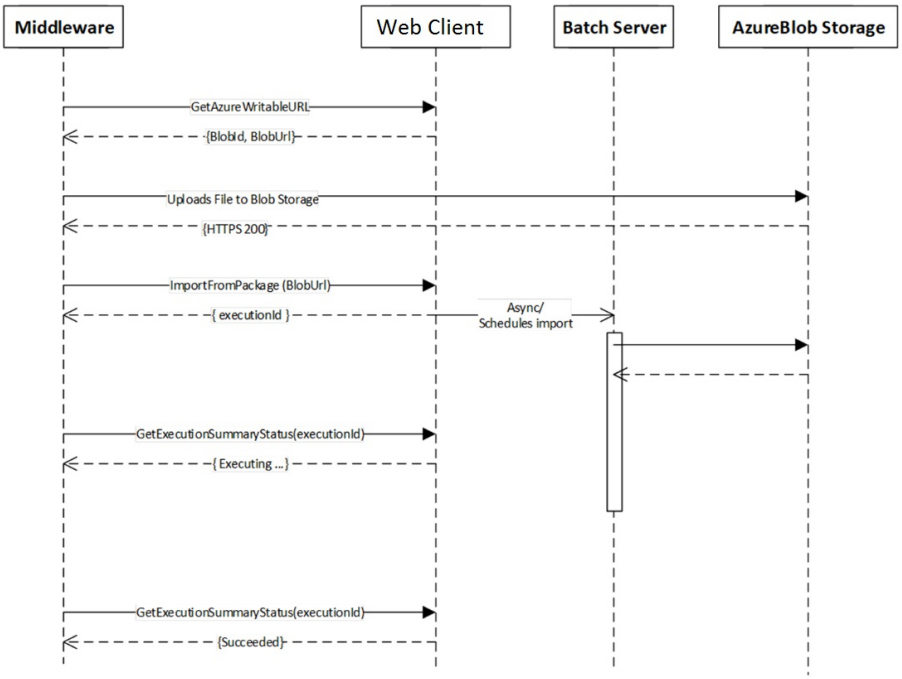
Getting the list of errors

GetExecutionErrors can be used to get the list of errors in a job execution. The API takes the Execution ID as the parameter, and returns a set of error messages in a JSON list.

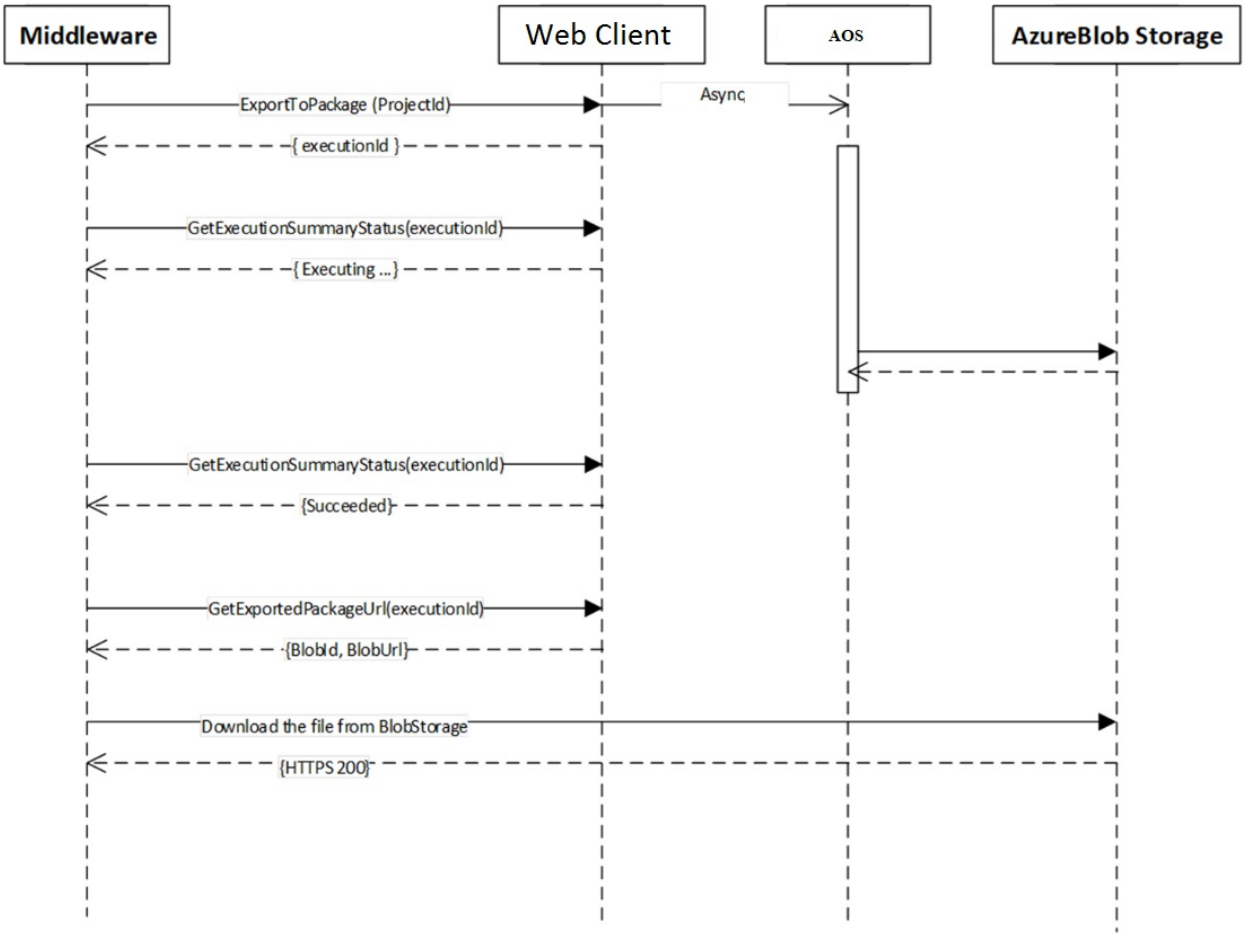
```
POST /data/DataManagementDefinitionGroups/Microsoft.Dynamics.DataEntities.GetExecutionErrors
BODY
{"executionId":"<executionId>"}
```

Import and export processes

The following illustration shows how the Data management package methods can be used to import data packages.



The following illustration shows how the Data management package methods can be used to export data packages.



A sample console application that showcases the data import and data export methods is available on GitHub. For more information, go to https://github.com/Microsoft/Dynamics-AX-Integration/tree/master/FileBasedIntegrationSamples/ConsoleAppSamples.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Service endpoints overview

2/18/2021 • 4 minutes to read • Edit Online

This topic describes the service endpoints that are available in Microsoft Dynamics 365 Finance. It also provides a comparison to the endpoints that are available in Microsoft Dynamics AX 2012.

List of services

The following table lists all the service endpoints, and compares the endpoints available for the application, and AX 2012.

SERVICE ENDPOINT	AX 2012	FINANCE AND OPERATIONS
Document services (AXDs)	Yes	No – Replaced by data entities
SOAP-based metadata service	Yes	No – Replaced by REST metadata
SOAP-based query service	Yes	No – Replaced by OData
OData query service	Yes	No – Replaced by OData
SOAP-based custom service	Yes	Yes
JSON-based custom service	No	Yes
OData Service	No	Yes
REST Metadata Service	No	Yes

This topic describes authentication for services, and the REST Metadata service. The following links provide detailed documentation for:

- Custom service development
- Open Data Protocol (OData)

Authentication

OData services, JSON-based custom services, and the REST metadata service support standard OAuth 2.0 authentication.

We currently support both Authorization Code Grant flow and Service to service calls using client credentials (shared secret or certificate).

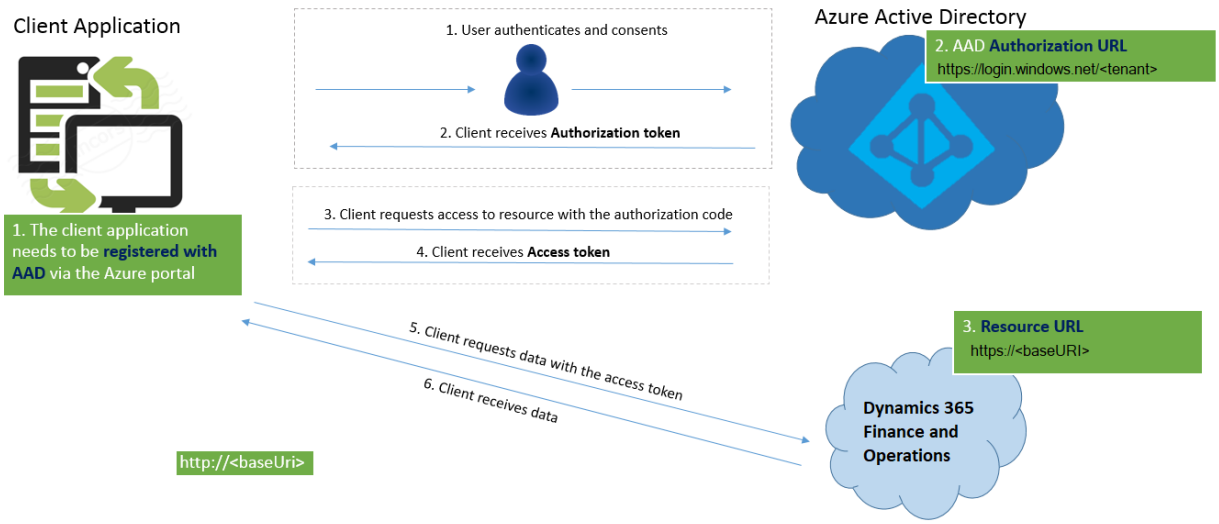
Two kinds of application are supported in Microsoft Azure Active Directory (AAD):

- **Native client application** – This flow uses a user name and password for authentication and authorization.
- **Web application (Confidential client)** – A confidential client is an application that can keep a client password confidential to the world. The authorization server assigned this client password to the client application.

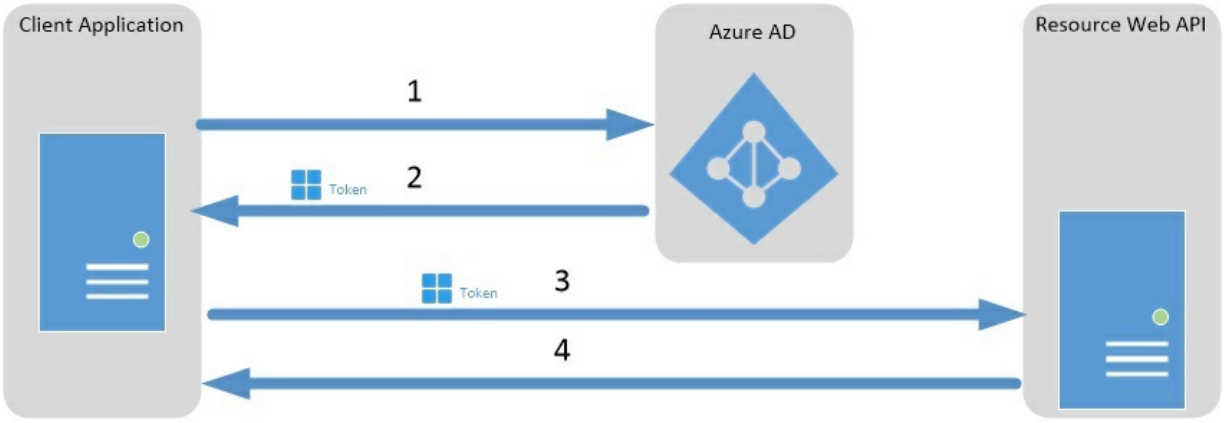
For more information, see:

- Authorize access to web applications using OAuth 2.0 and Azure Active Directory
- Troubleshoot service authentication issues

The following illustration describes how authorization must be configured for Authorization code grant flow.



And below is the illustration describes how authorization works for Service to service calls using client credentials (shared secret or certificate).



Register a web application with AAD

> **NOTE**
> These steps don't have to be completed by all the people in your organization. Only one Azure Service Administrator user can add the application and share the client ID with the developers.

Prerequisite: You must have an Azure subscription and admin access to Azure Active Directory (Azure AD).

Before any clients can communicate with the services, they must be registered in (Azure AD). These steps will help you register an application with (Azure AD). The steps are explained in the Azure app registration training guide. For specific configuration in this process, the following additional information must be used in context.

Select **Microsoft Dynamics ERP (Microsoft.ERP)**. If you search for **Microsoft Dynamics ERP** in the search field within **Select an API** it might appear to be unavailable. In that case, make sure that you search for the full name, as shown above. Under **Delegated permissions**, you must select, at a minimum, the following options:

- Access Dynamics AX Custom Service
- Access Dynamics AX data

- Access Dynamics AX online as organization users

> **IMPORTANT**
> Make sure that you copy the key, because you won't see it again. You will be required to know this secret key to complete your OAuth authentication and receive an Azure AD token.

Register your external application

1. In Finance and Operations apps, go to **System administration** > **Setup** > **Azure Active Directory applications**.
2. Select **New**.
3. Fill in the fields for the new record:
   - In the **Client Id** field, enter the application ID that you registered in Azure AD.
   - In the **Name** field, enter a name for the application.
   - In the **User ID** field, select an appropriate service account user ID. For this example, we have selected the **Admin** user. However, as a better practice, you should provision a dedicated service account that has the correct permissions for the operations that must be performed.

   When you've finished, select **Save**.

You've now finished setting up the prerequisites. After the external application retrieves an Azure AD authentication token, it should now be able to use the token in an authorization HTTP header to make subsequent service calls via OData or SOAP, for example.

Client sample code

The following is C# sample code for getting a token from AAD. In this flow, the user will be presented with a consent form (for cross-tenant application) and a sign-in form.

```
UriBuilder uri = new UriBuilder ("https://login.windows.net/contoso2ax.onmicrosoft.com");

AuthenticationContext authenticationContext = new AuthenticationContext(uri.ToString());

//request token for the resource - which is the URI for your organization. NOTE: Important do not add a
trailing slash at the end of the URI
AuthenticationResult authenticationResult =
authenticationContext.AcquireToken("https://axdynamics1001aos.cloud.dynamics.com", clientId, redirectURI);

//this gets the authorization token, which needs to be passed in the Header of the HTTP Requests
string authenticationHeader = authenticationResult.CreateAuthorizationHeader();
```

To pass the user name and password without showing a pop-up, you can use the following overload of **AcquireToken**.

```
UserCredential userCred = new UserCredential (username, password);
authenticationContext.AcquireToken("https://axdynamics1001aos.cloud.dynamics.com", clientId, userCred);
```

REST metadata service

The REST metadata service is a read-only service. In other words, users can make only GET requests. The main purpose of this endpoint is to provide metadata information for elements. It is an OData implementation.

This endpoint is hosted at `http://\[baseURI\]/Metadata`.

Currently, this endpoint provides metadata for the following elements:

- **Labels** – Returns labels from the system. Labels have a dual pair key of language and ID, so that you can retrieve the value of the label.

  Example: `https://[baseURI\]/metadata/Labels(Id='@SVC\_ODataLabelFile:Label1',Language='en-us')`

- **Data entities** – Returns a JSON-formatted list of all the data entities in the system.

  Example: `https://[baseURI\]/Metadata/DataEntities`

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Troubleshoot service authentication issues

2/18/2021 • 2 minutes to read • Edit Online

This topic provides some tips for troubleshooting issues that involve service authentication.

When you troubleshoot service authentication issues, there are a few basic and common procedures that can help resolve the issues that are most often encountered. These procedures also provide a hands-on demonstration of how the authentication mechanism works. This topic includes instructions and also lists a few common issues that users have encountered so far.

Inspect the JWT

Capture the JWT from an HTTP request

1. Download Fiddler from https://www.telerik.com/fiddler.
2. Set up HTTPS capture to watch the HTTPS traffic from the client.
3. Find the Open Authorization (OAuth) JSON Web Token (JWT). It's the value of the HTTP "Authorization" header without the "bearer" segment.

Use a deserializer tool to look at the token contents

1. Go to https://jwt.io, and paste the JWT into the input panel.
2. View the contents in the form of name-value pairs. See the example that follows.
3. Verify that the following information is correct:

   - **"aud"** – The value corresponds to the Microsoft Azure Active Directory (Azure AD) resource concept. Here are some typical issues that involve "aud":
     - The **"aud"** segment of the JWT contains a URI that has a trailing slash.
     - The **"aud"** segment of the JWT contains a URI that uses an incorrect capitalization style. The URI must be all lowercase.
   - **"appid"** – The value corresponds to the Azure AD Native Client App ID (or Service App ID).
   - **"upn"** – The value corresponds to the user who is being authenticated through a Native Client App.

The following illustration shows an example of the contents of the JWT.

```
{
  "aud":
"https://usnconeboxax1aos.cloud.onebox.dynamics.com",
  "iss": "https://sts.windows-ppe.net/4dbfcf74-c5a6-4727-
b638-d56e51d1f381/",
  "iat": 1464816034,
  "nbf": 1464816034,
  "exp": 1464819934,
  "acr": "1",
  "amr": [
    "pwd"
  ],
  "appid": "d8a9a121-b463-41f6-a86c-041272bdb340",
  "appidacr": "0",
  "family_name": "User 1",
  "given_name": "Test",
  "ipaddr": "131.107.174.200",
  "name": "Test User 1",
  "oid": "b45cc6d9-e4b7-42bf-b75d-3482ada55655",
  "puid": "100300008AC14B8D",
  "scp": "user_impersonation",
  "sub": "n6IT5RzrSP1_5kqlsYMZS4nXSZxitgh_UJOLdtc9Sd0",
  "tid": "4dbfcf74-c5a6-4727-b638-d56e51d1f381",
  "unique_name": "tusr1@taeofficial.ccsctp.net",
  "upn": "tusr1@taeofficial.ccsctp.net",
  "ver": "1.0"
}
```


Review the event logs

You can also look at the event logs of the instance machine, if you have access to the virtual machine (VM).

1. Start Event Viewer by running the **eventvwr** command from the **Run** window.
2. Go to the following channels:
   - Application and Services Logs > Microsoft > Dynamics > AX-IntegrationServices > Channel:Operational (Microsoft-Dynamics-AX-IntegrationServices/Operational)
   - Application and Services Logs > Microsoft > Dynamics > AX-SystemRuntime > Channel:Operational (Microsoft-Dynamics-AX-SystemRuntime/Operational)

Other approaches

- For more information about how OAuth is configured, see Service endpoints overview.
- You can also try to call the service in parallel by using your own client code. The sample code that we published is available at https://github.com/Microsoft/Dynamics-AX-Integration.
- If the second method works, you can compare the JWTs from each method.

Known issues

AADSTS65001: The user or administrator hasn't consented to use the application

- The **"aud"** segment of the JWT might contain a URI that has a trailing slash. The slash must be removed.
- The **"aud"** segment of the JWT might contain a URI that uses an incorrect capitalization style. The URI must be all lowercase.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Open Data Protocol (OData)

2/18/2021 • 8 minutes to read • Edit Online

This topic provides information about Open Data Protocol (OData) and explains how you can use OData V4 to expose updatable views.

What is OData?

OData is a standard protocol for creating and consuming data. The purpose of OData is to provide a protocol that is based on Representational State Transfer (REST) for create, read, update, and delete (CRUD) operations. OData applies web technologies such as HTTP and JavaScript Object Notation (JSON) to provide access to information from various programs. OData provides the following benefits:

- It lets developers interact with data by using RESTful web services.
- It provides a simple and uniform way to share data in a discoverable manner.
- It enables broad integration across products.
- It enables integration by using the HTTP protocol stack.

For more information about OData, see the following webpages.

TOPIC	WEBPAGE
OData standards	https://www.odata.org/documentation/
OData: Data access for the web, the cloud, mobile devices, and more	/aspnet/web-api/overview/odata-support-in-aspnet-web-api/

The public OData service endpoint enables access to data in a consistent manner across a broad range of clients. To see a list of all the entities that are exposed, open the OData service root URL. The URL for the service root on your system has the following format: **[Your organization's root URL]/data**

> **NOTE**
> OData actions added via extensions are currently not supported.

Addressing

The following table describes the resources and the corresponding URLs in the Fleet Management sample.

RESOURCE	URL	DESCRIPTION
Service endpoint	[Your organization's root URL]/data/	The root service endpoint for OData entities
Entity collection	[Your organization's root URL]/data/Customers	The collection of all customers
Entity	[Your organization's root URL]/data/Customers("[key]")	A single entity from the entity collection

RESOURCE	URL	DESCRIPTION
Navigation property	[Your organization's root URL]/data/Customers("[key]")/Reservations	The navigation from a customer to that customer's reservations
Property	[Your organization's root URL]/data/Customers("[key]")/FirstName	The customer's first name

OData services

We provide an OData REST endpoint. This endpoint exposes all the data entities that are marked as **IsPublic** in the Application Object Tree (AOT). It supports complete CRUD (create, retrieve, update, and delete) functionality that users can use to insert and retrieve data from the system. Detailed labs for this feature are on the LCS methodology.

> **NOTE**
> When working with data entities using OData, all fields in the entity key must be provided to make a successful OData call.

Code examples for consuming OData services are available in the Microsoft Dynamics AX Integration GitHub repository.

Supported features from the OData specification

The following are the high-level features that are enabled for the OData service, per the OData specification.

- CRUD support is handled through HTTP verb support for POST, PATCH, PUT, and DELETE.
- Available query options are:
  - $filter
  - $count
  - $orderby
  - $skip
  - $top
  - $expand (only first-level expansion is supported)
  - $select
- The OData service supports serving driven paging with a maximum page size of 10,000.

For more information, see: OData actions that are bound to entities.

Filter details

There are built-in operators for $filter:

- Equals (eq)
- Not equals (ne)
- Greater than (gt)
- Greater than or equal (ge)
- Less than (lt)
- Less than or equal (le)
- And

- Or
- Not
- Addition (add)
- Subtraction (sub)
- Multiplication (mul)
- Division (div)
- Decimal division (divby)
- Modulo (mod)
- Precedence grouping ({ })

You can also use the **Contains** option with $filter requests. It has been implemented as a wildcard character. For example: `http://host/service/EntitySet?$filter=StringField eq '\*retail\*'`

The operators 'has' and 'in' are not supported.

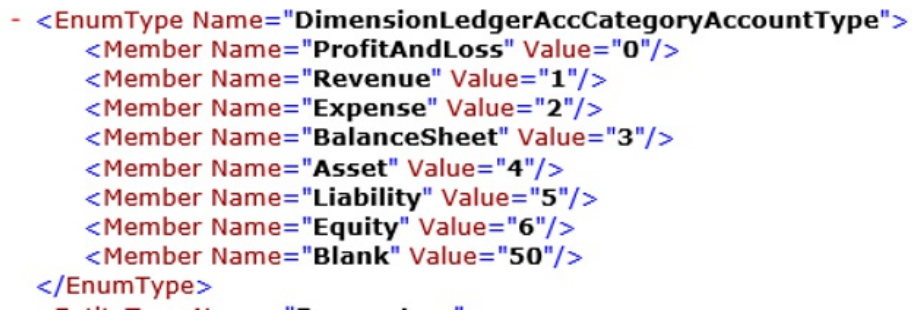
For more information, see OData operators.

Batch requests

Batch requests are supported in the OData service. For more information, see OData batch requests.

Metadata annotations

/data/$metadata provides annotations. EnumType is support in $metadata.



Cross-company behavior

By default, OData returns only data that belongs to the user's default company. To see data from outside the user's default company, specify the **?cross-company=true** query option. This option will return data from all companies that the user has access to.

Example: `http://[baseURI\]/data/FleetCustomers?cross-company=true`

To filter by a particular company that isn't your default company, use the following syntax:

```
http://[baseURI\]/data/FleetCustomers?$filter=dataAreaId eq 'usrt'&cross-company=true
```

Validate methods

The following table summarizes the validate methods that the OData stack calls implicitly on the corresponding data entity.

ODATA	METHODS (LISTED IN THE ORDER IN WHICH THEY ARE CALLED)

ODATA	METHODS (LISTED IN THE ORDER IN WHICH THEY ARE CALLED)
Create	1. **Clear()** 2. **Initvalue()** 3. **PropertyInfo.SetValue()** for all specified fields in the request 4. **Validatefield()** 5. **Defaultrow** 6. **Validatewrite()** 7. **Write()**
Update	1. **Forupdate()** 2. **Reread()** 3. **Clear()** 4. **Initvalue()** 5. **PropertyInfo.SetValue()** for all specified fields in the request 6. **Validatefield()** 7. **Defaultrow()** 8. **Validatewrite()** 9. **Write()**
Delete	1. **Forupdate()** 2. **Reread()** 3. **checkRestrictedDeleteActions()** 4. **Validatedelete()** 5. **Delete()**

Exposing OData entities

OData entities are based on the concept of an updatable view. When the **IsPublic** property for an updatable view is set to **TRUE**, that view is exposed as a top-level OData entity.

Setting navigation properties between OData entities

Links between OData entities are described by a navigation property. Navigation properties describe the navigation from one end of an association to the other end.

Adding actions on OData entities

Actions let you inject behaviors into the data model. To add actions, add a method to the updatable view, and decorate that method with specific attributes. Here is an example.

```
[SysODataActionAttribute("CalcMaintenanceDuration", true)]
public int CalculateMaintenanceDuration()
{
    //do something
    return 0;
}
```

In this example, the **SysODataActionAttribute** class decorates the **CalculateMaintenanceDuration** method that is exposed as an action. The first argument of the attribute is the publicly exposed name of the action, and the second argument indicates whether this action is always available. Methods that are exposed as actions can return any primitive type or another public updatable view. After this method is exposed, it appears in the OData

$metadata. Here is an example.

```
<Action Name="CalcMaintenanceDuration" IsBound="true">
    <Parameter Name="ViewMaintenance" Type="Microsoft.Dynamics.AX.Resources.ViewMaintenance"/>
    <ReturnType Type="Edm.String" />
</Action>
```

The following example of an OData action takes in a parameter and returns a list.

```
[SysODataActionAttribute("GetColors", true),
    SysODataCollectionAttribute("return", Types::Record, "CarColor")]
public List GetColorsByAvailability(boolean onlyAvailableVehicles)
{
    List returnList = new List(Types::Record);
    // do something
    return returnList;
}
```

In this example, the **SysODataCollectionAttribute** class enables OData to expose strongly typed collections from X++. This class takes in three parameters:

- The name of the parameter that is a list (Use **return** for the return value of the method.)
- The X++ type of the members of this list
- The public name of the OData resource that is contained in the collection

After these actions are exposed, they can be invoked from the service root URL.

You can find actions that are defined on data entities by searching for the **SysODataActionAttribute** attribute in the source code.

Querying or browsing an OData endpoint

OData enables an SQL-like language that lets you create rich queries against the database, so that the results include only the data items that you want. To create a query, append criteria to the resource path. For example, you can query the **Customers** entity collection by appending the following query options in your browser.

URL	DESCRIPTION
[Your organization's root URL]/data/Customers	List all the customers.
[Your organization's root URL]/data/Customers?$top=3	List the first three records.
[Your organization's root URL]/data/Customers?$select=FirstName,LastName	List all the customers, but show only the first name and last name properties.
[Your organization's root URL]/data/Customers?$format=json	List all the customers in a JSON format that can be used to interact with JavaScript clients.

The OData protocol supports many similar filtering and querying options on entities. For the full set of query options, see Windows Communication Foundation.

Using Enums

Enums are under namespace **Microsoft.Dynamics.DataEntities**. Enums can be included in an OData query is by using the following syntax.

`Microsoft.Dynamics.DataEntities.Gender'Unknown'`

`Microsoft.Dynamics.DataEntities.NoYes'Yes'`

An example query for using the above enum values is shown below.

```
https://environment.cloud.onebox.dynamics.com/data/CustomersV3?\$filter=PersonGender eq
Microsoft.Dynamics.DataEntities.Gender'Unknown'
```

```
https://environment.cloud.onebox.dynamics.com/data/Currencies?\$filter=ReferenceCurrencyForTriangulation eq
Microsoft.Dynamics.DataEntities.NoYes'No'
```

The operations supported for enums are **eq** and **ne**.

Authentication

OData sits on the same authentication stack as the server. For more information about the authentication, see Service endpoints overview.

Tips and tricks

Run multiple requests in a single transaction

The OData batch framework uses *changesets*. Each changeset contains a list of requests that should be treated as single atomic unit. In other words, either all the requests are run successfully or, if any request fails, none of the requests are run successfully. The following example shows how to send a batch request that has a list of requests in a single changeset.

The **SaveChangesOptions.BatchWithSingleChangeset** option in **SaveChanges()** helps guarantee that all requests are bundled into a single changeset.

```
public static void CreateProductColors(Resources context)
{
    var productColorsCollection = new DataServiceCollection<ProductColor>(context);

    var color1 = new ProductColor();
    productColorsCollection.Add(color);
    color1.ColorId = "New Color1"; // set any other properties needed

    var color2 = new ProductColor();
    productColorsCollection.Add(color1);
    color2.ColorId = "New Color2"; // set any other properties needed

    context.SaveChanges(SaveChangesOptions.BatchWithSingleChangeset);
}
```

Prevent unset records from being posted when you use an OData client

When you create a new record by using an OData client, as shown in example 1, properties that aren't set are included in the body of the request, and default values are assigned to them. To prevent this behavior and post only properties that are set explicitly, use the **SaveChangesOptions.PostOnlySetProperties** option in **SaveChanges()**, as shown in example 2.

Example 1

```
public static void CreateVendor(Resources context)
{
    var vendorCollection = new DataServiceCollection<Vendor>(context);
    var vendor = new Vendor();
    vendorCollection.Add(vendor);
    // set properties
    context.SaveChanges();
}
```

Example 2

```
public static void CreateVendor(Resources context)
{
    var vendorCollection = new DataServiceCollection<Vendor>(context);
    var vendor = new Vendor();
    vendorCollection.Add(vendor);
    // set properties

    // Save specifying PostOnlySetProperties flag
    context.SaveChanges(SaveChangesOptions.PostOnlySetProperties);
}
```

Handling duplicate names between enums and entities in metadata

There are instances where enums and entities share the same name. This name duplication results in OData client code generation errors. To recover from this error, the helper code in GitHub can be used to identify duplicate name instances that must be removed. The generated metadata document can be used for further processing of the OData logic on the client side.

Array fields

OData does not support array fields in entities. This must be taken into consideration when designing entities that will be used with OData.

After restarting AOS, the first OData call may take a long time to process

The first OData call processed by an AOS that was restarted may take a long time to process because the metadata is not being cached. This latency can be avoided by warming up OData on AOS startup. For more details, see Build OData metadata cache when the AOS starts.

Custom service development

2/18/2021 • 2 minutes to read • Edit Online

You can develop custom services for Finance and Operations. When a developer writes a custom service under a service group, the service group is always deployed on two endpoints:

- SOAP endpoint
- JSON endpoint

SOAP-based custom service

SOAP-based services remain the same as they were in Dynamics AX 2012.

Code examples for consuming custom services using SOAP are available in the Microsoft Dynamics AX Integration GitHub repository.

Key changes

- All the service groups under the **AOTService group** node are automatically deployed.
- All services that must be deployed must be part of a service group.

Example endpoint for a dev environment

```
https://usnconeboxax1aos.cloud.onebox.dynamics.com/soap/services/UserSessionService?wsdl
```

Example endpoint for a non-dev environment

```
https://<baseurl>/soap/services/UserSessionService?wsdl
```

For more information about custom services, see:

- Using Custom Services [AX 2012] (TechNet)
- Walkthrough: Exposing an X++ Class as a Data Contract (TechNet)

JSON-based custom service

This feature enables X++ classes to be consumed as JSON services. In other words, the return data set is in JSON format. JSON, which stands for JavaScript Object Notation, is a compact, lightweight format that is commonly used communicate data between the client and the server.

The JSON Endpoint is at `https://host_uri/api/services/service_group_name/service_group_service_name/operation_name`.

Example

```
https://usnconeboxax1aos.cloud.onebox.dynamics.com/en/api/services/UserSessionService/AifUserSessionService/GetUserSessionIr
```

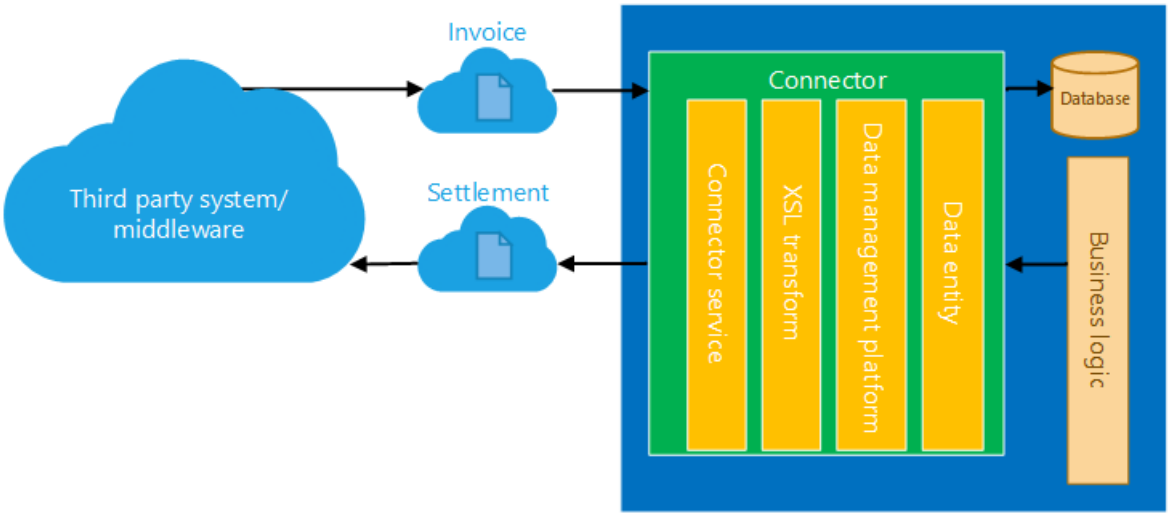
Code examples for consuming JSON services are available in the Microsoft Dynamics AX Integration GitHub repository.

Recurring integrations

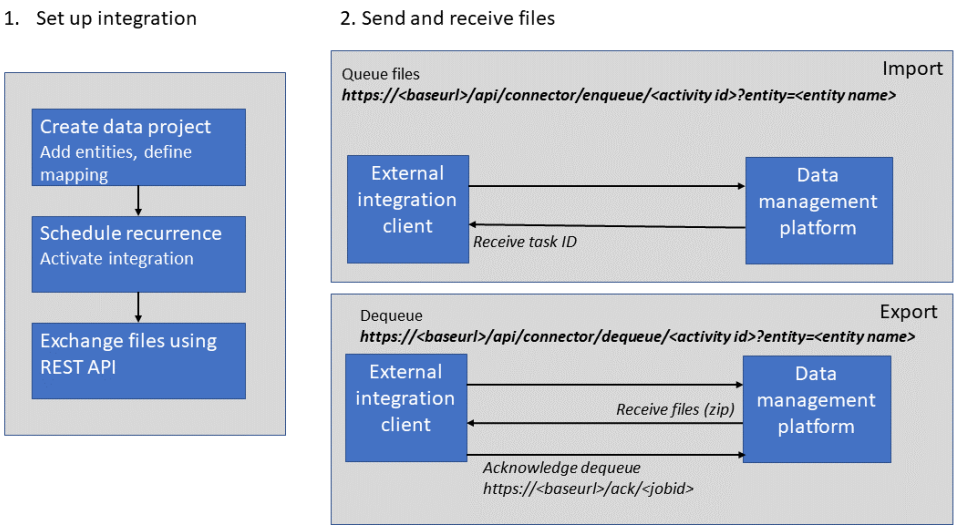
2/18/2021 • 8 minutes to read • Edit Online

Recurring integration does the following things:

- It builds on data entities and the Data management framework.
- It enables the exchange of documents or files between Finance and Operations and any third-party application or service.
- It supports several document formats, source mapping, Extensible Stylesheet Language Transformations (XSLT), and filters.



- It uses secure REST application programming interfaces (APIs) and authorization mechanisms to receive data from, and send data back to, integration systems.



Authorization for the integration REST API

The integration REST API uses the same OAuth 2.0 authentication model as the other service endpoints. Before the integrating client application can consume this endpoint, you must create an application ID in Microsoft

Azure Active Directory (Azure AD) and give it appropriate permission to the application. When you create and enable a recurring job, you're prompted to enter the Azure AD application ID that will interact with that recurring job. Therefore, be sure to make a note of the application ID.

> **NOTE**
> This feature is not supported with Dynamics 365 Finance + Operations (on-premises).

Set up a data project and recurring data jobs

Create a data project

1. On the main dashboard, select the **Data management** tile to open the **Data management** workspace.
2. Select the **Import** or **Export** tile to create a new data project.

   > **NOTE**
   > If you have an existing data project, select **Load project** on the card for any data project on the **Data projects** tab.

3. Enter a valid job name, data source, and entity name.
4. Upload a data file for one or more entities. Make sure that each entity is added, and that no errors occur.

   > **NOTE**
   > You can select each entity data card to set up, review, or modify field maps, and to set up XSLT-based transforms that must be applied to inbound data. For export data projects, the entity card also shows a filter link, so that you can set up filters to filter data. Currently, all recurring data jobs in a data project use the same filter.

5. Select **Save**.

Create a recurring data job

1. On the **Data project** page, select **Create recurring data job**.
2. Enter a valid name and a description for the recurring data job.
3. On the **Set up authorization policy** tab, enter the application ID that was generated for your application, and mark it as enabled.
4. Expand **Advanced options** tab, and specify either **File** or **Data package**.
   - **File** – Your external integration will push individual files so that they can be processed via this recurring data job. In this case, the format of the file that is expected is the same as the format that was specified when the entity was added to the data project.
   - **Data package** – You can push only data package files for processing. A data package is a new format that lets you submit multiple data files as a single unit that can be used in integration jobs.
   - **Process messages in order** – You can enable this option to force sequential processing of incoming files in an import scenario. This option is only applicable to files and not data packages.
5. Select **Set processing recurrence**, and then, in the **Define recurrence** dialog box, set up a valid recurrence for your data job.
6. Optional: Select **Set monitoring recurrence**, and set up a monitoring recurrence.

> **NOTE**
>
> Currently, monitoring recurrence enables load monitoring only on the queue for your recurring data job. No additional policies are supported via this service. You can use this feature to fine-tune the processing recurrence as the load demand requires.

7. Select **OK**, and then select **Yes** in the confirmation message box.

Manage recurring data jobs

1. In the **System administration** workspace (not the **System administration** module), select the **Data Management IT** workspace.
2. In the workspace, on the **Recurring data job** tab, select the recurring job to view the details. The **Manage scheduled data jobs** page contains a grid that lists any messages that are waiting in the queue. Therefore, you can monitor messages and the processing status.

Submitting data to recurring data jobs

You can use integration REST endpoints to integrate with the client, submit documents (import), or poll available documents for download (export). These endpoints support OAuth.

Integration REST APIs

The following set of APIs is used to exchange data between the integration client and the application.

API for import (enqueue)

Make an HTTP POST call against the following URL.

```
https://<base URL>/api/connector/enqueue/<activity ID>?entity=<entity name>
```

In the message body, you can the pass the data as a memory stream.

Example

```
POST https://usncax1aos.cloud.onebox.dynamics.com/en/api/connector/enqueue/%7B6D31E09F-0249-459F-94F0-
AAD9C2C47B64%7D?entity=Customer%20Groups
```

To get the activity ID, on the **Manage scheduled data jobs** page, in the **ID** field, copy the globally unique identifier (GUID).

API for export (dequeue)

To return a data package that contains all the data entities that were defined in the data project, and that the client application can unzip and consume, use the following structure.

```
https://<base URL>/api/connector/dequeue/<activity ID>
```

Example

```
GET https://usncax1aos.cloud.onebox.dynamics.com/en/api/connector/dequeue/%7BC03BB937-09ED-46DE-86EE-4520D7D7E373%7D
```

After the client downloads the data, an acknowledgment must be sent back to the application, so that you can mark the data as received.

In cases when there was no file uploaded to the blob, the dequeue API will return a response indicating as such.

API for acknowledgment

Use the following API.

> **NOTE**
> The body of the response of **/dequeue** must be sent in the body of the **/ack** POST request.

```
https://<base URL>/api/connector/ack/<activity ID>
```

Example

```
POST https://usncax1aos.cloud.onebox.dynamics.com/en/api/connector/ack/%7BC03BB937-09ED-46DE-86EE-4520D7D7E373%7D
```

> **NOTE**
> Until a message is successfully acknowledged, the same message will become available to dequeue every 30 minutes. In cases when a message is being dequeued more than one time, the dequeue response will send the last dequeued date time. This will be blank for the first dequeue of a message. It is important to ensure that a message is successfully acknowledged to prevent a repeated download of the same message. When an acknowledgement fails, having re-try logic to acknowledge the failure is recommended.

API for getting message status

The API to get the status of a message is available as of hotfix KB 4058074 for Platform update 12. This API is particularly useful in import scenarios to determine if a message has been successfully processed. A message is created when the enqueue process is completed. If the message returns a failed status, you can set your integration app to retry or take another action.

Example

```
POST /data/DataManagementDefinitionGroups/Microsoft.Dynamics.DataEntities.GetMessageStatus
BODY
{
    "messageId":"<string>"
}
```

The following table lists the possible status values.

VALUE	MEANING
Enqueued	The file has been successfully enqueued to blob storage
Dequeued	The file has been successfully dequeued from blob storage
Acked	The exported file has been acknowledged to be downloaded by the external application
Preprocessing	The import/export operation is pre-processing the request
Processing	The import/export operation is in process
Processed	The import/export operation completed successfully
PreProcessingError	The import/export operation failed in the pre-processing stage
ProcessedWithErrors	The import/export operation completed with errors
PostProcessingFailed	the import/export operation failed during post-processing

> **NOTE**
> The file in the blob storage will remain in the storage for seven days, after which it will be automatically deleted.

API to get the list of execution errors

GetExecutionErrors can be used to get the list of errors in a job execution. The API takes the Execution ID as the parameter, and returns a set of error messages in a JSON list.

```
POST /data/DataManagementDefinitionGroups/Microsoft.Dynamics.DataEntities.GetExecutionErrors
BODY
{"executionId":"<executionId>"}
```

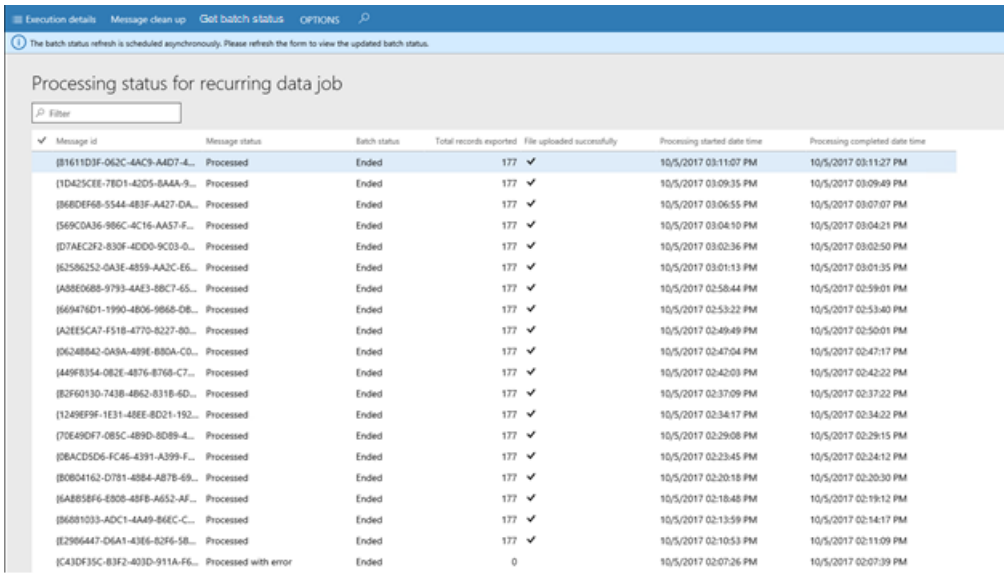
Tips and tricks

Viewing the batch job status for recurring integrations from the Data management workspace

Recurring integration data jobs run in batch mode. If a recurring job fails, you must investigate the instance of the batch job as part of the troubleshooting process. To make this investigation easier, click **Manage messages** to get to the **Process status for recurring data job** page, which now shows the status of the batch job.

The batch job status is retrieved asynchronously from the batch framework for the specified recurring data job. To see the most up-to-date batch job status, select **Get batch status**, and then refresh the page.

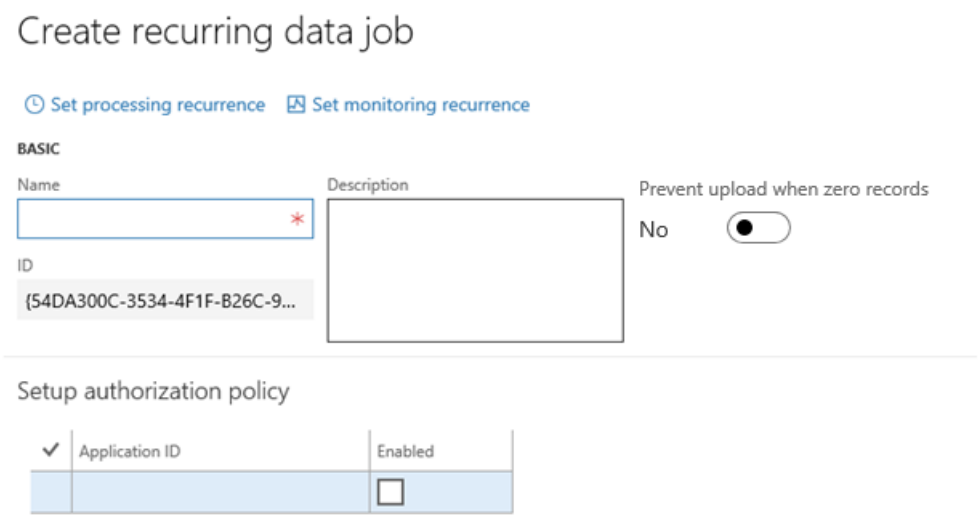
> **NOTE**
> If the record for the batch history is deleted, the status for the batch job on the **Processing status for recurring data job** page will be blank.



Preventing uploads when there are no records

When you use recurring exports you can choose not to upload a generated file or package if the total record count in that file or package is 0 (zero).

You can set the **Prevent upload when zero records** option either when you configure a recurring export job or after a job has been created. This option is available only when you use files or packages as data sources.



Your implementation might include runs of recurring jobs where files or packages were uploaded. Your implementation might also include runs where no files or packages were uploaded, because there was nothing to upload. If you suspect that a file that should have been uploaded wasn't uploaded, or that a file that should not have been uploaded was uploaded, you can use the **Manage messages** page for the recurring export job to help with the debugging process.

> **NOTE**
>
> These features were added in Microsoft Dynamics 365 for Finance and Operations, Enterprise edition platform update 12. Jobs that were run before you upgraded to Platform update 12 won't have values in the following columns.

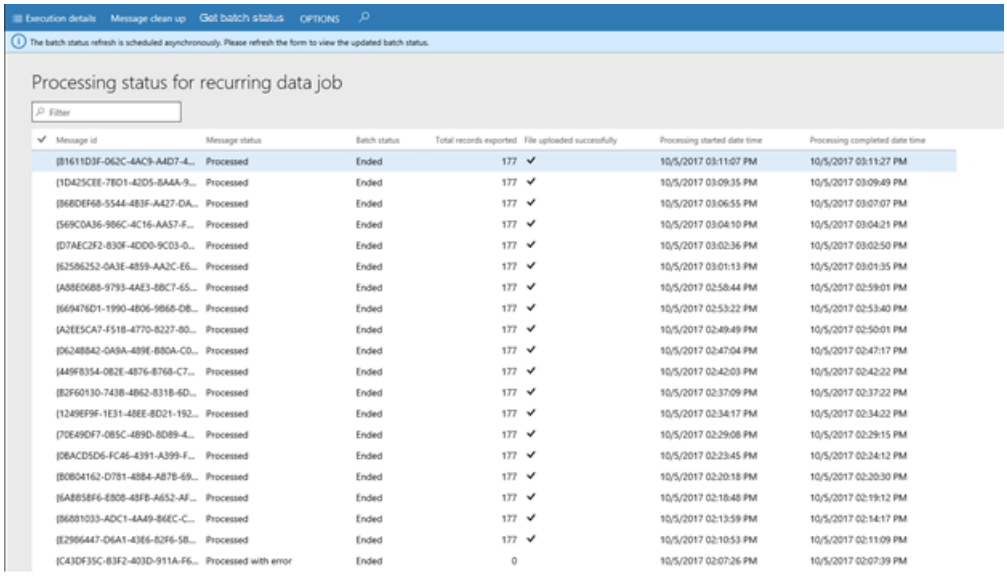
The **Total records exported** column shows the total count of records that were exported. A value of **0** (zero) indicates that no records were exported to the file or included in the package.

The **File uploaded successfully** column contains a check mark if the file or the package was uploaded

successfully. If the file wasn't uploaded because of an error, or because there were no records, the column will be blank.

Http vs Https

The dequeue API returns HTTP instead of HTTPS. This behavior can be seen in application environments that use a load balancer, such as production environments. (You cannot see the behavior in one box environments). We recommend that you change the URI scheme to HTTPS in the middleware application that is trying to dequeue from the application.



Test services by using third-party utilities

2/18/2021 • 7 minutes to read • Edit Online

At https://github.com/Microsoft/Dynamics-AX-Integration, Microsoft provides sample code for consuming services. However, there are many scenarios where the other endpoint in an integration might not use a Microsoft stack. Even when the other endpoint does use, for example, the Open Data Protocol (OData) client code that Microsoft makes available, you might find it useful to perform the following actions:

- Explore and analyze how an interaction's messages are constructed.
- Test the response of a service to a well-known request.
- Determine how exceptions will appear to the other endpoint.

Many frequently used tools that will help you perform these actions are available. This topic isn't an endorsement of any tool. Although it provides examples that use some frequently used software utilities, the principles should broadly apply to other, similar tools.

Prerequisites

Before you can test a service by using an external application, you must register the application in Microsoft Azure, and in Finance and Operations.

For details, see:

- Register an application with AAD
- Register your external application

Query OData by using Postman

Postman (https://www.getpostman.com/postman) is a tool that is often used to interact with RESTful services (such as OData) in scenarios that involve the development and testing of application programming interfaces (APIs). This procedure isn't an endorsement of Postman, and other similar tools are available. However, we are using Postman to illustrate the concepts and messages that are involved when you use OAuth to authenticate with Azure AD, and then make OData requests to and receive responses from the application.

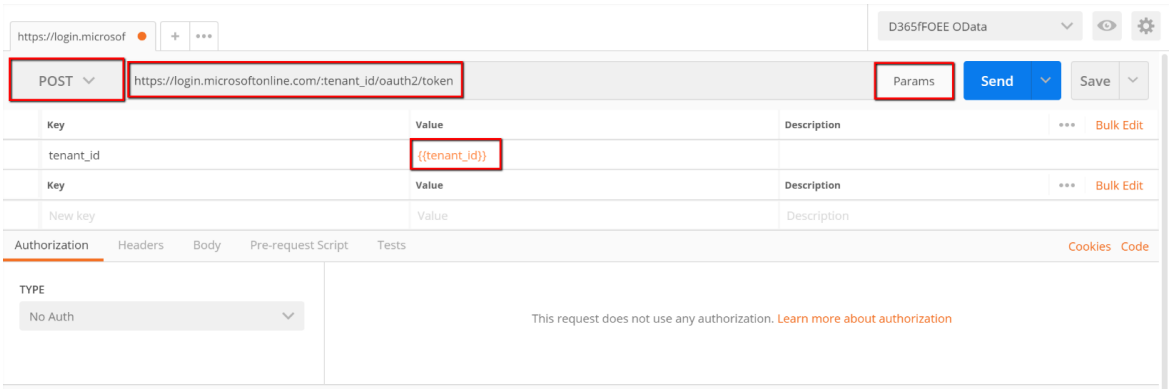
1. Start Postman.
2. In the upper-right corner, select the gear button, and then select **Manage environments** to create or update an environment.
3. Enter a name for the environment, and then select **Bulk Edit**.
4. Enter key-value pairs as shown in the following table. Enter one pair per line, and separate the key and value by using a colon (:).

KEY	VALUE
tenant_id	The Azure tenant ID that you looked up during the setup of prerequisites
client_id	The Azure AD application ID that you registered during the setup of prerequisites

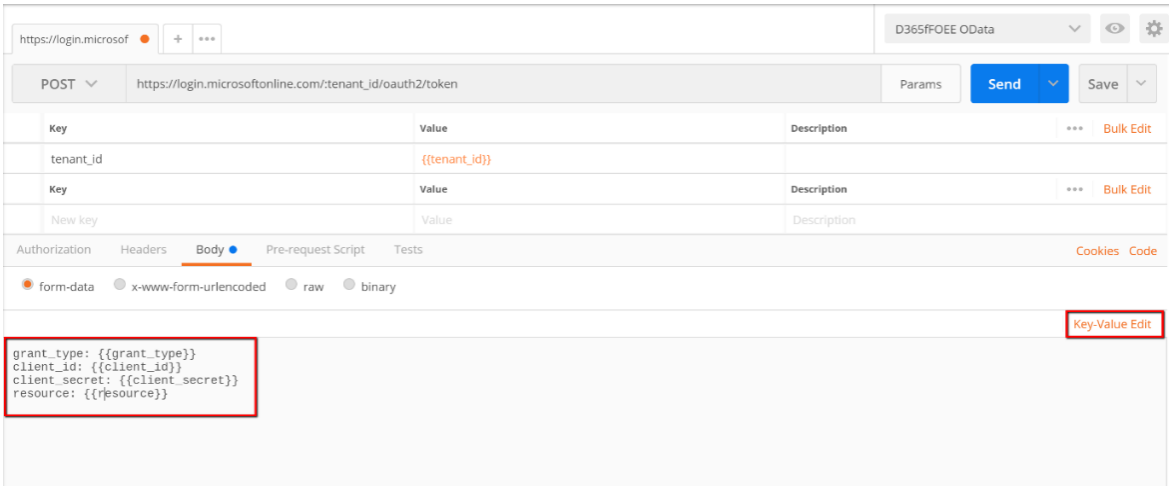
KEY	VALUE
client_secret	The secret key that you generated during application registration during the setup of prerequisites
grant_type	client_credentials
resource	The base URL of the instance without the trailing '/'

5. To verify that the key-value pairs can be parsed correctly, select **Key-Value Edit**, and review the results.
6. Close the environment page.
7. In the field to the left of the gear and eye buttons, select the new or updated environment.
8. To retrieve an Azure AD token, create a POST request that has a URL in the format `https://login.microsoftonline.com/[tenant ID]/oauth2/token`.

   You can use a URL parameter that refers to the **tenant_id** environment variable, such as `https://login.microsoftonline.com/:tenant_id/oauth2/token`.



9. On the **Body** tab, add body elements as request parameters that refer to the environment variables that you created earlier. Select **Bulk Edit**, enter the keys from the previous table, enter a colon (:), and then enter the key name again but enclose it in double braces ({{}}). Enter one request parameter per line. For example, enter **grant_type:{{grant_type}}**. Here is an example.

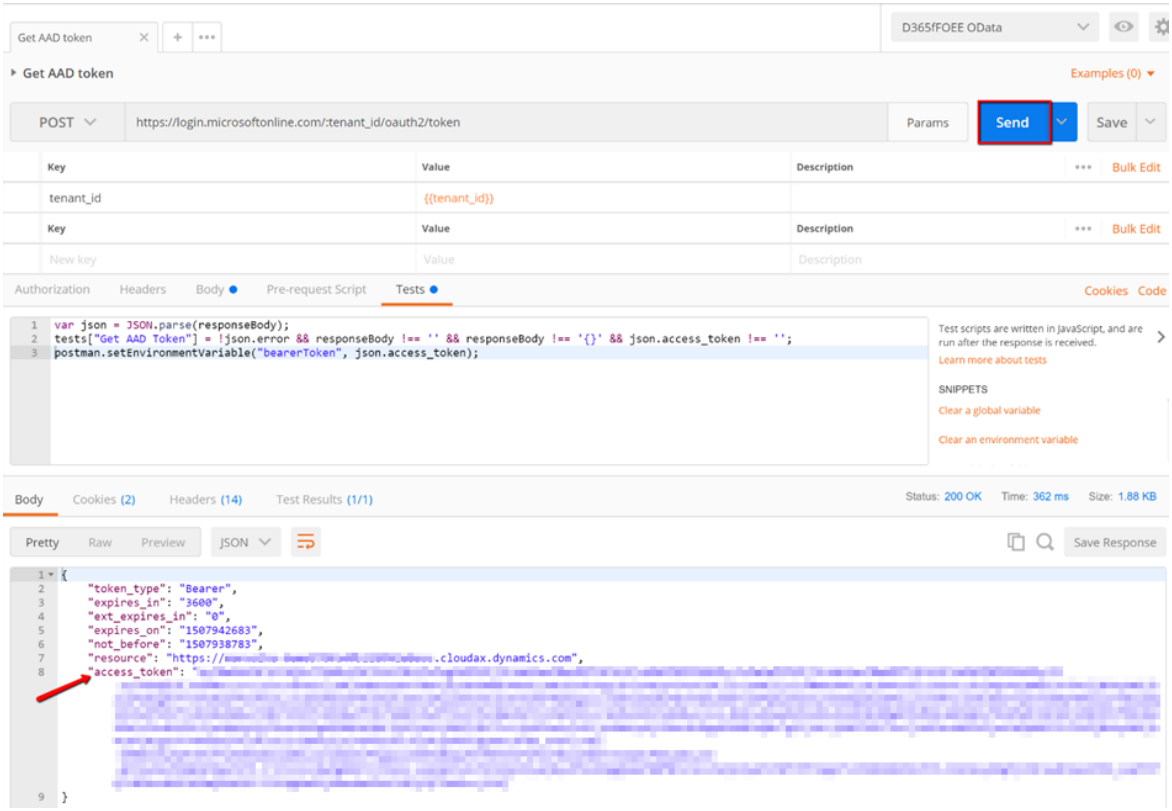


10. On the **Tests** tab, create a test that validates that the response is reasonable, and that stores the returned authorization token in an environment variable. Here is an example.

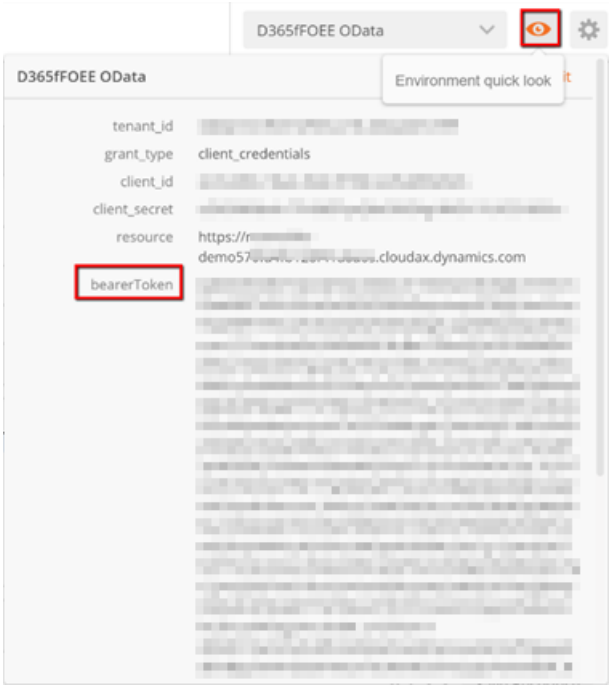
```
var json = JSON.parse(responseBody);
tests["Get Azure AD Token"] = !json.error && responseBody !== '' && responseBody !== '{}' &&
json.access_token !== '';
postman.setEnvironmentVariable("bearerToken", json.access_token);
```

11. Select **Save**, enter a name and collection for the request, and then select **Save** again.

12. Select **Send** to make the authorization request. The **Body** tab should now contain an Azure AD token together with other response details.



13. Because of the test code, the token is now in an environment variable. You can see that the token is an environment variable by selecting the **Environment quick look** button (the eye button).

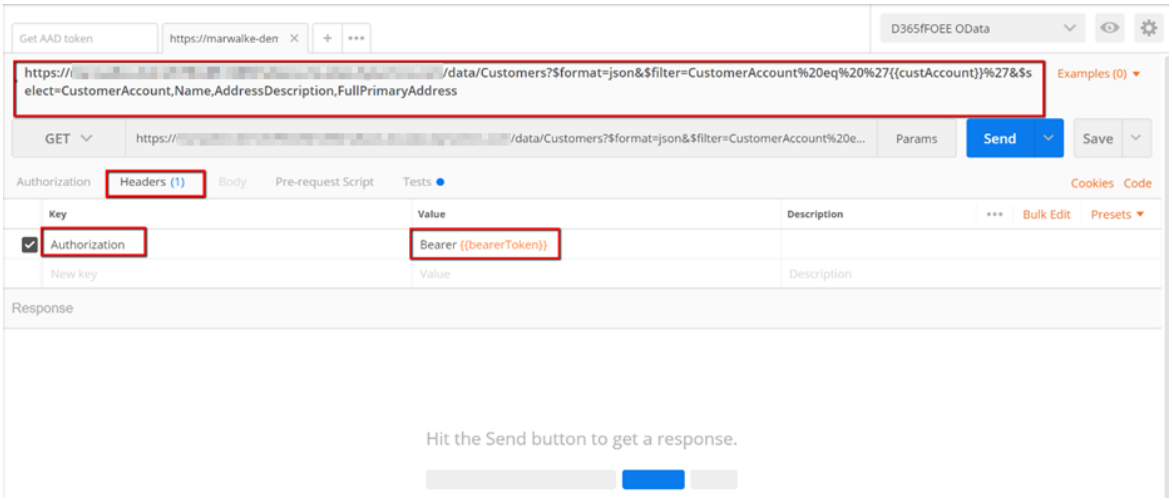


14. Create a request to perform create, read, update, or delete (CRUD) operations on the desired data entity

via the OData service. Create the URL according to your requirements. For more information, see Open Data Protocol (OData). You might find it useful to parameterize the request by using a variable that is stored in the environment, as shown earlier. The following example of a GET query uses a **Customer Account** parameter. The query returns name and address details for the customer account that is specified in the environment variable. Note that special characters must be correctly URL-encoded.

```
https://[Finance and Operations instance URL]/data/Customers?
$format=json&$filter=CustomerAccount%20eq%20%27{{custAccount}}%27&$select=CustomerAccount,Name,Addres
sDescription,FullPrimaryAddress
```

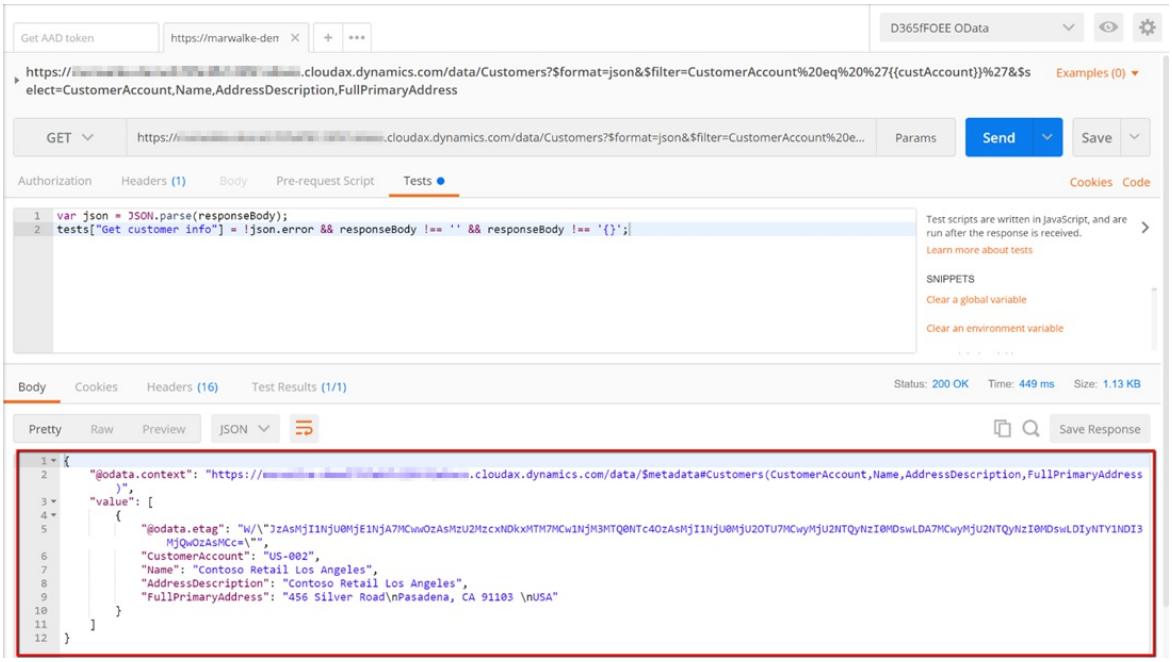
15. Add an Authorization header that refers to the authorization token that was retrieved earlier and stored in the **bearerToken** environment variable. The token must be prefixed by **Bearer** in the header.



16. Create a test to help validate the response. The following example tests that non-empty, JSON-formatted data is returned in the response body.

```
var json = JSON.parse(responseBody);
tests["Get customer info"] = !json.error && responseBody !== '' && responseBody !== '{}';
```

17. Save and send the request, and then verify the result. You must ensure that the user account being used is set to a default company that has data. Alternatively, you can also specify cross-company=true as the query parameter in the OData request.



In our example, we have now successfully authenticated and then used the OData service to read a customer record.

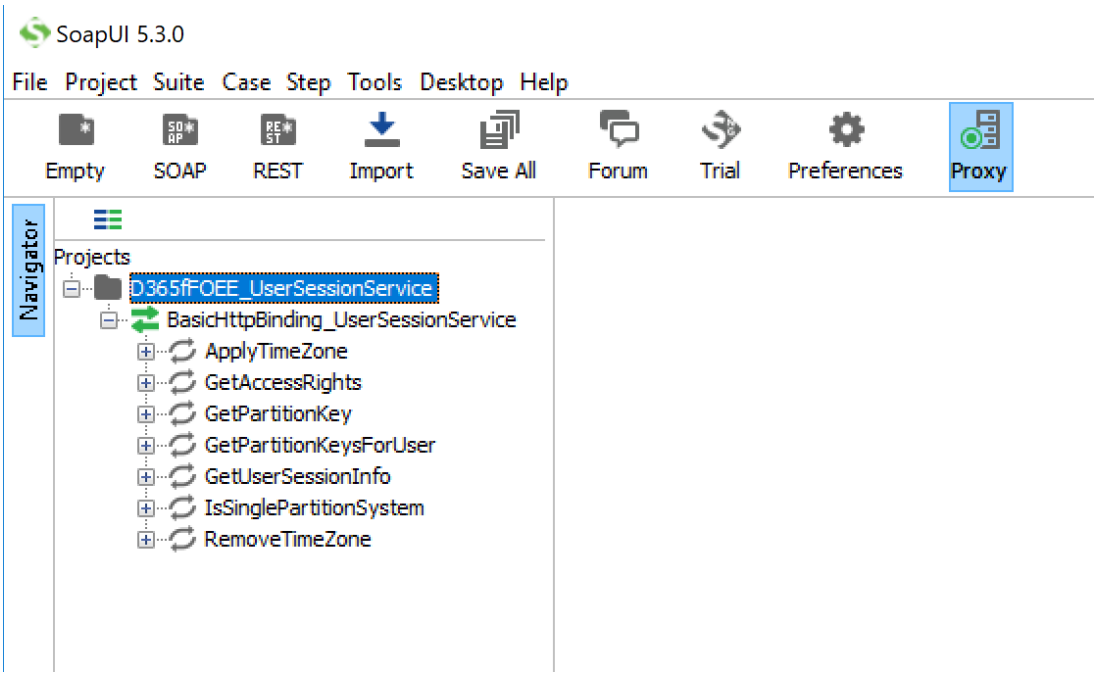
Query the SOAP custom service in your application by using SoapUI

SoapUI (https://www.soapui.org/) is a tool that is often used to interact with SOAP and REST web services in scenarios that involve API development and testing. This procedure isn't an endorsement of SoapUI, and other similar tools are available. However, we are using SoapUI to illustrate the concepts and messages that are involved when you use OAuth to authenticate with Azure AD, and then make SOAP requests to and receive responses.

1. Start SoapUI, and select the **SOAP** button to create a project.
2. Complete the information for the project:
   - In the **Project Name** field, enter a name for the project.
   - In the **Initial WSDL** field, enter the service address, and add the suffix **?wsdl**. (The service address should be in the format [Finance and Operations instance base URL]/soap/services/[service group name].) For more information, see the Services home page.

     For example, we are querying the user session service at the URL `https://[Finance and Operations base URL]/soap/services/UserSessionService?wsdl`.
   - Select the **Create sample requests for all operations?** check box.

     Because you selected to create sample requests, one sample request is created for each service operation that is available.



3. Right-click the new project, and then select **New TestSuite** to create a test suite. This test suite will generate a POST request for an Azure AD authorization token.
4. Right-click the test suite, and then select **New TestCase**.
5. Expand the test case, right-click **Test Steps**, select **Add Step**, and then select **HTTP Request**.
6. Enter a name for the request, and then select **OK**.
7. Enter a name for the test step. The endpoint that you should use for the POST request is

```
[https://login.microsoftonline.com/[tenant_id]/oauth2/token]
(https://login.microsoftonline.com/%5btenant_id%5d/oauth2/token)
```

.

8. Use the plus sign (**+**) button next to **Parameters** to add the following values.

PARAMETER	VALUE
grant_type	client_credentials
client_id	The application ID from the Azure AD application registration
client_secret	The secret key value from the Azure AD application registration
resource	The URL of the instance without the trailing '/'

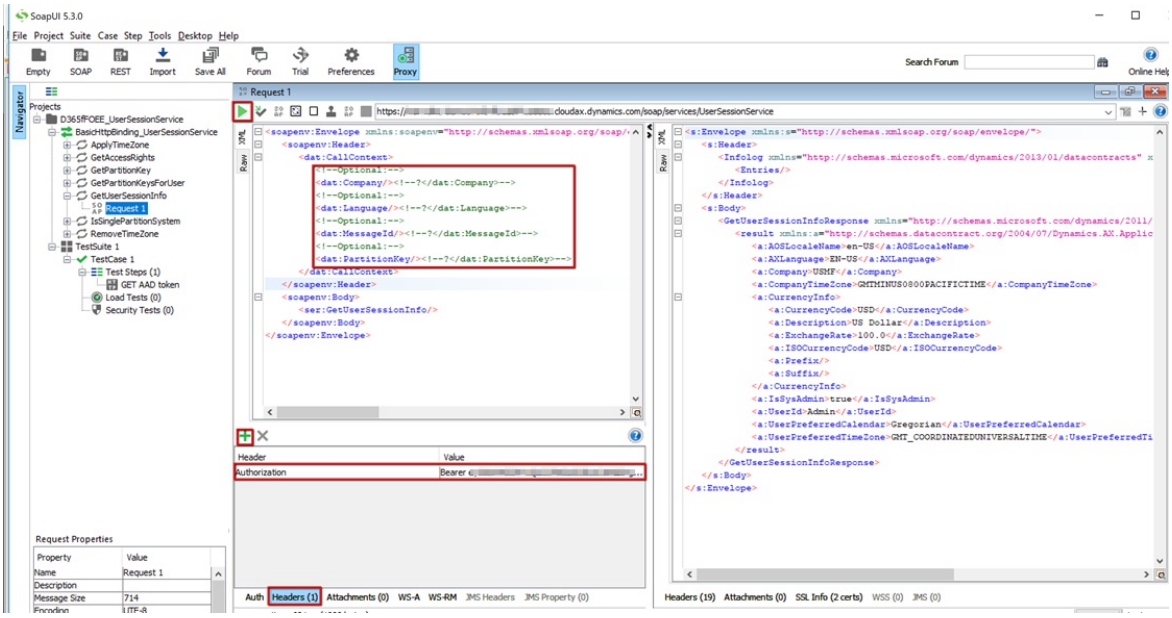
9. To make sure that the parameters are in the POST body, select **Post QueryString**, and then select **Play**. An access token should be returned in the response pane. The values will be most readable if you use the **JSON response** tab. Copy the access token so that you can use it in the authorization header of subsequent requests.

10. Go back to the first request node under the **GetUserSessionInfo** SOAP sample request. In the request pane on the left, select the plus sign (**+**) button to add a header that is named **Authorization**. Paste the access token into the **Value** field, and add the prefix **Bearer**.

11. The sample requests that SoapUI creates won't work unless you modify them. You must edit the call context and body so that they are consistent with the schema for what you're trying to do.

    For our simple scenario, you can edit the optional call context elements so that they are null-valued. Insert a forward slash (/) before the greater than sign (>) in the opening tags. Then comment out the question marks (?) and the closing tags by using the standard <!--...--> syntax to delimit the start and end of the comments. (Question marks aren't valid content for the XML schema.) Alternatively, you can just delete the question marks (?) so that the context elements are empty.

12. The SOAP request is now ready. Select **Play**, and validate the result on the right.



In our example, we have now successfully authenticated and then queried UserSessionService via SOAP.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Application Connector

2/18/2021 • 3 minutes to read • Edit Online

The application connector allows Microsoft Power Automate, Power Apps, Data Integrator, and Logic Apps to integrate with Finance and Operations. An external application can use the available trigger and actions to integrate with them.

> **IMPORTANT**
>
> The Application Connector cannot be used for integrations with Dynamics 365 Finance + Operations (on-premises) instances.

Prerequisites

We recommend that you read the following topics as a prerequisite to familiarize yourself with connectors before proceeding further

- Connectors
- Data management package REST API
- Open Data Protocol (OData)
- Recurring integrations

Triggers

Business events are exposed using the trigger *When a business event occurs*. For detailed information about business events, refer to Business events in Microsoft Power Automate and Business events.

Actions

This section describes the actions that are available in the connector.

Get a record

This action can be used to fetch a record for a specific data entity from the target instance.

Instance refers to the URL of the target instance of the application to which the connector must connect. The expected value is to enter the URL without the 'https://' prefix or choose one from the drop-down menu. This lists of all the environments that are deployed in the Azure Active Directory tenant for the user account that was used to sign in to the specific client like Power Automate, Power Apps, or Logic App.

Entity name refers to the data entity from which the record must be fetched. The drop-down menu shows the list of data entities from the target environment.

Object ID refers to the primary keys fields that must be specified to uniquely identify the record that must be fetched. The values must be specified as a comma-separated list of values in the order that is defined in the entity.

Create a record

This action can be used to create data records for a data entity.

Instance refers to the URL of the target instance to which the connector must connect. The syntax for this value

is to enter the URL without the 'https://' prefix or choose one from the drop- menu. This lists of all the environments that are deployed in the Azure Active Directory tenant for the user account that was used to sign in to the specific client like Power Automate, Power Apps, or Logic App.

Entity name refers to the data entity in which the record must be created. The dropdown menu shows the list of data entities from the target environment.

Based on the selected data entity, the list of fields displayed will be vary.

Update a record

This action can be used to update an existing data record for a data entity. The usage is the same as the create a record action.

Delete a record

This action can be used to delete an existing data record for a data entity. The usage is the same as the get a record action.

Execute action

This action can be used to invoke methods on a data entity to perform a business action.

Instance refers to the URL of the target instance to which the connector must connect. The syntax for this value is to enter the URL without the 'https://' prefix or choose one from the drop- menu. This lists of all the environments that are deployed in the Azure Active Directory tenant for the user account that was used to sign in to the specific client like Power Automate, Power Apps, or Logic App.

Action refers to the method on the data entity that must be executed. Based on the selected method, the list of fields displayed will be vary. These fields represent the parameters for the selected method.

Get list of entities

This action can be used to get the list of entities for further use in the app that is being developed.

Instance refers to the URL of the target instance to which the connector must connect. The syntax for this value is to enter the URL without the 'https://' prefix or choose one from the drop- menu. This lists of all the environments that are deployed in the Azure Active Directory tenant for the user account that was used to sign in to the specific client like Power Automate, Power Apps, or Logic App.

Data management and integration by using data entities overview

2/18/2021 • 2 minutes to read • Edit Online

This topic provides a brief overview of the mechanics of synchronous and asynchronous integration.

Synchronous services

Synchronous integrations are relatively straightforward. Any entity that has **Is public** enabled is automatically available as a service application programming interface (API) in the following URL: `https://[BaseURL]/Data/<<Data Entity Public Collection Name>>`.

Currently, OData protocol is used to expose endpoints where all public-enabled entities can be interacted with.

Supported protocol: OData V4.0

Data format: JSON

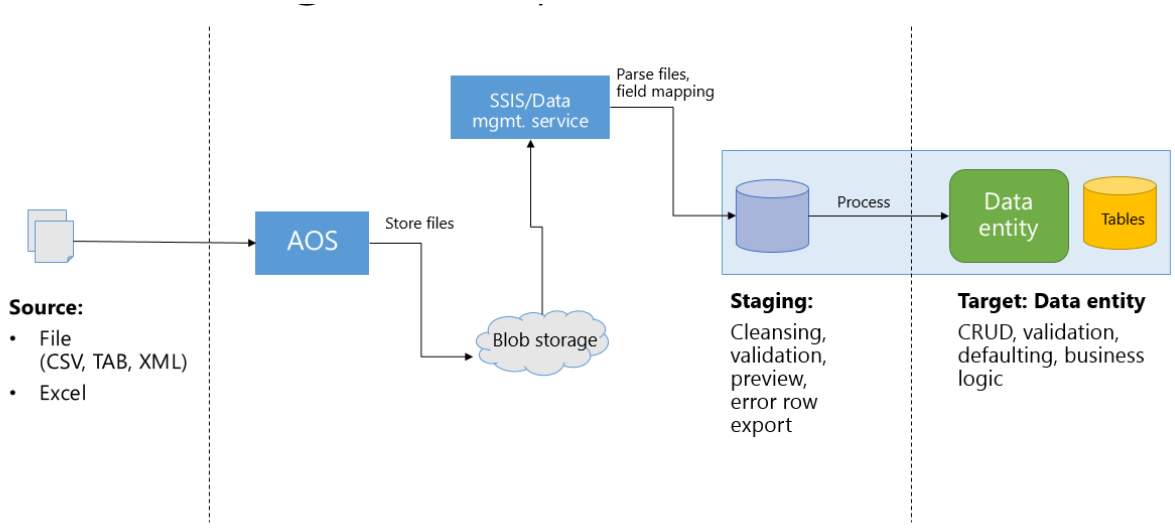
Metadata URL: `https://[BaseURL]/Data/$metadata`

Data import/export and recurring integration scenarios

Integration through the data management platform provides more capabilities and higher throughput for inserting/extracting data through entities. Typically, data goes through three phases in this integration scenario:

- **Source** – These are inbound data files or messages in the queue. Typical data formats include CSV, XML, and tab-delimited.
- **Staging** – These are automatically generated tables that map closely to the data entity. When **Data management enabled** is **true**, staging tables are generated to provide intermediary storage. This enables the framework to do high-volume file parsing, transformation, and some validations.
- **Target** – This is the data entity where data will be imported.

The following diagram shows an inbound flow.



Known limitations in data import/export

When you import text files, string sizes are limited to 32,768 characters. If there is a string larger than this, the imported string will be truncated. This is a limitation in the underlying implementation and is due to SQL Server Integration Services (SSIS).

If you need to import strings that are larger than 32,768 characters, we suggest that you use container entity fields.

For more information, watch the FastTrack Tech Talk video: Dynamics 365 for Operations – Tech Talk: Integration.

Develop entities for data migration

2/18/2021 • 7 minutes to read • Edit Online

This tutorial shows how to develop data entities in Microsoft Visual Studio and then use them for data migration.

This tutorial is broken out into two sections and four exercises. In the first section, you will build a **Project Category** entity in Visual Studio. You will then use this entity to export data. In the second section, you will use **Customer Groups** and **Customers** entities to import multiple sets of files by using the new Data Import/Export Framework.

> **NOTE**
>
> This tutorial is designed to be slightly more challenging than Build and consume data entities. Instead of providing a step-by-step guide, it has scenario exercises and describes the expected outcomes. The assumption is that you've already familiarized yourself with entities.

Prerequisites

This tutorial requires that you access the environment by using Remote Desktop. You must be provisioned as an administrator on the instance.

Base URL

Throughout this tutorial, "base URL" refers to the base URL of the instance.

- In the cloud environment, you obtain the base URL from Microsoft Dynamics Lifecycle Services (LCS).
- On a local virtual machine (VM), the base URL is `https://usnconeboxax1aos.cloud.onebox.dynamics.com`.

Developing an entity in Visual Studio and enabling it for data export

Business problem

You're developing a new solution for a Project module. As part of your implementation, you must represent the data from project categories, so that this data can be imported into the system or exported from it. To accomplish this goal, you will first build an entity for the project category and then use the export functionality to test data extraction.

Exercise 1: Build a Project Category entity

In this exercise, you will build an entity, **Project Category**, that uses the ProjCategory table as its primary data source. This entity has the following properties.

PROPERTY	VALUE
Entity AOT name	ProjectCategoryEntity
Label	Project Categories
Entity category	Reference

PROPERTY	VALUE
Public name	ProjectCategory
Collection name	ProjectCategories
Enable public API	Yes
Enable data management	Yes

The entity also has the following fields:

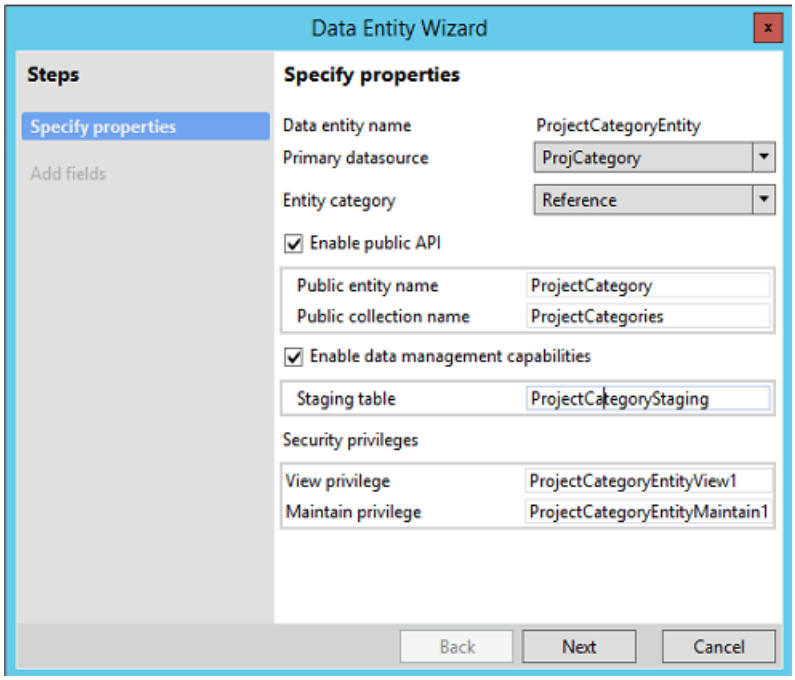
- ActiveInJournals
- CategoryGroup
- Category
- TransactionType
- CategoryName
- Worker
- CustomerPaymentRetention
- IndirectCostComponent
- ItemSalesTaxGroup
- ServiceCode
- Absence

Steps

1. In Visual Studio, create a new application project.
2. In Solution Explorer, select the project, and then right-click **Properties**.
3. Specify the following project properties, and then click **OK**.

PROPERTY	VALUE
Model	Application Suite
Company	USSI
Synchronize Database on build	True

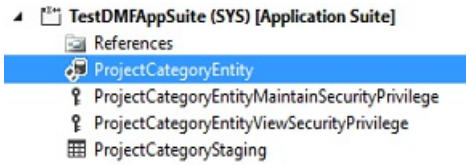
4. From the project, right-click **Add** > **New item**.
5. Select **Data Model** > **Data Entity** as your new item.
6. Enter a name, and then click **Add** to start the **Data Entity** wizard.
7. On the first page of the wizard, specify the set of properties for the entity by using the table earlier in this exercise. Then click **Next**.



8. On the next page, add fields from the primary data source. Make sure that each field name reflects the public contract (see the table earlier in this exercise). To use the field's label as the field name, select the **Convert label to field names** option. Clear the option for any fields that are not required for the entity.
9. Click **Finish** to complete the wizard, and to add the entity and its artifacts to the project.
10. Build your project, so that you can start to use the entity.

Expected outcome

- In Visual Studio, the following artifacts will appear in the project after you've completed the **Data Entity** wizard.



- Right-click the data entity, and then select **Open table browser**.

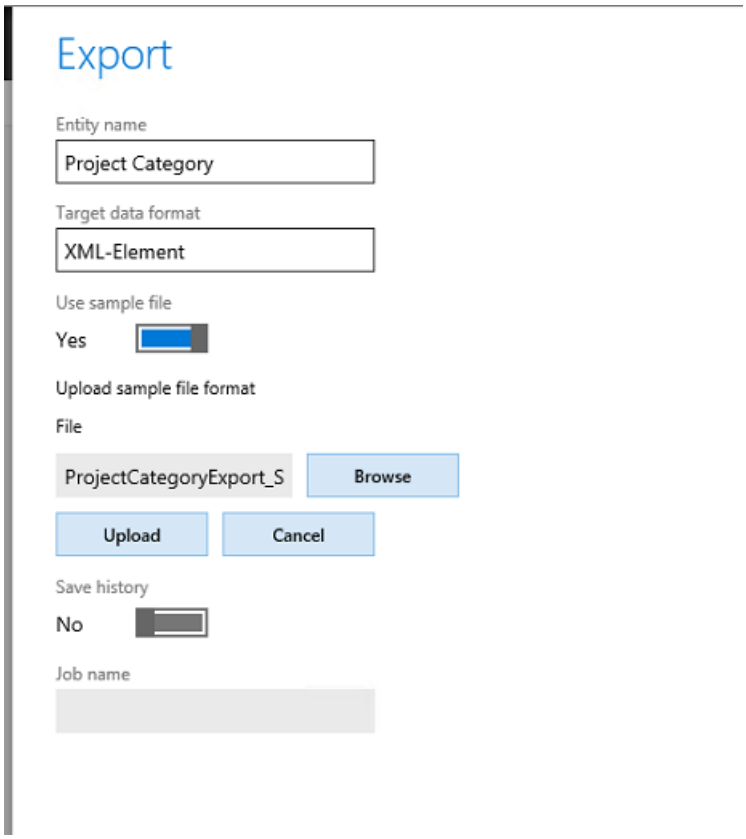
> **NOTE**
>
> Make sure that your company is set to **USSI**.

Exercise 2: Export a limited set of data by using a sample file mapping and filters

In this exercise, you will use the **Project Category** entity that you just built to export data. To export only a subset of the data, you will use a sample file mapping and filters. The exported data will be in XML format.

Steps

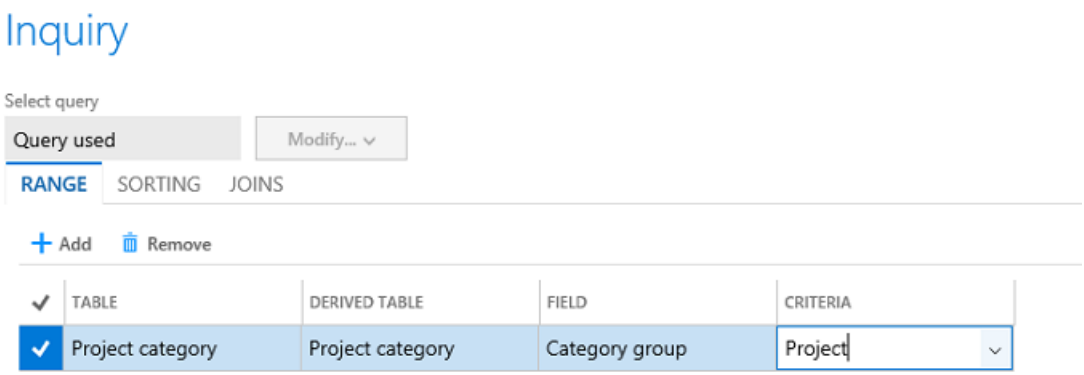
1. After you've finished building the **Project Category** entity, start the client.
2. Change the company to **USSI**.
3. In the **Data management** workspace, click **Export** to begin data extraction.
4. Enter the export details, such as entity name and target data format.



Use the following file as the sample file format for XML: ProjectCategoryExport_Sample.

Open this file in a text editor, and save it as an XML file. If the sample file mapping isn't valid, there is an incorrect field name in the entity. Fix either the entity or the sample file to continue.

5. Click **Filter**, and then specify **Project** as the filter criterion, so that only limited data is exported.



6. In the **Export** dialog box, click **OK**.

Expected outcome

- Fifteen records are successfully exported.
- The output is similar to the following file: ProjectCategoryExport_Output. (Open the file in a text editor to verify this outcome.)

Migrating data in multiple files by using the Data Import/Export Framework

Business problem

You're implementing a new environment. As part of this implementation, you want to migrate some legacy customer data. The data is contained in two sets of files, each of which has data for the **Customers** and **Customer Groups** entities. This migration is slightly complex, because some columns in the data files don't

map directly to the entities. Additionally, the files have validation errors that must be corrected during the import process.

Exercise 3: Create a data project and import multiple files

In this exercise, you will import two files into the **USRT** company by using the new Data Import/Export Framework. These files need to be imported in sequence by using a single data project. The **Customers** entity has a reference to the **Customer Groups** entity. Because the Customers1 file doesn't map correctly to the **Customers** entity, you will receive an error when you upload the file. Therefore, to complete the import process, you will have to provide the correct column mappings for the **Customers** entity.

Steps

1. Open the following files in Microsoft Excel, and save them as CSV files in your local directory:
   - Customers1
   - CustomerGroups1
2. In the client, change the company to **USRT**.
3. From the **User** dashboard, open the **Data Management** workspace.
4. Click **Import** to configure a new data project.
5. Enter the project details, such as the name and file format.
6. For each file, select an entity, and then upload the data file.
7. Because the Customer1.csv file doesn't map correctly to the **Customers** entity, you will receive an error when you upload the file. After the file is uploaded, click **View mappings** to fix the column mappings for the **Customers** entity.

   > **TIP**
   >
   > **CustomerAccount** is required in the entity during import. It is mapped from **AccountNum** in the source file. Address fields are optional for the import.

8. When you've finished, click the **Back** button in your browser to return to the data project.
9. On the **Data Project** page, click **Import now**.

Expected outcome

Four updates and 23 inserts are successfully imported and the **Execution summary** page shows the results.

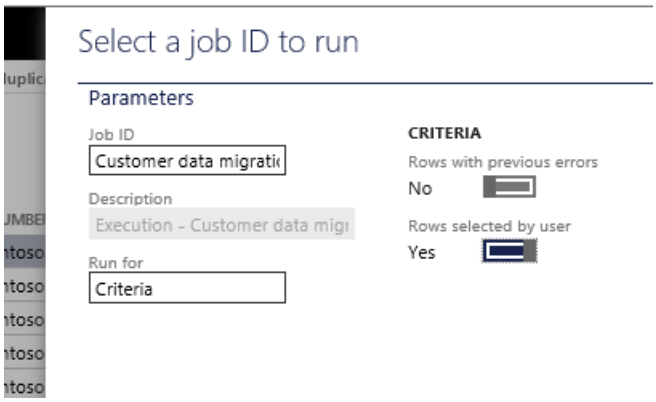
Exercise 4: Re-import by using an existing data project and manage data in staging

In this exercise, you will use a new set of files to import data through the existing data project. Customers2 contains new and updated data for the **Customers** entity. CustomerGroups2 contains updated data for the **Customer Groups** entity. Customers2 contains some error records. You will fix these errors in staging, validate the data, and then push it to the target to complete the import.

Steps

1. In the **Data management** workspace, select the existing data project, and the click **Re-import**. By using the re-import functionality, you can preserve your previous settings for the data project and use new files for the import. However, if you click **Reload data project** and upload new files instead, the previous mappings will be overridden.
2. Open the following files in Excel, and save them as CSV files in your local directory:
   - Customers2
   - CustomerGroups2

3. Upload the new files for each entity, and then click **Import now**.
4. On the **Status** page, click **View execution log** to investigate the errors.
5. On the **Status** page, click **View staging** to view the data in a staging table. This view will also show records that have errors.
6. Click **Edit** to fix records that have errors. (The **Customer Group** value for records DM10221 and DM1022 isn't valid.)
7. Select the records that you fixed, and then click **Validate**. Refresh the page to verify that the status of the records is **Validated**.
8. Click **Copy data to target**.
9. In the **Select a job ID to run** dialog box, in the **Run for** field, select **Criteria**, and set **Row selected by user** to **Yes**. Then click **OK**.



10. On the **Target data execution** page, click **Run**.
11. When the run is completed, refresh the page to see the latest staging status.

Expected outcome

- On the first try, the import succeeds for the **Customer Groups** entity and partially succeeds for the **Customers** entity.
- The **Execution summary** page shows that five records were created, three records were updated, and two records have errors.
- In the staging view, two records have errors.
- After you fix the records and run the import again, the staging view shows that all records are completed.

Develop composite data entities

2/18/2021 • 4 minutes to read • Edit Online

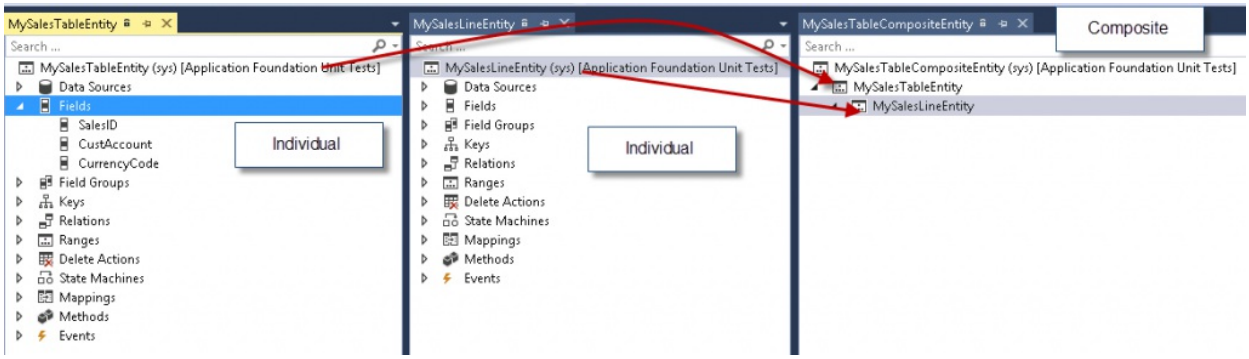
A composite entity is a concept that allows you to build a single entity by leveraging multiple entities that are related to each other.

What is a composite entity?

Composite entity is concept that allows you to build a single entity by leveraging multiple entities that are related to each other. The concept is heavily used in scenarios where an entity can be represented as a single document, like Sales header/line, Invoice header/line and Vendor Catalog. This concept is applicable in asynchronous integration scenarios rather than synchronous OData scenarios, and it will only be supported from a data management platform. There is no programmatic interface for composite entities in X++. It is only supported for a data management platform that is part of XML file-based imports/exports.

Example

Sales Header and Sales Line are two different entities in the system. In case the customer requirement suggests that header and lines are part of a single document, then these two entities can be merged as a composite entity. Sample sales order entity: The composite entity (MySalesTableCompositeEntity) represents a sales orders document which is comprised of Sales Order header entity (MySalesTableEntity) and Sales Order Line entity (MySalesTableLineEntity).



Based on the linked entities, these entities can be exposed as an XML document with embedded element tags for entities. XML is the only way to expose a composite entity in data management.

```
<?xml version="1.0" encoding="utf-8"?>
<Document>

<MySalesTableEntity SalesID="SO1" CurrencyCode="USD" CustAccount="Acc001">
<MySalesLineEntity SalesPrice="2.00" QtyOrdered="10.00" LineAmount="20.00" ItemId="1000" LineNum="1.00"
SalesID="SO1"/>
<MySalesLineEntity SalesPrice="2.00" QtyOrdered="10.00" LineAmount="20.00" ItemId="4401" LineNum="2.00"
SalesID="SO1"/>
</MySalesTableEntity>

<MySalesTableEntity SalesID="SO2" CurrencyCode="USD" CustAccount="Acc002">
<MySalesLineEntity SalesPrice="2.00" QtyOrdered="10.00" LineAmount="20.00" ItemId="4402" LineNum="1.00"
SalesID="SO2"/>
</MySalesTableEntity>

</Document>
```

Each node in the XML represents attributes from an individual entity. For example - <MySalesTableEntity SalesID="SO1" CurrencyCode="USD" CustAccount="Acc001"> SalesId, CurrencyCode and CustAccount are attributes from MySalesTableEntity.

Building the composite entity

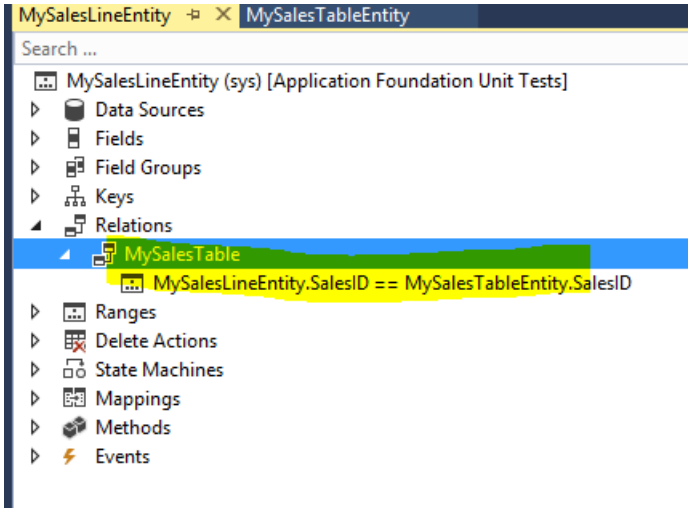
There is a node for composite data entities under Data model. Let's take the example of MySalesTableEntity.

Step 1: Identify and create the individual entities for the composite entity

Make sure that the entities are related to each other. In this example the individual entities are MySalesTableEntity and MySalesLineEntity.

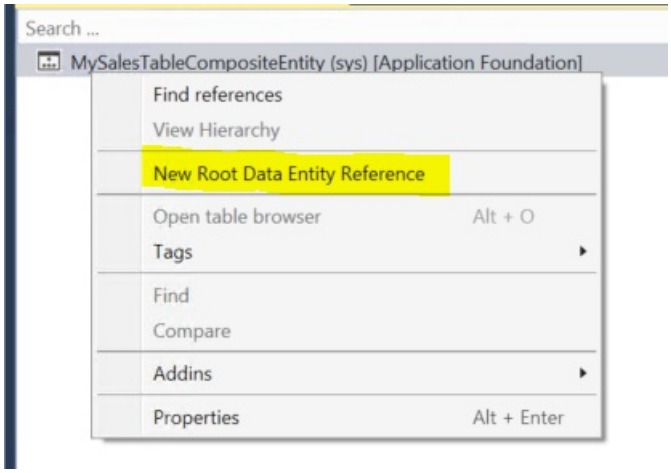
Step 2: Add relations between individual entities

Add a relation to parent entity in the relations node. Example – MySalesLineEntity has relationship to MySalesTableEntity.

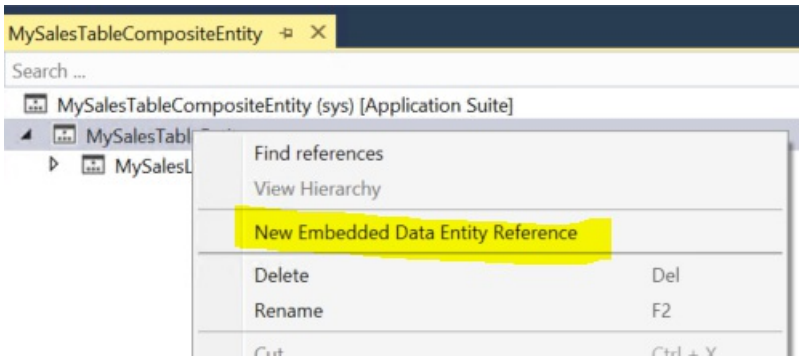


Step 3: Create a new composite entity

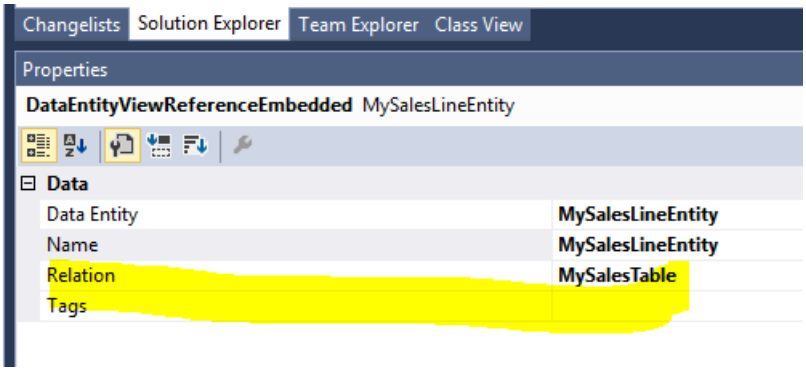
1. Add a new **Dynamics 365** artifact item of type **Composite entity** to the project.
2. In designer mode, right-click the entity and select **New Root Data Entity Reference**.



3. Set the data entity to parent data entity. In this case its MySalesTableEntity.
4. Right-click the parent entity node and select **New Embedded Data Entity Reference**.



5. Set the embedded data entity as the child entity. In this case it is MySalesLineEntity.
6. Set the **Relation** property from the drop-down list on the embedded data entity properties.

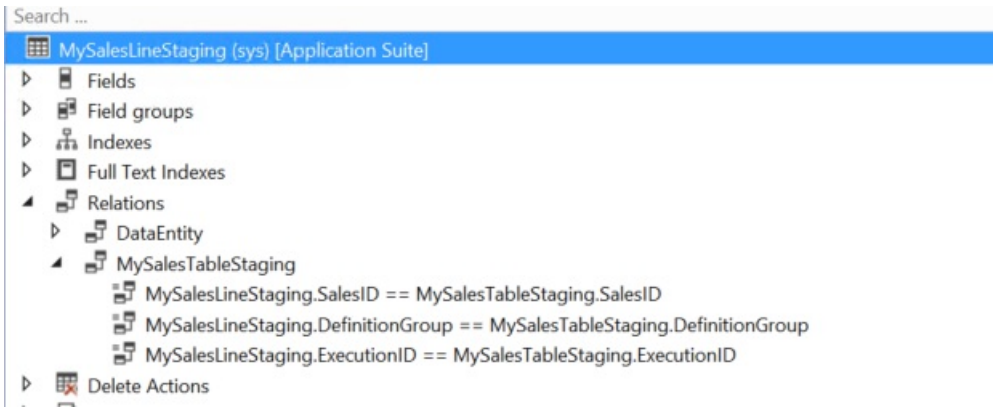


7. Composite entity supports multi-level child entities.

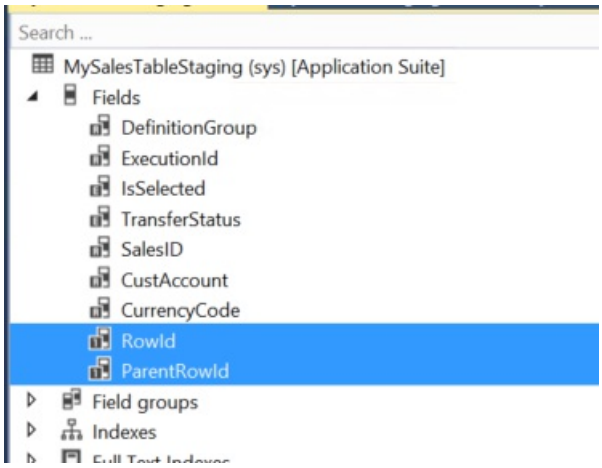
Step 4: Create relationships between staging tables

You need to create relationships between the parent and child entity staging tables based on the natural keys. For example, staging tables for MySalesTable and MySalesLine are linked by SalesID, DefinitionGroup, and ExecutionId.

1. Add a foreign key relation on MySalesLineStaging table.



2. Add two columns, RowId and ParentRowId (type int), on all the staging tables associated with the composite data entity. Refer to SysCompositeHeaderStaging table for the columns properties.



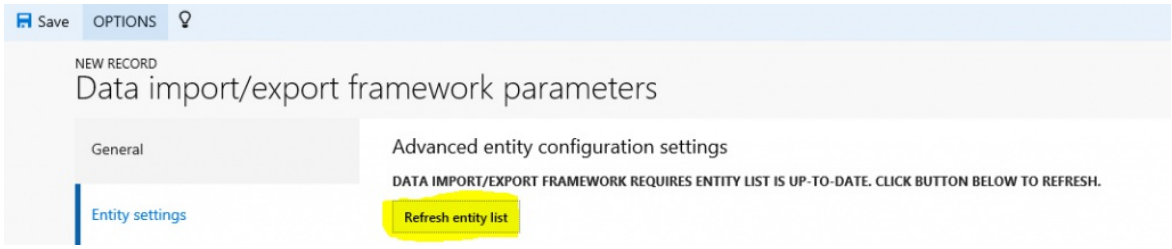
These columns are used to define runtime relationships during the target data movement.

- Create a cluster index on the staging tables which includes RowId, ParentRowid,DefinitionGroup, and ExecutionId. This is for performance reasons.
- Compile and synchronize the artifacts.

Step 5: Set up the metadata for DMFEntity

For local testing the composite entity metadata needs to be refreshed.

1. Go to **DIXF Parameters > Entity settings**. Click **Refresh entity list**.



2. Alternately, you can write the following job to refresh the composite entity list metadata.

```
DMFDataPopulation::refreshCompositeEntityList();
```

3. Execute the job. This refreshes the metadata required for the entity lookup.

> **NOTE**
> Currently this is a workaround. In the future a feature will be enabled to refresh the list at compile/sync time.

Step 6: Test the entity locally

We recommend that you import and export the data as a normal entity from DIXF standard process. Refer to the following the steps for importing and exporting entity.

> **NOTE**
> The source types of XML-Attribute or XML-Element are supported for composite entity. In entity execution parameters, composite entities cannot be imported in parallel using the parallel processing settings.

Import a composite entity

1. Click **Import**.

2. Enter **Name**, **Source data format**, and **Entity name**.
3. The **Source data format** is either xml-attribute or xml-element.
4. Click **Import now**.
5. The number of records created/updated/pending are shown.

Export a composite entity

1. Click **Export**.
2. Enter **Name**, **Source data format**, **and Entity name**.
3. Click **Add entity** and **Export now**.
4. Click **Download package**.

General troubleshooting guidelines

- Issue: The exported composite XML file is not imported. The scenario that produces this is:
  - Export a file for composite entity.
  - Import the same file.
  - Mapping fails and the file is not imported.
- Root cause:
  - Check if the exported file has lines or related child entity information.
  - If there no lines or related child entity information, then the lines will not be mapped during import.
- Resolution:
  - Create a sample file with all of the child entities.
  - Use this file for initial mapping only.
  - When the mapping is successful, import the actual file which does not have the line data into the entity. Use reimport or upload a new file.
  - This should import files with partial data (blank child records), depending on the validity of the records.

Configure financial cross-company data sharing

2/18/2021 • 2 minutes to read • Edit Online

This procedure shows how to configure, enable, validate, and resolve conflicts when sharing data between companies. It uses the USMF company and the Financial data sharing template.

This task guide requires Dynamics AX platform 7.1 or later.

1. Go to **Navigation pane > Modules > System administration > Workspaces > Data management**.
2. Click **Import**.
3. In the **Name** field, type a value.
4. In the **Source data format** field, select the 'Package' type. Click **Upload**. Navigate to the location of the Financial data sharing template package file and select that file.
5. Click **Save**.
6. In the list, mark the selected row. Select the data sharing policy that was just created.
7. Click **Import**.
8. Click **Close**.
9. Refresh the page.
10. Close the page.
11. Close the page.
12. Close the page.
13. Go to **Navigation pane > Modules > System administration > Setup > Configure cross-company data sharing**.
14. In the list, find and select **Payment days**.
15. Click **Add**.
16. In the list, mark the selected row.
17. In the **Company** field, type 'USSI'.
18. Click **Add**.
19. In the **Company** field, type 'USMF'.
20. Click **Save**.
21. Click **Enable**.
22. Click **Yes**.
23. Click **Find sharing issues**.
24. Click **Yes**.
25. Click **Update field value**.
26. Click **Use value from company 1**.
27. Refresh the page.
28. Close the page.

Create a record template to facilitate data entry

2/18/2021 • 2 minutes to read • Edit Online

This topic demonstrates how to create a record template so that field values that are used often do not have to be entered explicitly for each new record. In this procedure, you'll create a new record for new laptops that should be added to your fixed assets. This procedure uses the USMF sample company.

1. In the navigation pane, go to **Modules > Fixed assets > Fixed assets > Fixed assets**.
2. Select **New**.
3. In the **Fixed asset group** field, enter or select a value.
4. In the **Name** field, type a value. For example, enter **Corporate lead laptop**.
5. In the **Search name** field, type a value. For example, enter **laptop**.
6. Expand the **Technical information** section.
7. In the **Make**, **Model**, and **Model year** fields, type values.
8. On the Action Pane, select **Options**.
9. Select **Record info**.
10. Select **User template**.
11. In the **Name** field, type a value.
12. In the **Description** field, type a value.
13. Select **OK**.
14. Select **Close**.

Use record template to create a new record

2/18/2021 • 2 minutes to read • Edit Online

This procedure shows how to use a previously defined record template to create a new record. To complete this procedure, you must first complete the "Create a record template to facilitate data entry" procedure.

This procedure uses the USMF company.

1. In the **Navigation pane**, go to **Fixed assets > Fixed assets > Fixed assets**.
2. Click **New**. You will be prompted to select a template. Select the one that corresponds to your business need.
3. In the list, find and select the desired record.
4. Click **OK**.

Data integration using Dataverse overview

2/18/2021 • 2 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

You can use Dataverse to enable the flow of data between Finance and Operations apps and Dynamics 365 Sales. For example, customer information in Sales can flow to Finance and Operations apps. You don't have to manually move the data or use a third-party data integration tool.

For more information about Dataverse data integration, see Integrate data into Dataverse for Apps in the Microsoft Power Apps documentation.

For an example of using Dataverse, see Prospect to cash.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Dual-write home page

2/18/2021 • 2 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

These topics describe dual-write integration.

- What is dual-write?
  - Top reasons to use dual-write
  - What does dual-write mean for developers and architects of customer engagement app?
- What's new or changed in dual-write
- Frequently asked questions

Dual-write setup

- System requirements for dual-write
- Guidance for how to set up dual-write
- Considerations for initial synchronization
- Dual-write setup from Lifecycle Services
- Enable dual-write for existing Finance and Operations apps
  - Enable dual-write for existing Finance and Operations apps
  - System requirements and prerequisites
  - How to use the dual-write wizard to link your environments
  - Enable table map for dual-write
- Currency data-type migration for dual-write
- Set up the mapping for the sales order status columns
- Filter intercompany orders to avoid synchronizing Orders and OrderLines

Managing dual-write after setup

- Customize table and column mappings
- Handling multiple table maps
- Edit a legal table after dual-write setup
- Error management and alert notifications
- Application lifecycle management

Mapping concepts between apps

These topics describe mapping between concepts in Finance and Operations applications and concepts in model-driven apps in Microsoft Dynamics 365.

- Integrated customer master
- Integrated vendor master
  - Switch between vendor designs
- Customer loyalty cards and reward points
- Unified product experience
  - Integrated sites and warehouses
- Company concept in Dataverse
  - Initialize company data
- Organization hierarchy awareness
- Access to finance and tax reference data
  - Integrated ledger
  - Integrated tax master
- Integrate procurement in Supply Chain Management with Field Service
- Sync on-demand with the Supply Chain Management price engine
- Prospect to cash in dual-write
- In-house assets for servicing
- Integrated worker, job, and position
- Onhand inventory availability

Support

- Support for Field Service and Project Service Automation solutions
- Migrate Prospect to cash data from Data Integrator to dual-write

Troubleshooting

- Verify dual-write configuration in Finance and Operations apps and Dataverse
- Troubleshoot issues during initial setup
- Troubleshoot issues during initial synchronization
- Troubleshoot dual-write issues in Finance and Operations apps
- Troubleshoot live synchronization issues
- Troubleshoot issues related to solution awareness
- Troubleshoot issues from upgrades of Finance and Operations apps
- General troubleshooting

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

What's new or changed in dual-write

2/18/2021 • 7 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

Dual-write is an out-of-box infrastructure that provides near-real-time interaction between customer engagement apps in Microsoft Dynamics 365 and Finance and Operations apps. To get started with dual-write, see the Dual-write home page.

Check out the latest information about dual-write features and changes in the release plans.

- Data in Dataverse – Phase 1
- Data in Dataverse – phase 1 & 2
- Finance and Operations data in Dataverse – Phase 3

January 2021 release

The January 2021 release of the Dual-write application orchestration solution version 2.2.1.30 is based on Dual-write core solution version 10.0.24 and version 10.0.14 of Finance and Operations apps.

This release contains the features and bug fixes listed in the following table.

FEATURE	DESCRIPTION	STATUS
Bug fix	French-localized strings in the user interface exceeded the maximum limit of 100 characters.	General availability
Bug fix	Error while starting the Dataverse released distinct products map.	General availability

The January 2021 release of the Dual-write application orchestration solution version 2.2.1.23 is based on Dual-write core solution version 10.0.24 and version 10.0.14 of Finance and Operations apps.

This release contains the features and bug fixes listed in the following table.

FEATURE	DESCRIPTION	STATUS
Purchase order integration	Integrates purchase order functionality between Dynamics 365 Field Service and Dynamics 365 Supply Chain Management.	General availability
Bug fix	Localization updates.	General availability

FEATURE	DESCRIPTION	STATUS
Bug fix	In customer engagement apps, on the **Contact** form, after you set **Is Sellable** to **Yes** and save the record, the contact is considered a customer who can transact. Because customers are associated with transactions, **Is Sellable** becomes read-only after saving. You can't change it back to **No**.	General availability

December 2020 release

The December 2020 release of the Dual-write core solution (10.0.24) contains the features and bug fixes listed in the following table.

FEATURE	DESCRIPTION	STATUS
Signal repeater service	Enables the dual-write runtime plugin to communicate with the Finance and Operations signal repeater service with authentication support.	General availability

November 2020 release

The November 2020 release of the Dual-write core solution (10.0.23) contains the features and bug fixes listed in the following table.

FEATURE	DESCRIPTION	STATUS
Authentication	Support for new authentication certificate to ensure security.	General availability

October 2020 release

The October 2020 release of the Dual-write application orchestration solution and the Dual-write core solution contains the features and bug fixes listed in the following table.

FEATURE	DESCRIPTION	STATUS
Camel-cased column mappings	Adds support for column mappings with camel-cased navigation properties.	General availability
Bug fix	Fixes the bug where an unrecognized tag configuration would cause dual-write execution to be skipped	General availability

September 2020 release

The September 2020 release of the Dual-write application orchestration solution version 2.0.777.493 is based on Dual-write core solution version 10.0.21.

The September 2020 release contains the features and bug fixes listed in the following table.

FEATURE	DESCRIPTION	STATUS
Lead qualification process in Sales is now company striped	Dynamics 365 Sales users can create a lead, qualify the lead to an opportunity, convert an opportunity into a quote, activate a quote, and create an order. This process was broken in dual-write due to lack of company striping on the **Lead** entity. We implemented company striping on the **Lead** entity, which cascades the company to the underlying **Account** and **Opportunity** tables. Thus the application behavior is restored to support the process. During the **Lead** qualification process, the **Contact** entity isn't company striped. This design supports the **Party** entity model that is due in October 2020. To learn about the **Party** and **GlobalAddressBook** model for dual-write, join the dual-write Yammer group.	General availability
Map state transitions from **Order** to **SalesOrder**	The **Order** form in Dynamics 365 Sales is always set to **Active**. To create state transitions from **Order** in Dynamics 365 Sales to **SalesOrder** in Dynamics 365 Supply Chain Management, we introduced the **ProcessingStatus** column.	General availability
Money to decimal data type conversion	Dataverse environments are limited to 4 decimal places for currency and 10 decimal places for exchange rates. Finance and Operations apps support more decimal places than Dataverse. You can now opt in to extend the decimal support in Dataverse to help ensure there's no loss of decimal place data when using dual-write.	General availability
Security role for company and currency exchange	Company and currency exchange tables are global in nature and all dual-write users require read access to these 2 tables. To simplify the experience, we've added a new security role named **dual-write app user**. Each dual-write user must be added to this security role.	General availability
Security role for setup	Adds the **Dual-write Runtime User** security role. This role allows non-administrator users to create rows that are set up for dual-write. This feature is part of Dual-write core solution 10.0.21.	General availability
Tracing	Internal column added for use in tracing. This feature is part of Dual-write core solution 10.0.21.	General availability

FEATURE	DESCRIPTION	STATUS
Bug fix	Fixes issues where dual-write fails because of a mismatch between the plugin and the destination environments. This fix is part of Dual-write core solution 10.0.21.	General availability
Bug fix	Support to ensure that unused plugins are deleted. This fix is part of Dual-write core solution 10.0.21.	General availability

August 2020 release

The August 2020 release of the dual-write orchestration package contains the features and bug fixes listed in the following table.

FEATURE	DESCRIPTION	STATUS
Manage multiple table maps	As part of day-to-day operations, you might need to bulk handle table maps. For example, you might want to simultaneously enable or pause a set of table maps. Instead of doing this one-by-one, which is cumbersome and time consuming, you can now enable, pause, resume, or stop more than one table map at the same time in the dual-write list page.	General availability
Bug fix	Fixes issues where rows would be skipped in certain cases during project execution. This fix is part of Dual-write core solution version 10.0.19.	General availability

June 2020 release

The June 2020 release of the dual-write orchestration package contains the features and bug fixes listed in the following table.

FEATURE	DESCRIPTION	STATUS
Edit legal entity after setup	The company or legal entity list isn't static and is constantly changing. You might need to add new companies, for example, during a phased rollout or acquisition. Previously, you couldn't add a company or legal entity without system downtime. During this downtime, you would have to unlink and relink your environment. That can be expensive, especially if you have pre-existing data. With this feature, you can add a company in a live environment without having to unlink and relink.	General availability

May 2020 release

The May 2020 release of the dual-write orchestration package (version 2.0.777.353) contains the features and bug fixes listed in the following table.

FEATURE	DESCRIPTION	STATUS
Look up on-hand inventory	Ability to look up on-hand inventory and available-to-promise dates on forms in customer engagement apps.	General availability
Unit conversions	When unit conversions occur in a Finance and Operations app at the quote line and order line, the customer engagement app honors the unit conversions and reflects the respective changes to unit and price in the customer engagement app quote detail and order detail.	General availability
Currency change restriction	When you try to change the currency in a Finance and Operations app for an existing quote or order, the change fails.	General availability
Parity in **Account** and **Contact** forms	Bring attribute parity in **Account** and **Contact** forms in customer engagement apps for B2B and B2C customers.	General availability
No address duplication	Don't duplicate an address in a Finance and Operations app when there's a create or update action on a customer engagement app quote or order.	General availability
SalesTaxGroup support	Support for **SalesTaxGroup** in **Account** and **Contact** forms for business-to-business (B2B) and business-to-consumer (B2C) customers.	General availability
Create sellable contacts	Allow creation of a sellable contact using the **Quick Create: Contact** form in customer engagement apps.	General availability
Quote and order creation	Enable quote and order creation for B2C customers.	General availability
Removal of tenant admin-level consent requirement	Until now, before you could enable dual-write, a tenant admin needed to explicitly give consent to the applications. This wasn't always practical and required additional approval, which can be time consuming. With this feature, we removed this prerequisite and the need for explicitly giving consent to the applications.	General availability

FEATURE	DESCRIPTION	STATUS
Force unlink dual-write environment	Previously, while testing dual-write, you had to disable all the table maps before unlinking a dual-write environment. This seemed cumbersome and sometimes not possible if one of the environments wasn't available. This new feature provides a quick way to unlink your test and trial environments.	General availability

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Dual-write overview

2/18/2021 • 4 minutes to read • Edit Online

NOTE

Effective November 2020:

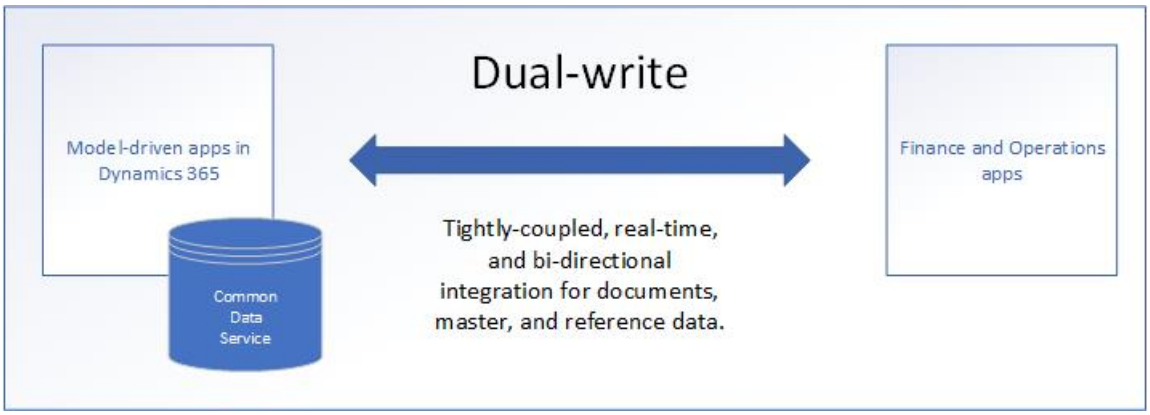
- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

What is dual-write?

Dual-write is an out-of-box infrastructure that provides near-real-time interaction between customer engagement apps and Finance and Operations apps. When data about customers, products, people, and operations flows beyond application boundaries, all departments in an organization are empowered.

Dual-write provides tightly coupled, bidirectional integration between Finance and Operations apps and Dataverse. Any data change in Finance and Operations apps causes writes to Dataverse, and any data change in Dataverse causes writes to Finance and Operations apps. This automated data flow provides an integrated user experience across the apps.



Dual-write has two aspects: an *infrastructure* aspect and an *application* aspect.

Infrastructure

The dual-write infrastructure is extensible and reliable, and includes the following key features:

- Synchronous and bidirectional data flow between applications
- Synchronization, together with play, pause, and catchup modes to support the system during online and offline/asynchronous modes.
- Ability to sync initial data between the applications
- Combined view of activity and error logs for data admins
- Ability to configure custom alerts and thresholds, and to subscribe to notifications
- Intuitive user interface (UI) for filtering and transformations
- Ability to set and view table dependencies and relationships
- Extensibility for both standard and custom tables and maps
- Reliable application lifecycle management

- Out-of-box setup experience for new customers

Application

Dual-write creates a mapping between concepts in Finance and Operations apps and concepts in customer engagement apps. This integration supports the following scenarios:

- Integrated customer master
- Access to customer loyalty cards and reward points
- Unified product mastering experience
- Awareness of organization hierarchy
- Integrated vendor master
- Access to finance and tax reference data
- On-demand price engine experience
- Integrated prospect-to-cash experience
- Ability to serve both in-house assets and customer assets through field agents
- Integrated procure-to-pay experience
- Integrated activities and notes for customer data and documents
- Ability to look up on-hand inventory availability and details
- Project-to-cash experience
- Ability to handle multiple addresses and roles through the party concept
- Single source management for users
- Integrated channels for retailing and marketing
- Visibility into promotions and discounts
- Request-for-service functions
- Streamlined service operations

Top reasons to use dual-write

Dual-write provides data integration across Microsoft Dynamics 365 applications. This robust framework links environments and enables different business applications to work together. Here are the top reasons why you should use dual-write:

- Dual-write provides tightly coupled, near-real-time, and bidirectional integration between Finance and Operations apps and model-driven apps in Dynamics 365. This integration makes Microsoft Dynamics 365 the one-stop shop for all your business solutions. Customers who use Dynamics 365 Finance and Dynamics 365 Supply Chain Management, but who use non-Microsoft solutions for customer relationship management (CRM), are moving toward Dynamics 365 for its dual-write support.
- Data from customers, products, operations, projects, and the Internet of Things (IoT) automatically flows to Dataverse through dual-write. This connection is useful for businesses that are interested in Power Platform expansions.
- The dual-write infrastructure follows the no-code/low-code principle. Minimal engineering effort is required to extend the standard table-to-table maps and to include custom maps.
- Dual-write supports both online mode and offline mode. Microsoft is the only company that offers support for online and offline modes.

What does dual-write mean for developers and architects of customer engagement apps?

Dual-write automates the data flow between Finance and Operations apps and customer engagement apps. Dual-write consists of two AppSource solutions that are installed on Dataverse. The solutions expand the table

schema, plugins, and workflows on Dataverse so that they can scale to ERP size. For a successful implementation, developers and architects of customer engagement apps must understand these changes and collaborate with their counterparts on Finance and Operations apps.

To create parity with Finance and Operations applications, dual-write makes some crucial changes in the Dataverse schema. If you understand the plan, you can avoid some design and development rework in the future.

- When the dual-write AppSource package is installed, Dataverse will have new concepts such as company and party. These concepts help applications built on Dataverse, including Dynamics 365 Sales, Dynamics 365 Marketing, Dynamics 365 Customer Service, and Dynamics 365 Field Service, to interact seamlessly with Finance and Operations apps.
- Activities and notes are unified and expanded to support both C1s (users of the system) and C2s (customers of the system).
- To prevent data loss during currency transmission between Finance and Operations apps and the Dataverse, you'll be able to extend the number of decimal places in the currency data type of customers engagement apps. The feature autotranslates existing rows to the new extended state at the metadata layer. During this process, the currency value is translated to decimal data rather than money data, and the currency value supports 10 decimal places. This feature is opt-in, and organizations that don't need more than 4 decimal places of precision do not need to opt in. For more information, see Currency data-type migration for dual-write.
- Date effectivity will be added to Dataverse. It will support past, present, and future data on the same table.
- Product unit conversions are supported for products, quotes, orders, and invoices.

For more information about upcoming changes, see What's new or changed in dual-write.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

System requirements for dual-write

2/18/2021 • 2 minutes to read • Edit Online

The setup of a dual-write connection has the following requirements:

- Finance and Operations apps that have build version 10.0.9 (10.0.383.20013) (Quality update) and platform update 33 or later
- Customer engagement apps that have platform version 9.1.0000.11732 or later

Dual-write has these limitations:

- You can't run dual-write and the Prospect to cash solution for Data integrator side by side. If you're running the Prospect to cash solution for Data integrator, you must uninstall it.
- Dual-write setup is not supported on trial instances of Finance and Operations apps.
- Dual-write must be used to integrate a single Finance and Operations app instance and a single customer engagement app instance.
- Dual-write currently has a limit of 40 legal entities.
- Dual-write does not support cross-company data sharing.
- Dual-write requires that the Finance and Operations app and the customer engagement app must be in the same Microsoft Azure Active Directory (Azure AD) tenant.
- Dual-write requires that the Finance and Operations app and the customer engagement app must be deployed in the same Microsoft Azure datacenter.
- Dual-write is not triggered by the **doInsert**, **doUpdate**, and **doDelete** events of Finance and Operations apps. Use the **Insert**, **Update**, and **Delete** events in Finance and Operations apps when you want to trigger dual-write.
- Dual-write doesn't support distributed transactions. For example, if the product receipt posting process is cancelled, dual-write might create the product receipt in Dataverse but not create it in Supply Chain Management.

One Version

Future updates of the dual-write solution will be available through One Version.

Guidance for dual-write setup

2/18/2021 • 6 minutes to read • Edit Online

IMPORTANT

Some or all of the functionality noted in this topic is available as part of a preview release. The content and the functionality are subject to change. For more information about preview releases, see Service update availability.

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

You can set up a dual-write connection between a Finance and Operations environment and a Dataverse environment.

- A **Finance and Operations environment** provides the underlying platform for **Finance and Operations apps** (for example, Microsoft Dynamics 365 Finance, Dynamics 365 Supply Chain Management, Dynamics 365 Commerce, and Dynamics 365 Human Resources).
- A **Dataverse environment** provides the underlying platform for **customer engagement apps** (Dynamics 365 Sales, Dynamics 365 Customer Service, Dynamics 365 column Service, Dynamics 365 Marketing, and Dynamics 365 Project Service Automation).

IMPORTANT

The Human Resources module in Dynamics 365 Finance supports dual-write connections, but the Dynamics 365 Human Resources app doesn't.

The setup mechanism varies, depending on your subscription and the environment:

- For new instances of Finance and Operations apps, the setup of a dual-write connection begins in Microsoft Dynamics Lifecycle Services (LCS). If you have a license for Microsoft Power Platform, you will get a new Dataverse environment if your tenant doesn't have one.
- For existing instances Finance and Operations apps, the setup of a dual-write connection begins in the Finance and Operations environment.

Before you start dual-write on an entity, you can run an initial synchronization to handle existing data on both sides: Finance and Operations apps and customer engagement apps. You can skip the initial synchronization if you don't have to sync data between the two environments.

An initial synchronization lets you copy existing data from one app to another bidirectionally. There are several setup scenarios, depending on the environments that you already have and the type of data in them.

The following setup scenarios are supported:

- A new Finance and Operations app instance and a new customer engagement app instance

- A new Finance and Operations app instance and an existing customer engagement app instance
- A new Finance and Operations app instance that has data and a new customer engagement app instance
- A new Finance and Operations app instance that has data and an existing customer engagement app instance
- An existing Finance and Operations app instance and a new customer engagement app instance
- An existing Finance and Operations app instance and an existing customer engagement app instance

A new Finance and Operations app instance and a new customer engagement app instance

To set up a dual-write connection between a new instance of a Finance and Operations app that has no data and a new instance of a customer engagement app, follow the steps in Dual-write setup from Lifecycle Services. When the connection setup is completed, the following actions automatically occur:

- A new, empty Finance and Operations environment is provisioned.
- A new, empty instance of a customer engagement app is provisioned, where the CRM prime solution is installed.
- A dual-write connection is established for DAT company data.
- Table maps are enabled for live synchronization.

Both environments are then ready for live data synchronization.

A new Finance and Operations app instance and an existing customer engagement app instance

To set up a dual-write connection between a new instance of a Finance and Operations app that has no data and an existing instance of a customer engagement app, follow the steps in Dual-write setup from Lifecycle Services. When the connection setup is completed, the following actions automatically occur:

- A new, empty Finance and Operations environment is provisioned.
- A dual-write connection is established for DAT company data.
- Table maps are enabled for live synchronization.

Both environments are then ready for live data synchronization.

To sync the existing Dataverse data to the Finance and Operations app, follow these steps.

1. Create a new company in the Finance and Operations app.
2. Add the company to the dual-write connection setup.
3. Bootstrap the Dataverse data by using a three-letter International Organization for Standardization (ISO) company code.
4. Run the **Initial sync** functionality for the tables that you want to sync data for.

For links to an example and an alternative approach, see the Example section later in this topic.

A new Finance and Operations app instance that has data and a new customer engagement app instance

To set up a dual-write connection between a new instance of a Finance and Operations app that has data and a new instance of a customer engagement app, follow the steps in the A new Finance and Operations app instance and a new customer engagement app instance section earlier in this topic. When the connection setup is completed, if you want to sync the data to the customer engagement app, follow these steps.

1. Open the Finance and Operations app from the LCS page, sign in, and then go to **Data Management** > **Dual-write**.
2. Run the **Initial sync** functionality for the tables that you want to sync data for.

For links to an example and an alternative approach, see the Example section.

A new Finance and Operations app instance that has data and an existing customer engagement app instance

To set up a dual-write connection between a new instance of a Finance and Operations app that has data and an existing instance of a customer engagement app, follow the steps in the A new Finance and Operations app instance and an existing customer engagement app instance section earlier in this topic. When the connection setup is completed, if you want to sync the data to the customer engagement app, follow these steps.

1. Open the Finance and Operations app from the LCS page, sign in, and then go to **Data Management** > **Dual-write**.
2. Run the **Initial sync** functionality for the tables that you want to sync data for.

To sync the existing Dataverse data to the Finance and Operations app, follow these steps.

1. Create a new company in the Finance and Operations app.
2. Add the company to the dual-write connection setup.
3. Bootstrap the Dataverse data by using a three-letter ISO company code.
4. Run the **Initial sync** functionality for the tables that you want to sync data for.

For links to an example and an alternative approach, see the Example section.

An existing Finance and Operations app instance and a new customer engagement app instance

The setup of a dual-write connection between an existing instance of a Finance and Operations app and a new instance of a customer engagement app occurs in the Finance and Operation environment.

1. Set up the connection from the Finance and Operations app.
2. Run the **Initial sync** functionality for the tables that you want to sync data for.

For links to an example and an alternative approach, see the Example section.

An existing Finance and Operations app instance and an existing customer engagement app instance

The setup of a dual-write connection between an existing instance of a Finance and Operations app and an existing instance of a customer engagement app occurs in the Finance and Operation environment.

1. Set up the connection from the Finance and Operations app.
2. To sync the existing Dataverse data to the Finance and Operations app, bootstrap the Dataverse data by using a three-letter ISO company code.
3. Run the **Initial sync** functionality for the tables that you want to sync data for.

For links to an example and an alternative approach, see the Example section.

Example

For an example, see Enabling the Customers V3—Contacts table map

For an alternative approach that is based on data volumes in each entity that must run an initial synchronization, see Considerations for initial synchronization.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Considerations for initial synchronization

2/18/2021 • 7 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

Before you start dual-write on a table, you can run an initial synchronization to handle existing data on both sides: Finance and Operations apps and customer engagement apps. You can skip the initial synchronization if you don't have to sync data between the two environments.

The initial synchronization lets you copy existing data from one app to another bidirectionally, and there are several considerations when you run it. For example, you might have to migrate data before your go-live. In this case, data can be loaded into one side through data migration and then synced to the other side through the initial synchronization.

We recommend that you use the following approach for the initial synchronization:

- **Single-threaded tables**: First migrate data into the Finance and Operations app, and then trigger the initial synchronization to move the data over to Dataverse. Based on lab testing that Microsoft has done, this sequence has better performance than synchronization from Dataverse to Finance and Operations apps.
- **Multi-threaded tables**: First migrate data into Dataverse, and then trigger the initial synchronization to move the data over to the Finance and Operations app.

Constraints

Data migration slow-down with enabled dual-write

If you first activate the map in dual-write and then start to import data, migration performance will be poor. We recommend that you not activate running maps in dual-write until the data migration is completed.

Limit of 500,000 rows per run

The maximum number of rows that is allowed through initial synchronization is 500,000 per run. The limit of 500,000 rows applies to each legal table, because each legal entity runs separately. For more information, see Integrate data into Dataverse. In particular, pay attention to the note that states, "To optimize performance and not overload the apps, we currently limit project executions to 500k rows per execution per project."

If there must be more than 500,000 rows in a run when you the initial synchronization, we recommend that you migrate data into the Finance and Operations app and Dataverse separately, and skip the initial synchronization.

Twenty-four-hour limit

If you're running the initial synchronization from Dataverse to the Finance and Operations app, the import result must be received back from the Finance and Operations app within 24 hours. Otherwise, a time-out occurs. Therefore, if you're syncing lots of data, and the single run takes more than 24 hours, the initial synchronization might fail because of a time-out. For example, an initial synchronization from Dataverse to a Finance and Operations app for the **Customer/Account** table involves 70,000 rows. Therefore, the run might take more

than 24 hours and time out.

Don't run the initial synchronization from Dataverse to a Finance and Operations app for single-threaded tables if the data volume is more than 70,000 rows. Because these tables don't support multi-threading during import, a time-out might occur if the volume is more 70,000 rows. In this situation, you should migrate data into the Finance and Operations app and Dataverse separately, and skip the initial synchronization.

Limit of 40 legal entities while the environments are being linked

Currently, there is a limit of 40 legal entities while the environments are being linked. If you try to enable maps where more than 40 legal entities are linked between the environments, you will receive the following error message:

```
Dual-write failure - Plugin registration failed: [(Unable to get partition map for project DWM-1ae35e60-4bc2-4905-88ea-XXXXX. Error Exceeds the maximum partitions allowed for mapping DWM-1ae35e60-4bc2-4905-88ea-XXXXX)], One or more errors occurred.
```

Initial synchronization isn't currently supported for table maps that have 10 or more lookups

This limitation applies only to the initial synchronization from Dataverse for table maps that have 10 or more lookups. If you run the initial synchronization against a table map that has 10 or more lookups, you might receive the following error message:

```
5 Attempts to get data from https://XXXX.azure-apim.net/apim... Failed
```

As a workaround you can split the initial sync into these steps:

1. Remove some of the lookup columns that are not mandatory from the dual-write table map and bring the number of lookups to 10.
2. After the lookup columns are removed, save the map and do the initial sync.
3. After the initial sync for the first step is successful, add the remaining lookup columns and remove the lookup columns that were synced in first step. Once again make sure the number of lookup columns is 10. Save the map and run the initial sync. Repeat these steps to make sure all the lookup columns are synced.
4. Add all the lookup columns back to the map, save the map and run the map with skip initial sync. This will enable the map for live sync mode.

Five-minute limit for Finance and Operations data export

If you're running the initial synchronization from the Finance and Operations app to Dataverse and the Finance and Operations data export takes more than five minutes, then the initial sync might time out. The time-out can happen if the data table has virtual columns with the `postLoad` method, or the export query isn't optimized (for example, if it has missing indexes).

This type of synchronization is supported in Platform update 37 (PU37) and later. Therefore, you should update your Finance and Operations app to PU37 or later.

Error handling capabilities

Initial synchronization is always a full push

If an individual row fails to be synced, you can't resync only that individual row. The initial synchronization always pushes the whole data set. This behavior is known as a *full push*. If the initial synchronization only partially succeeds, a second synchronization runs for all the rows, not just the rows that failed to be synced during the initial synchronization.

Only the top five errors can be viewed

You can view only the top five errors from the initial synchronization error log.

Known issues

For information about known issues, see Troubleshoot issues during initial synchronization.

Guidance matrix

FINANCE AND OPERATIONS APP INSTANCE	DATAVERSE INSTANCE	HAS DATA TO RUN THE INITIAL SYNCHRONIZATION	DESCRIPTION	MAXIMUM VOLUME IN A TABLE	SINGLE-THREADED OR MULTI-THREADED	APPROACH
New	New	No	A new Finance and Operations app instance and a new customer engagement app instance, where neither app has initial data	Not applicable	Any	• Activate dual-write, and skip the initial synchronization.
New	New	Yes	A new Finance and Operations app instance and a new customer engagement app instance, where one of the apps has migrated data	< 500,000	Single-threaded	1. Migrate data to the Finance and Operations app. 2. Run the initial synchronization.
				< 500,000	Multi-threaded	1. Migrate data to Dataverse. 2. Run the initial synchronization.

FINANCE AND OPERATIONS APP INSTANCE	DATAVERSE INSTANCE	HAS DATA TO RUN THE INITIAL SYNCHRONIZATION	DESCRIPTION	MAXIMUM VOLUME IN A TABLE	SINGLE-THREADED OR MULTI-THREADED	APPROACH
				> 500,000	Any	1. Migrate data to each app outside the initial synchronization. 2. Activate dual-write, and skip the initial synchronization.
New	Existing	Yes	A new Finance and Operations app instance and an existing customer engagement app instance	< 70,000	Single-threaded	1. Create a new company in the Finance and Operations app. 2. Bootstrap Dataverse for the company code. 3. Run the initial synchronization.

FINANCE AND OPERATIONS APP INSTANCE	DATAVERSE INSTANCE	HAS DATA TO RUN THE INITIAL SYNCHRONIZATION	DESCRIPTION	MAXIMUM VOLUME IN A TABLE	SINGLE-THREADED OR MULTI-THREADED	APPROACH
				> 70,000	Single-threaded	1. Create a new company in the Finance and Operations app. 2. Bootstrap Dataverse for the company code. 3. Migrate data to each app outside the initial synchronization. 4. Activate dual-write, and skip the initial synchronization.

FINANCE AND OPERATIONS APP INSTANCE	DATAVERSE INSTANCE	HAS DATA TO RUN THE INITIAL SYNCHRONIZATION	DESCRIPTION	MAXIMUM VOLUME IN A TABLE	SINGLE-THREADED OR MULTI-THREADED	APPROACH
				< 500,000	Multi-threaded	1. Create a new company in the Finance and Operations app. 2. Bootstrap Dataverse for the company code. 3. Run the initial synchronization.

FINANCE AND OPERATIONS APP INSTANCE	DATAVERSE INSTANCE	HAS DATA TO RUN THE INITIAL SYNCHRONIZATION	DESCRIPTION	MAXIMUM VOLUME IN A TABLE	SINGLE-THREADED OR MULTI-THREADED	APPROACH
				> 500,000	Any	1. Create a new company in the Finance and Operations app. 2. Bootstrap Dataverse for the company code. 3. Migrate data to each app outside the initial synchronization. 4. Activate dual-write, and skip the initial synchronization.
Existing	New	Yes	An existing Finance and Operations app instance and a new customer engagement app instance	< 500,000	Any	• Run the initial synchronization.

FINANCE AND OPERATIONS APP INSTANCE	DATAVERSE INSTANCE	HAS DATA TO RUN THE INITIAL SYNCHRONIZATION	DESCRIPTION	MAXIMUM VOLUME IN A TABLE	SINGLE-THREADED OR MULTI-THREADED	APPROACH
				> 500,000	Any	1. Migrate data to each app. 2. Activate dual-write, and skip the initial synchronization.
Existing	Existing	Yes	An existing Finance and Operations app instance and an existing customer engagement app instance	< 70,000	Single-threaded	1. Bootstrap Dataverse for the company code. 2. Run the initial synchronization.

FINANCE AND OPERATIONS APP INSTANCE	DATAVERSE INSTANCE	HAS DATA TO RUN THE INITIAL SYNCHRONIZATION	DESCRIPTION	MAXIMUM VOLUME IN A TABLE	SINGLE-THREADED OR MULTI-THREADED	APPROACH
				> 70,000	Single-threaded	1. Bootstrap Dataverse for the company code. 2. Migrate data to each app outside the initial synchronization. 3. Activate dual-write, and skip the initial synchronization.
				< 500,000	Multi-threaded	1. Bootstrap Dataverse for the company code. 2. Run the initial synchronization.

FINANCE AND OPERATIONS APP INSTANCE	DATAVERSE INSTANCE	HAS DATA TO RUN THE INITIAL SYNCHRONIZATION	DESCRIPTION	MAXIMUM VOLUME IN A TABLE	SINGLE-THREADED OR MULTI-THREADED	APPROACH
				> 500,000	Any	1. Bootstrap Dataverse for the company code. 2. Migrate data to each app outside the initial synchronization. 3. Activate dual-write, and skip the initial synchronization.

Single-threaded tables

- Sales tax codes (msdyn_taxcodes)
- Customers V3 (accounts)
- Vendors V2 (msdyn_vendors)
- Warehouses (msdyn_warehouses)
- Product categories (msdyn_productcategories)
- Employment (cdm_employments)
- Position worker assignments (cdm_positionworkerassignmentmaps)
- Warehouse locations (msdyn_inventorylocations)
- Modes of delivery (msdyn_shipvias)

Dual-write setup from Lifecycle Services

2/18/2021 • 2 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

This topic explains how to set up a dual-write connection between a new Finance and Operations environment and a new Dataverse environment from Microsoft Dynamics Lifecycle Services (LCS).

Prerequisites

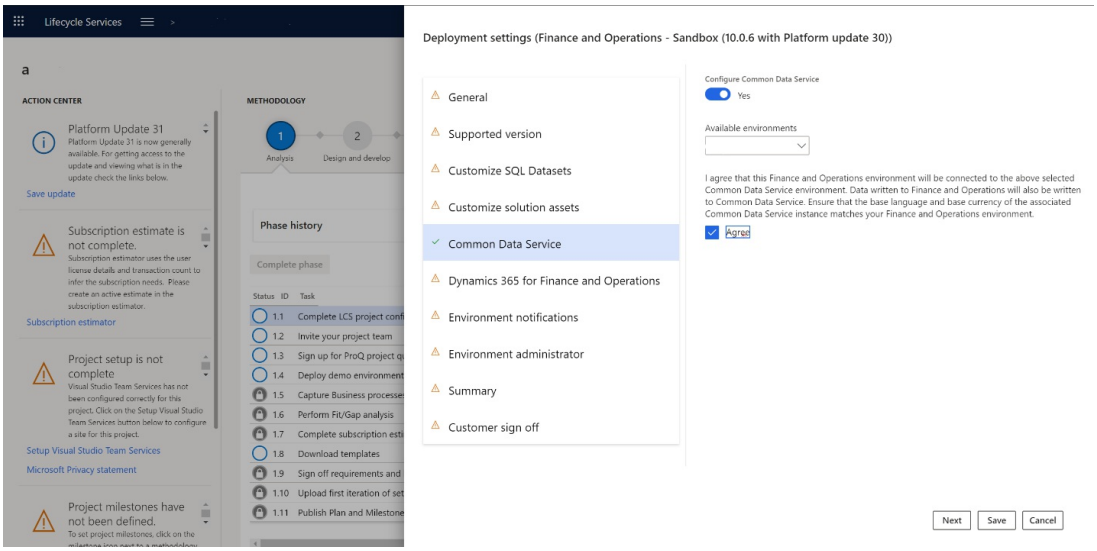
You must be an admin to set up a dual-write connection.

- You must have access to the tenant.
- You must be an admin in both Finance and Operations environments and Dataverse environments.

Set up a dual-write connection

Follow these steps to set up the dual-write connection.

1. In LCS, go to your project.
2. Select **Configure** to deploy a new environment.
3. Select the version.
4. Select the topology. If only one topology is available, it's automatically selected.
5. Complete the first steps in the **Deployment settings** wizard.
6. On the **Dataverse** tab, follow one of these steps:
   - If a Dataverse environment is already provisioned for your tenant, you can select it.
     a. Set the **Configure Dataverse** option to **Yes**.
     b. In the **Available environments** column, select the environment to integrate with your Finance and Operations data. The list includes all environments where you have admin privileges.
     c. Select the **Agree** check box to indicate that you agree to the terms and conditions.



- If your tenant doesn't already have a Dataverse environment, a new environment will be provisioned.

  a. Set the **Configure Dataverse** option to **Yes**.

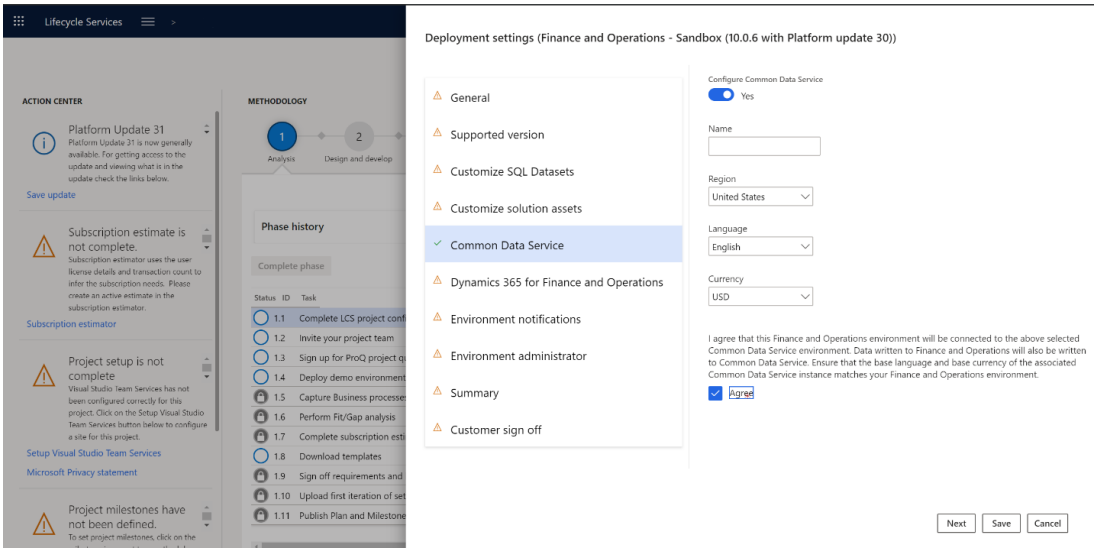
  b. Enter a name for the Dataverse environment.

  c. Select the region to deploy the environment in.

  d. Select the default language and currency for the environment.

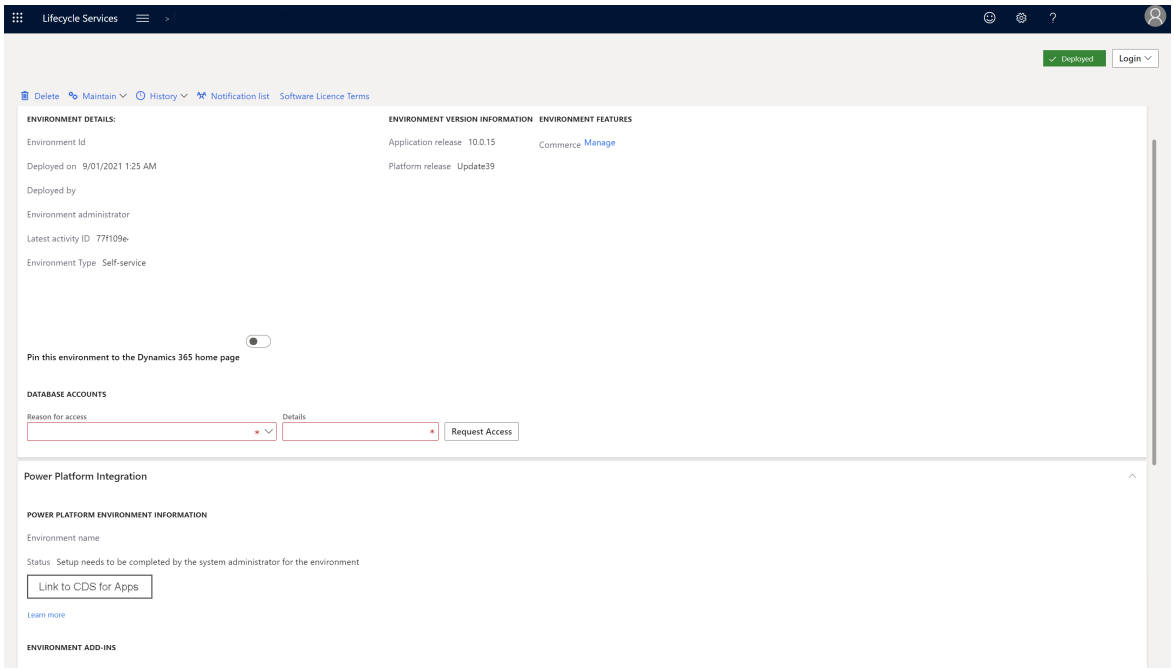
  > **NOTE**
  > You can't change the language and currency later.

  e. Select the **Agree** check box to indicate that you agree to the terms and conditions.



7. Complete the remaining steps in the **Deployment settings** wizard.

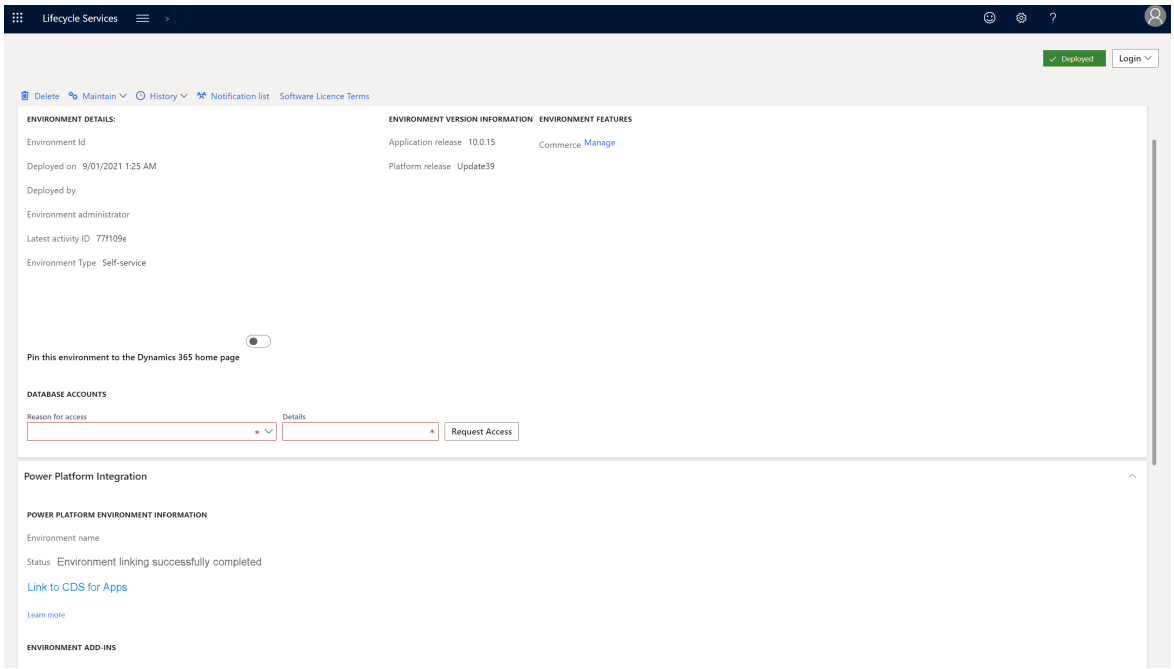
8. After the environment has a status of **Deployed**, open the environment details page. The **Power Platform Integration** section shows the names of the Finance and Operations environment and the Dataverse environment that are linked.



9. An admin of the Finance and Operations environment must sign in to LCS and select **Link to CDS for Apps** to complete the link. The environment details page shows the admin's contact information.

   After the link is completed, the status is updated to **Environment linking successfully completed**.

10. To open the **Data integration** workspace in the Finance and Operations environment and control the templates that are available, select **Link to CDS for Apps**.



NOTE

You can't unlink environments by using LCS. To unlink an environment, open the **Data integration** workspace in the Finance and Operations environment, and then select **Unlink**.

Enable dual-write for existing Finance and Operations apps

2/18/2021 • 2 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

This set of topics provides step-by step instructions that explain how to enable dual-write for existing instances of Finance and Operations apps (Microsoft Dynamics 365 Finance and Dynamics 365 Supply Chain Management), and also for a new or existing Dataverse environment.

Step-by-step instructions to enable dual-write for existing instances of Finance and Operations apps and a new or existing Dataverse environment

The process of enabling dual-write has three parts:

1. Make sure that you meet all the system requirements and complete all the prerequisites.
2. Link your Finance and Operations app environment to Dataverse by using the dual-write wizard.
3. Enable the table maps.

Each part is described in a separate topic.

Next steps

System requirements and prerequisites

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

System requirements and prerequisites

2/18/2021 • 7 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

What regions are available?

Currently, we support dual-write in the following regions:

- Asia
- Australia
- Canada
- Europe
- India
- Japan
- South America
- United Kingdom
- United States

Verify requirements and grant access

Before you enable dual-write, follow these steps to make sure that you meet the minimum system requirements and to grant access to the apps that must connect to each other. The dual-write health check validates the prerequisites as you complete the dual-write wizard to link a Finance and Operations app environment to a Dataverse environment.

You must set **Enable Dynamics 365 apps** to **Yes** when you set up the environment, as shown in the following image. Alternatively, you can choose a model-driven app for Dynamics 365 environment that comes with Dataverse and already has **Enable Dynamics 365 apps** set to **Yes**.

1. Validate the platform update and app version.

   Make sure that your Finance and Operations app environment is running Platform update 33 (app version 10.0.9) or later.

   Related health check result:

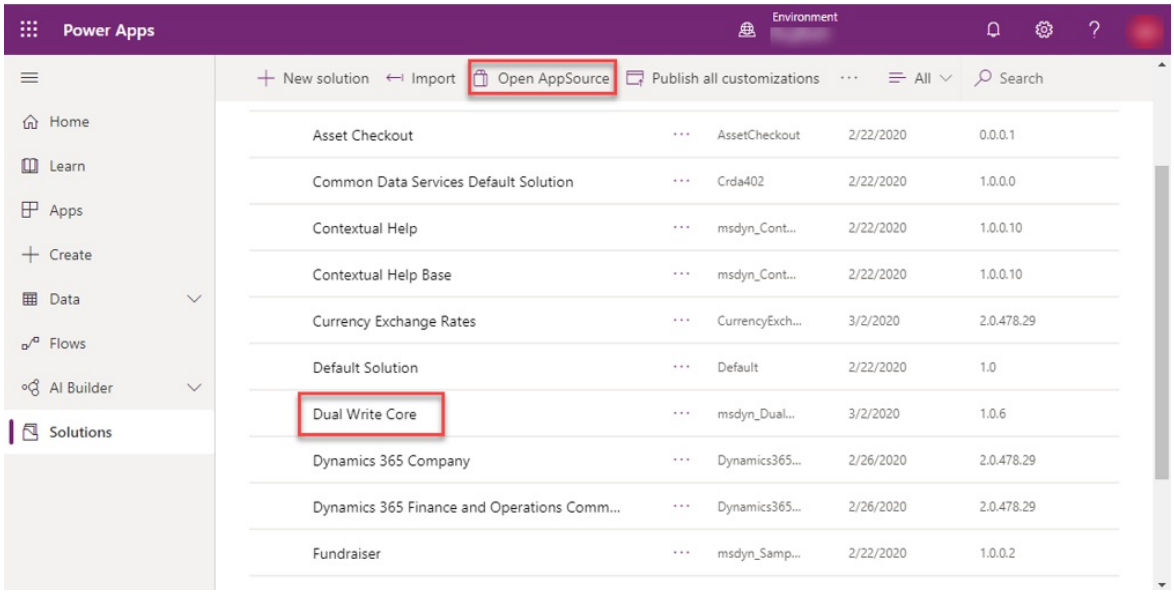
   App version is up to date

   Dual Write is supported on Finance and Operations app environments with Platform Update PU 33 (App version 10.0.9) or above

2. Install the dual-write core solution.

   The dual-write core solution contains metadata for your table maps and must be installed in your environments.

   a. In Power Apps, in the left pane, select **Solutions**.
   b. Select **Open AppSource**.
   c. Select the **Dual Write Core** solution.
   d. Follow the prompts to import the solution.



Related health check result:

The dual-write core solution was found

The dual-write core solution contains metadata for your table maps and must be installed in the environment

3. Grant Dataverse access so that it can connect to a Finance and Operations app.

   a. Open your instance of the Finance and Operations app, search and navigate to Azure Active Directory applications.

   b. Select **New** to add a new client ID row: **6f7d0213-62b1-43a8-b7f4-ff2bb8b7b452**. This row is the application ID for an app that will be used to connect from Dataverse to the Finance and Operations app.

   c. Repeat the previous two steps to add another client ID row: **2e49aa60-1bd3-43b6-8ab6-03ada3d9f08b**.

   When you've finished, follow these steps to refresh the list of tables:

   a. Go to **Workspaces** > **Data management**, select the **Data tables** tile, and make sure that the table list is filled in.

   b. Go to **Workspaces** > **Data management**, and select the **Framework parameters** tile. Then, on the **Entities** tab (

   ```
   https://<BaseFinanceandOperationsappsURL>/?
   cmp=USMF&mi=DM_DataManagementWorkspaceMenuItem&TableName=DMFDefinitionGroupEntity
   ```

   ), select **Refresh tables list**.

   Related health check result:

   The Dataverse can connect to the Finance and Operations app

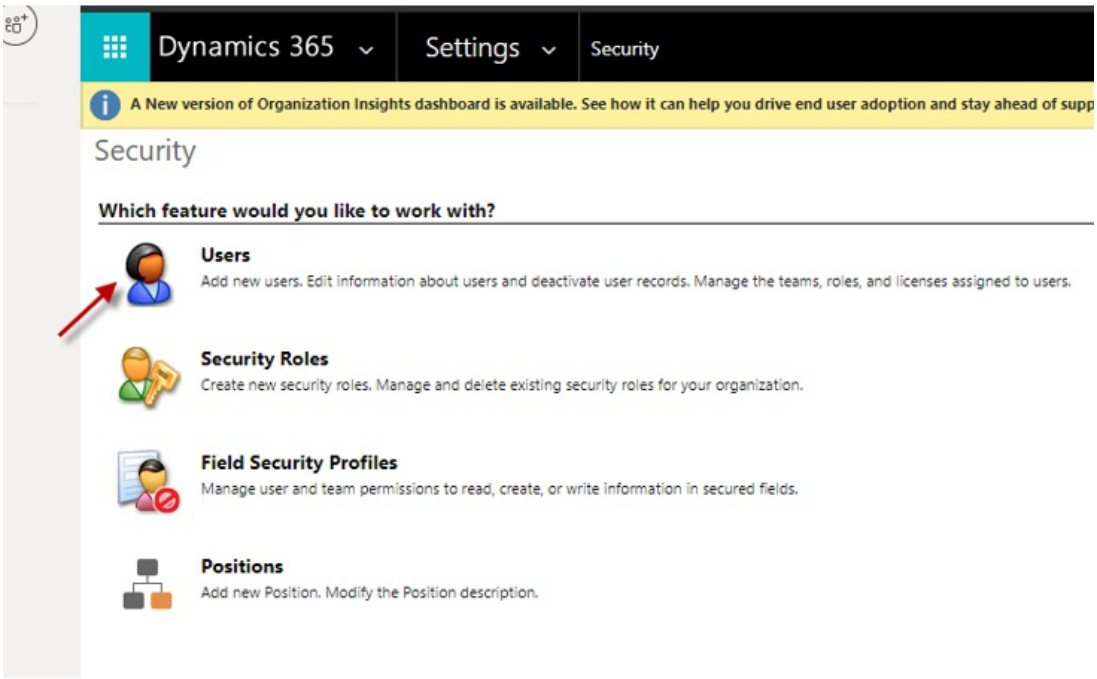
   Before you can enable dual-write, you must grant access to the apps to connect to each other

   App user with id 6f7d0213-62b1-43a8-b7f4-ff2bb8b7b452 exists

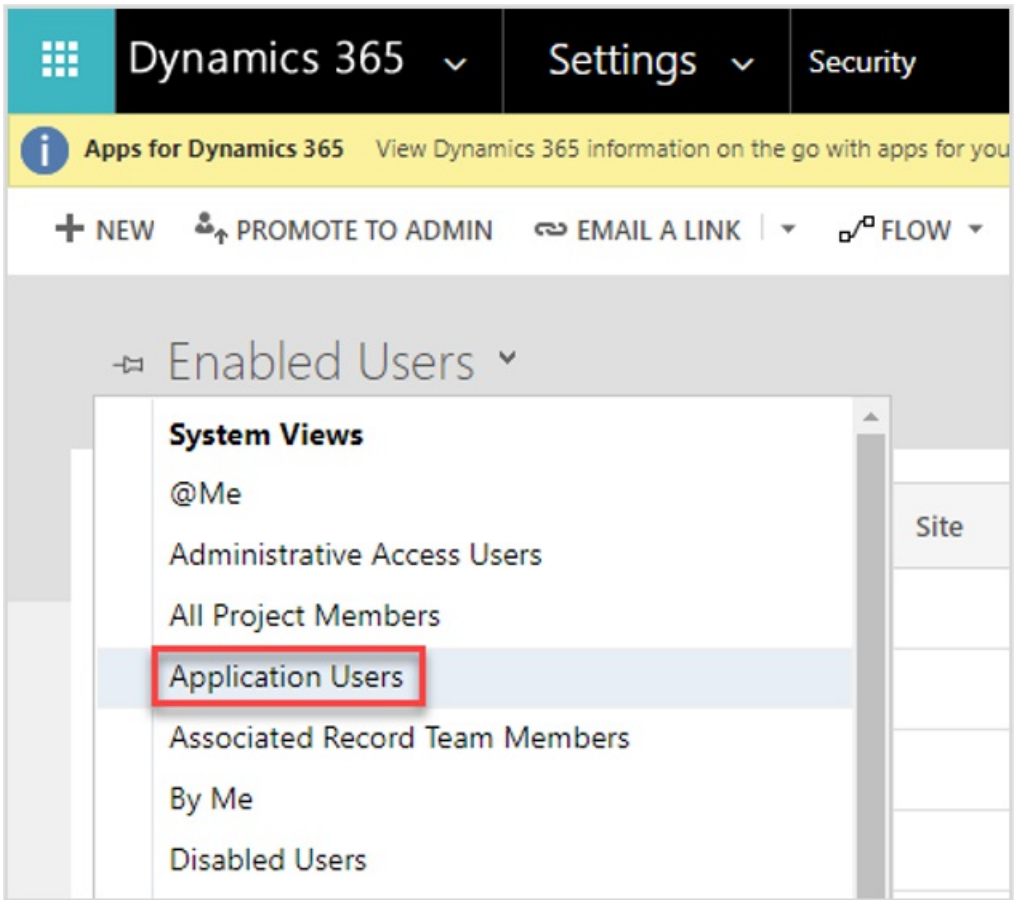
   App user with id 2e49aa60-1bd3-43b6-8ab6-03ada3d9f08b exists

4. Grant a Finance and Operations app access so that it can connect to Dataverse.

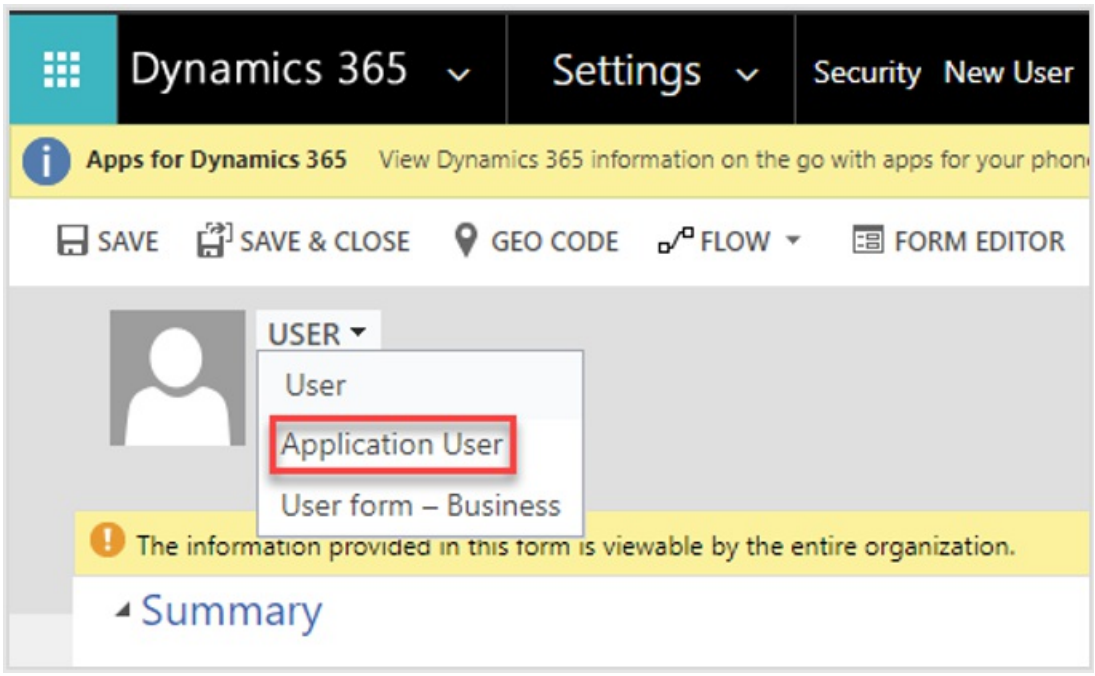
   a. In Power Apps, select the **Settings** button (gear symbol) in the upper-right corner, go to **Advanced settings** > **Security**, and then select **Users**.



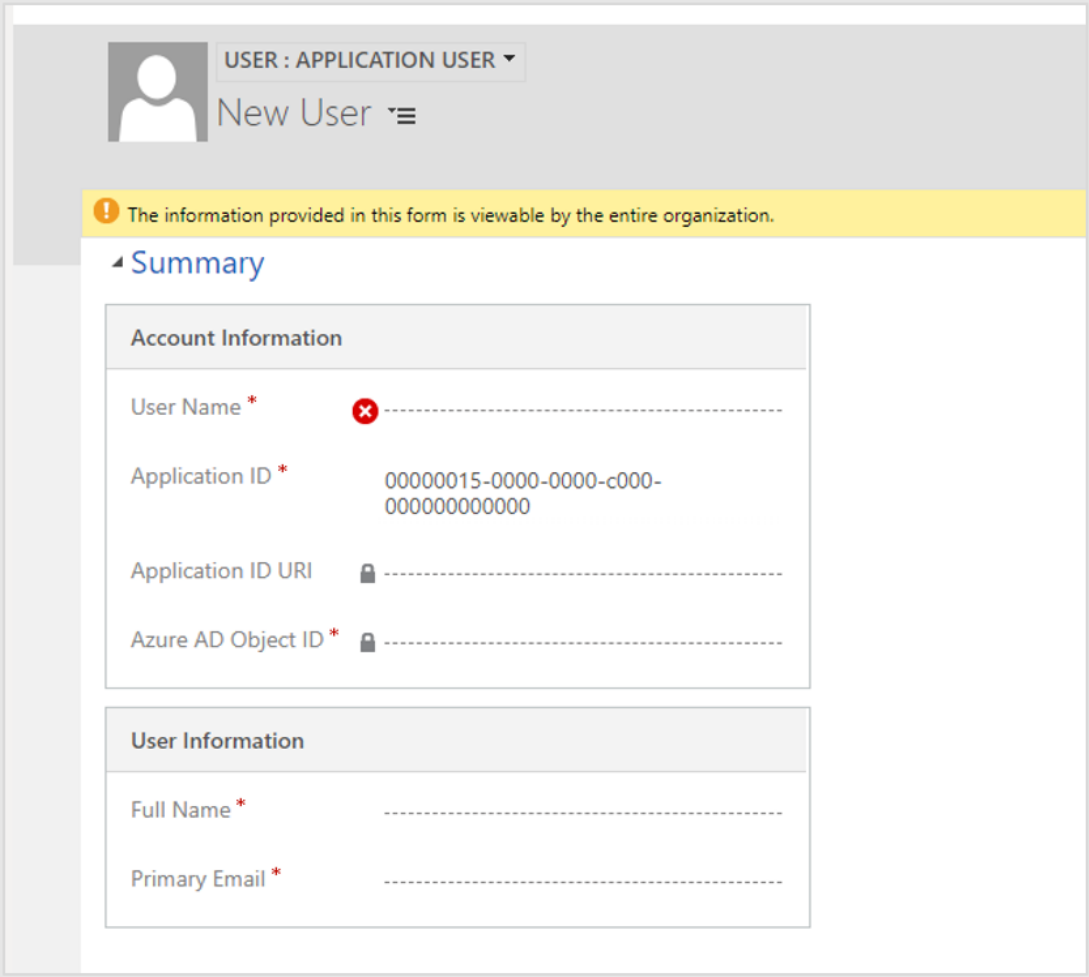
b. Use the drop-down menu to change the view from **Enabled Users** to **Application Users**.



c. Create a new user, and then, on the **User** menu, select **Application User**.

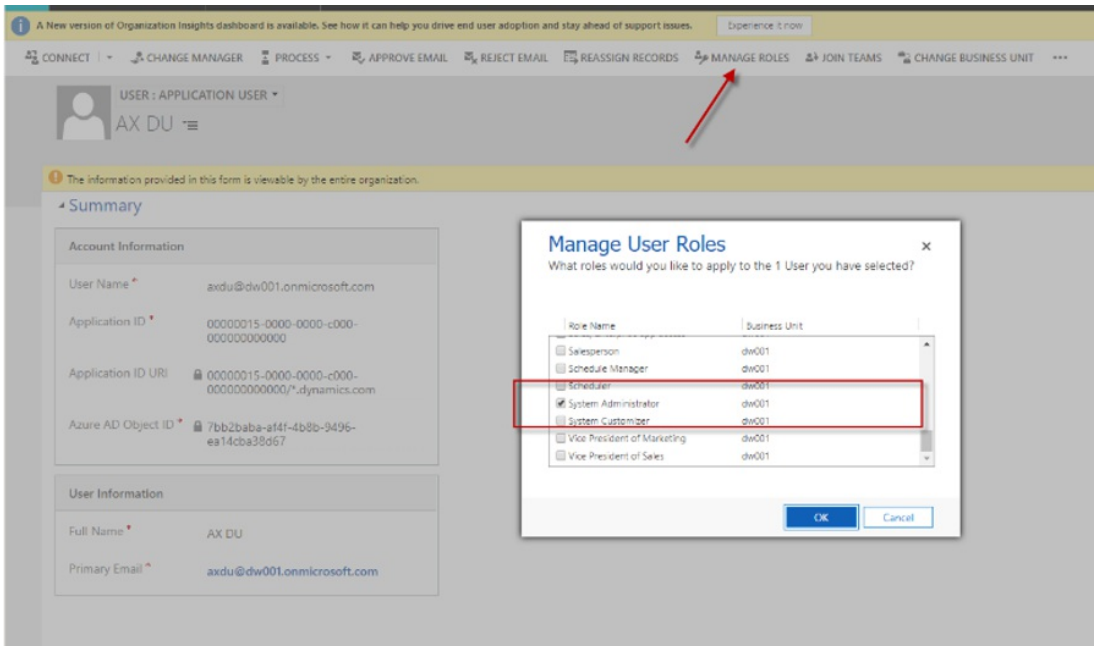


d. In the **Application ID** column, enter **00000015-0000-0000-c000-000000000000**. This application ID is for a Finance and Operations app and will enable the app to connect to Dataverse. When you've finished, follow the prompts to fill in the other columns, and then save the user account.



e. Provide a primary email address.

f. Select **Manage Roles**, and then, in the **Manage User Roles** dialog box, select the **System Administrator** check box to provide system admin rights to the selected application user.



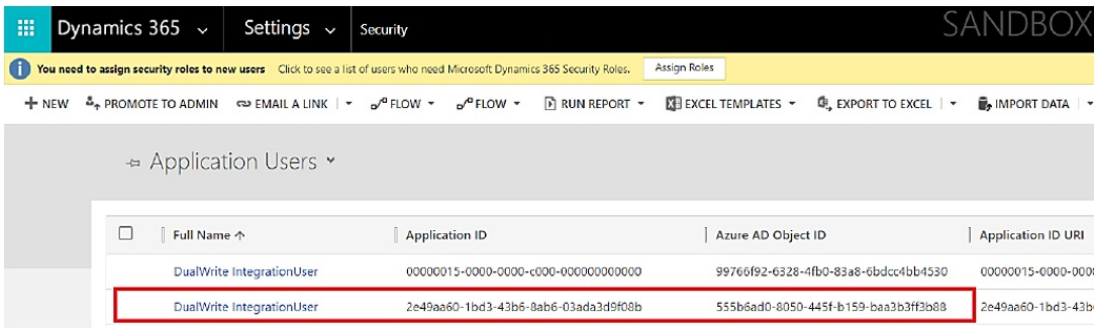
g. Go to **Dynamics 365 > Settings > Security**, select **Teams**, and then change the view to **All Owner Teams**.

h. Select **default team for the root Business Unit**, select **Manage Roles**, and then, in the **Manage Team Roles** dialog box, select a preconfigured **Security Role** to grant a **Read** privilege with a **User** scope for each table integrated through dual-write.

For instructions on how to create a Security Role, see Create or configure a custom security role.

> **NOTE**
>
> The root business unit's default team will become the default owner for all rows integrated through dual-write. Because that team must be assigned a security role, this means that all users in the root business unit will inherit the security role. This means that at the very least, **users from that business unit will have read access to all the rows that are owned by that team**. If this isn't the desired behavior, make sure that users are not a member of the root business unit.

i. Repeat the previous five steps for application ID **2e49aa60-1bd3-43b6-8ab6-03ada3d9f08b**.



Related health check result:
The Finance and Operations app can connect to the Dataverse
Before you can enable dual-write, you must grant access to the apps to connect to each other
App user with id 00000015-0000-0000-c000-000000000000 exists
App user with id 2e49aa60-1bd3-43b6-8ab6-03ada3d9f08b exists

5. Provide app consent in the tenant. For dual-write core solution version 1.0.16.0 or above, this step is no longer needed.

   Related health check result:
   Apps in tenant
   The required dual-write applications need to be installed in the tenant.
   App ID: 6f7d0213-62b1-43a8-b7f4-ff2bb8b7b452
   App ID: 2e49aa60-1bd3-43b6-8ab6-03ada3d9f08b

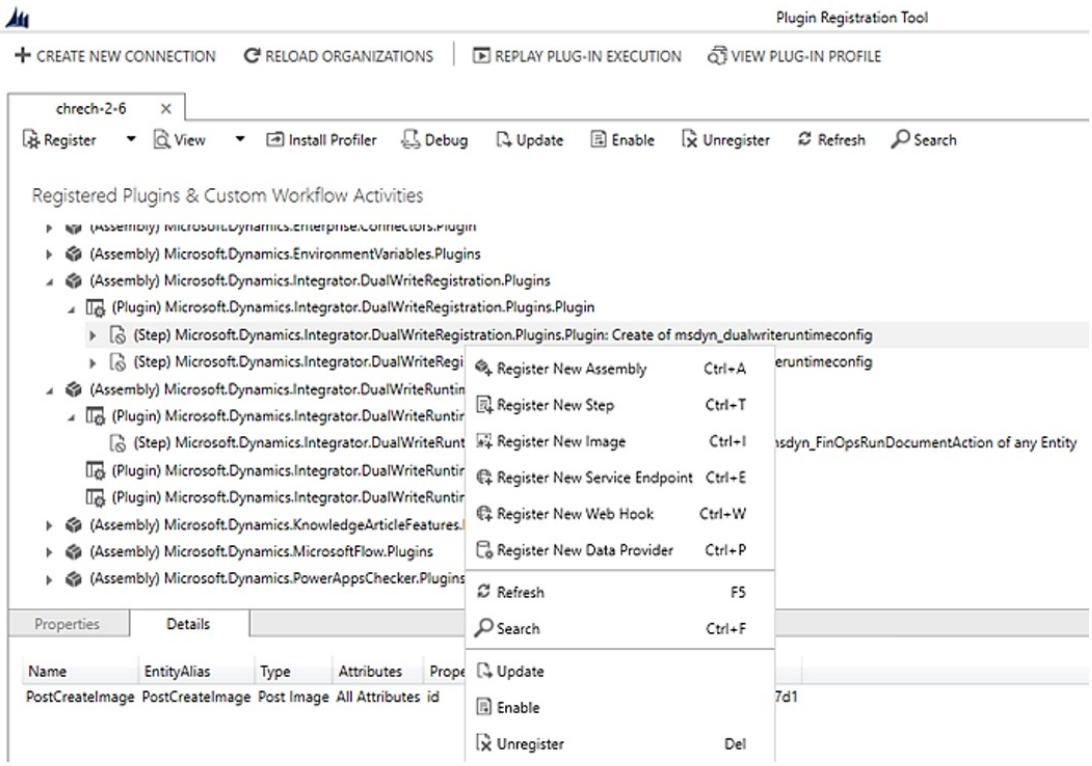
6. Make sure that the dual-write plug-ins are enabled.

   This step isn't usually required, because the plug-ins should be enabled as part of the process of installing the dual-write core solution. However, if the health check fails, follow these steps to manually enable the dual-write plug-ins:

   a. Download the Plug-in Registration Tool.

      In the Plugin Registration Tool, there should be two plug-in assemblies that are associated with dual-write: **DualWriteRegistration.Plugins** and **DualWriteRuntime.Plugins**. These assemblies have plug-in steps that must be enabled, in order, before dual-write can be used. To view the plug-in steps, expand a plug-in assembly and its plug-in types. All the steps that belong to the dual-write plug-in assemblies should be enabled.

   b. To enable a step, select and hold the step (or right-click it), and then select **Enable**. If no **Enable** option is available, only a **Disable** option, the step has already been enabled and doesn't have to be changed.



> **NOTE**
>
> If the dual-write plug-in assemblies can't be found, import the latest version of the dual-write core solution.

Related health check result:
The dual-write registration and runtime plugins are enabled
To ensure listening into CRUD operations on the Dataverse, the dual-write plugins need to be enabled

7. Install the **Dual-write application orchestration solution** maps solution.

   In Power Apps, in the left pane, select **Solutions**. Select **Open AppSource**, and search for the solution that is named **Dual-write application orchestration solution**. Select the solution, and follow the prompts to import it. After installation, you'll find several new solutions listed under **Solutions**. For more information, see Solutions overview.

   While the dual-write core solution contains metadata for your table maps, the dual-write application orchestration solution covers these additional master data scenarios:

   - Customers, products, and vendors.
   - End-to-end process flows like prospect to cash.
   - On-demand functions like pricing.
   - Reference data for ledger, tax, payment terms, and schedules.

   Dual-write will continue to expand in the future to support more scenarios including party, project, and hands-on inventory. The framework is extensible and accommodates customer-centric business data exchange through a few additional clicks.

   > **NOTE**
   >
   > You must select **Apply Solution** as part of the next steps, when you use the dual-write wizard to link your environments. It may take few minutes for the solution packages to be created in Power Apps solutions section. Wait for it to appear before moving to the next step.

8. Uninstall the Prospect to Cash (P2C) solution.

The P2C solution doesn't work concurrently with dual-write. Therefore, don't install the P2C solution. If it's already installed, you must uninstall it before you enable dual-write.

9. Provide the supported tenant configuration.

   Make sure that the Finance and Operations app and Dataverse are installed under the same tenant. Cross-tenant scenarios aren't currently supported.

   > **NOTE**
   > For dual-write core solution versions lower than 1.0.16.0, see the following section for modifications and additional steps.

For dual-write core solution lower than version 1.0.16.0 only

1. In step Step 3b above, create a new client ID row: **33976c19-1db5-4c02-810e-c243db79efde** (versus 6f7d0213-62b1-43a8-b7f4-ff2bb8b7b452).
2. Add app consent in the tenant:

   a. Open the following URL, and sign in by using your admin credentials. You should be prompted for consent.

      https://login.microsoftonline.com/common/oauth2/authorize?client_id=33976c19-1db5-4c02-810e-c243db79efde&response_type=code&prompt=admin_consent

   b. Select **Accept**.

      By selecting **Accept**, you indicate that you're providing consent to install the app that has application ID **33976c19-1db5-4c02-810e-c243db79efde** in your tenant. Dataverse requires this app to communicate with the Finance and Operations app.

Related health check result:
Apps in tenant
The required dual-write applications need to be installed in the tenant.
App ID: 33976c19-1db5-4c02-810e-c243db79efde
App ID: 2e49aa60-1bd3-43b6-8ab6-03ada3d9f08b

Next steps

Use the dual-write wizard to link your environments

> **NOTE**
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Use the dual-write wizard to link your environments

2/18/2021 • 2 minutes to read • Edit Online

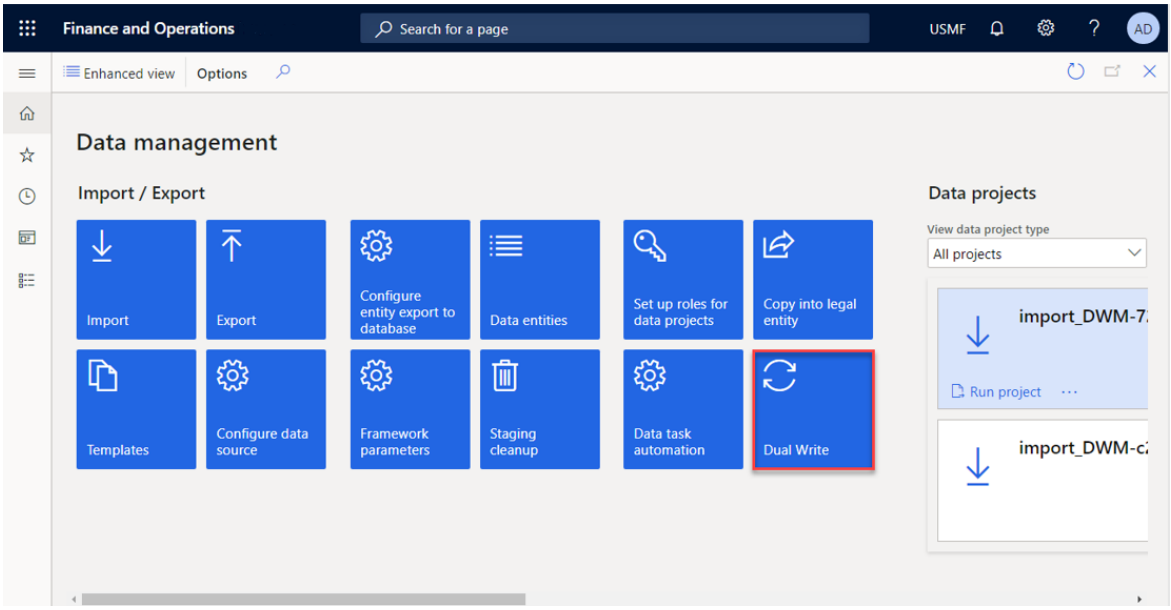
NOTE

Effective November 2020:

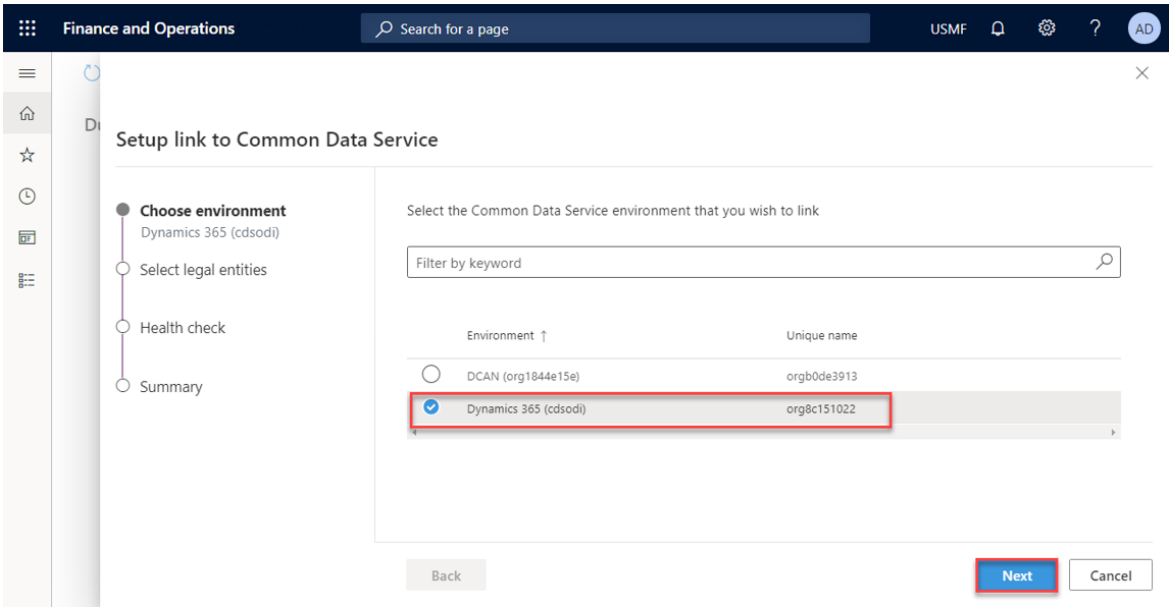
- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

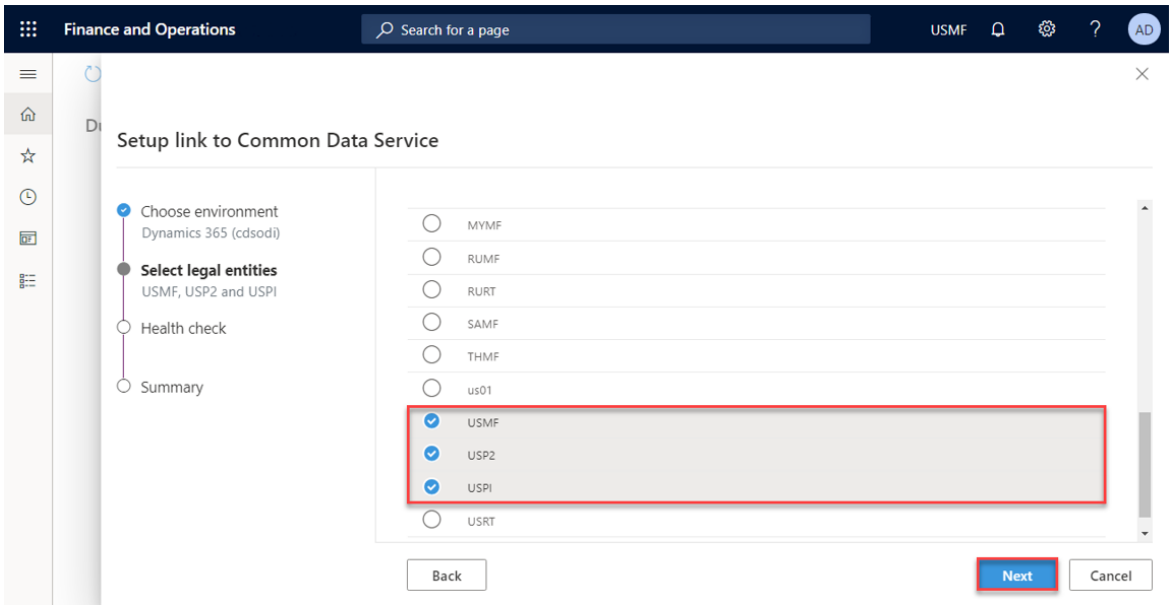
1. Sign in to the Finance and Operations app environment that you want to link to your Dataverse environment.
2. Go to **Workspaces** > **Data management**, and select the **Dual Write** tile.



3. Select **New link to environment** to open the **Setup link to Dataverse** wizard.
4. The **Choose environment** page lists all the Dataverse environments where the signed-in user is an environment admin. Select the Dataverse environment to link to, and then select **Next**.

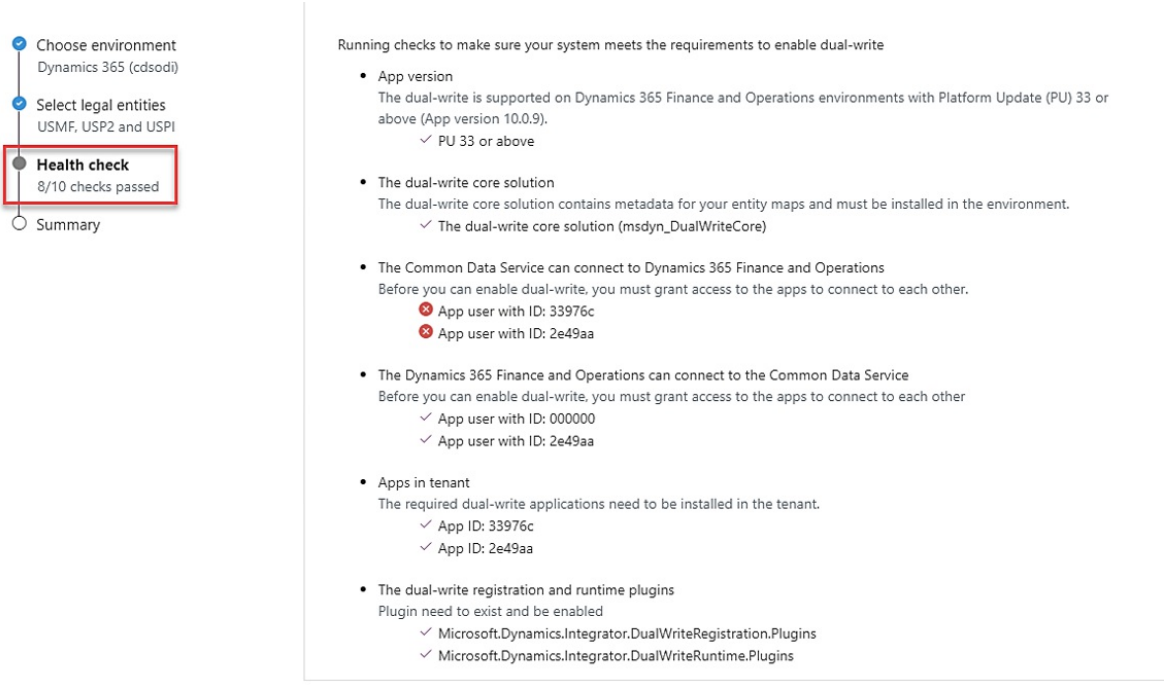


5. Select your legal entities, and then select **Next**.



A health check is run to verify that your system meets the requirements for enabling dual-write. The health check also verifies that all the prerequisites have been completed. If any health check test fails, make that you've completed all the prerequisites before you move on to the next step.

In the following example, the test about whether access was granted to connect the apps failed. In this case, you must first grant access to connect the apps by the creating the appropriate application IDs. You must then rerun the wizard.



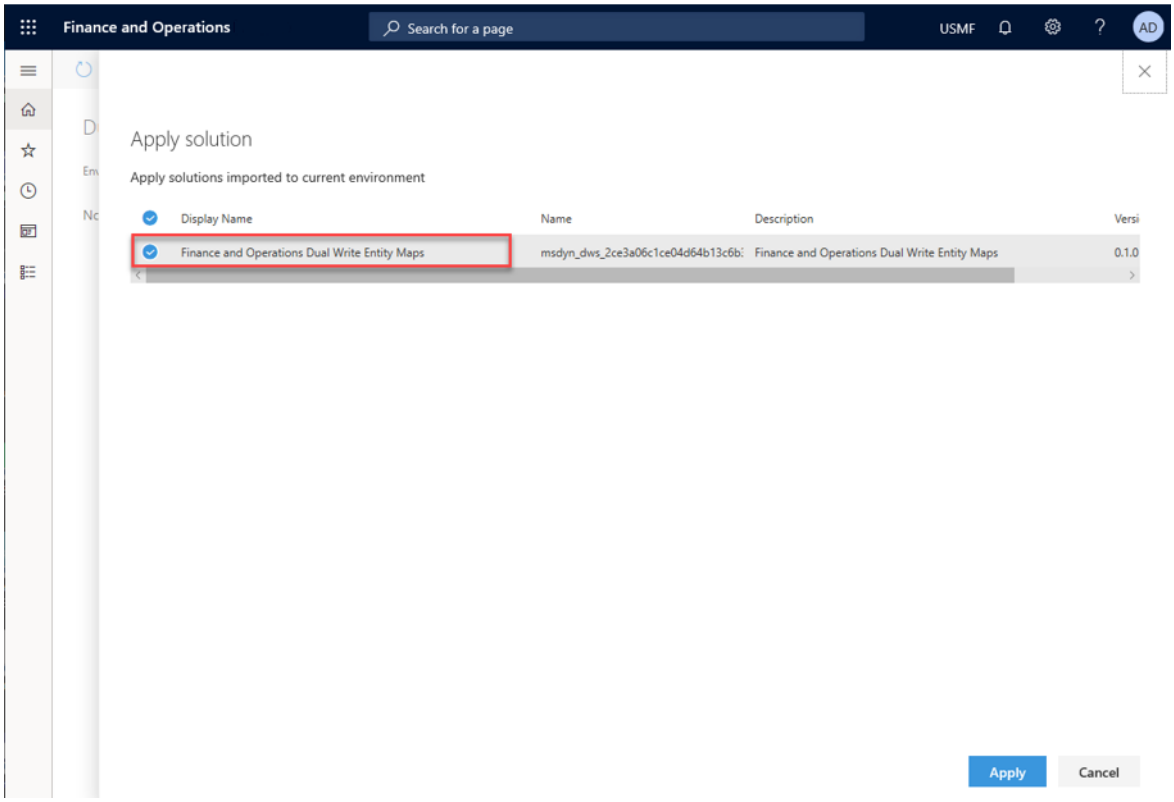
6. Review the summary, privacy notice, and consent, and then select **Create**.

You've now linked your Finance and Operations app to the Dataverse environment.

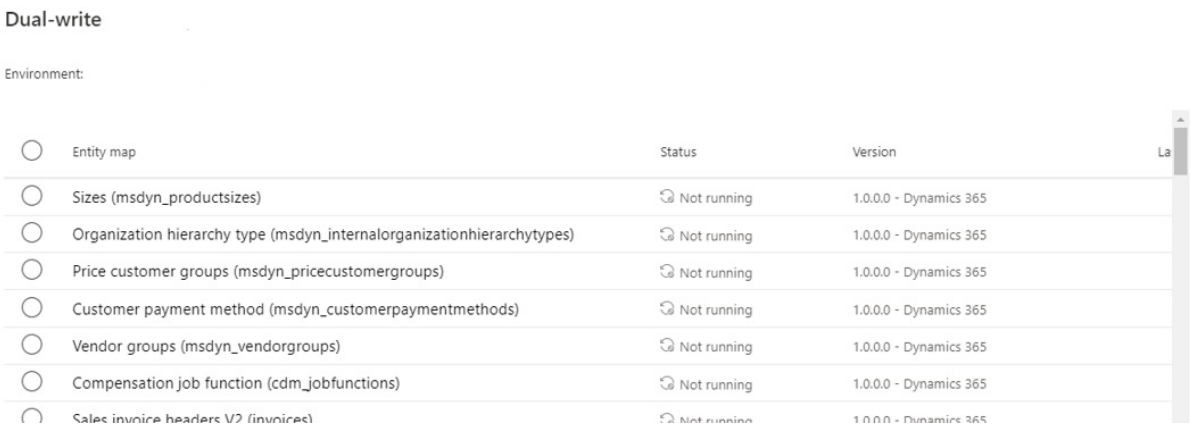
> **NOTE**
>
> If you don't see your table maps, or if you see a blank page, be sure to **Apply** the Dual-write application orchestration solution that you installed as part of the system requirements and prerequisities.

7. Apply the dual-write application orchestration solution.

   In the Finance and Operations app, on the **Dual-write** page, select **Apply Solution** to apply the table maps that you just downloaded and installed. After you apply the solution, you should see that the default table maps are published.



You've now successfully imported and applied a Microsoft-published dual-write table map solution to your environment.



Next steps

Enable table maps for dual-write

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Enable table maps for dual-write

2/18/2021 • 3 minutes to read • Edit Online

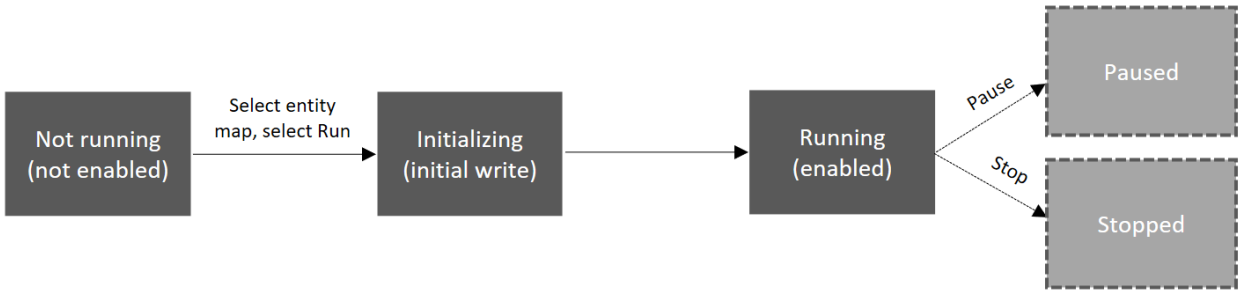
NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

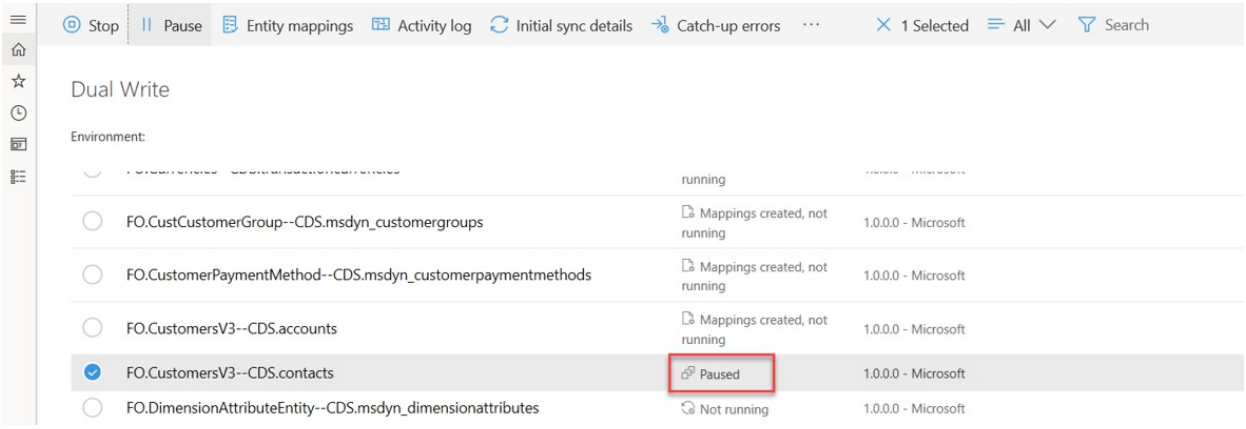
When you enable a table map for dual-write, it begins at the **Not running** status. The table map then goes through an initialization phase, where it does an initial write by copying pre-existing data on tables on both sides. Finally, when the table is completely enabled, the table map sets the status to **Running**.



Enabling entity map

While the status is **Running**, you can pause a table. All changes are then queued until you resume. When you resume, the table goes into "catch-up mode," where all the queued changes are played back.

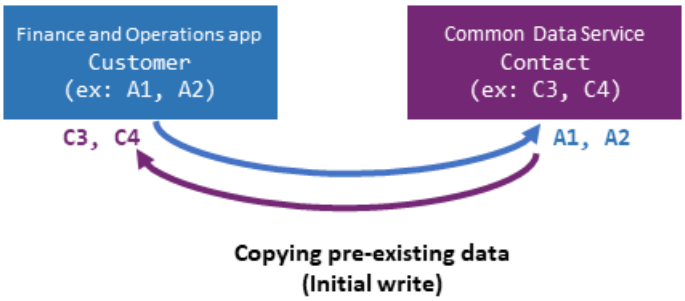
The following illustration show an example of a table that is paused.



STATUS	DESCRIPTION	AVAILABLE ACTIONS
Not running	The table has not yet been enabled for dual-write. Every table begins at the **Not running** status.	Run
Initializing	The initial write is occurring.	None

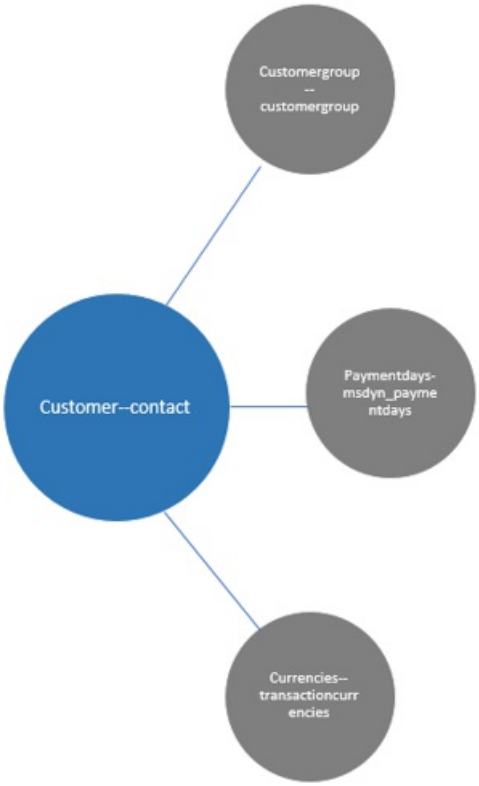
STATUS	DESCRIPTION	AVAILABLE ACTIONS
Running	The table has been enabled for dual-write.	Stop, Pause
Paused	The table is in a paused state, and all new requests are queued.	Run
Resuming	The table is catching up on rows that were queued while the table was paused.	None

During the initialization phase, any pre-existing data that you have is copied as part of the initial write phase.



Copying pre-existing data (Initial write)

Entities have several dependent tables. For example, Customer-Contact tables have customer groups and currencies as dependent tables.



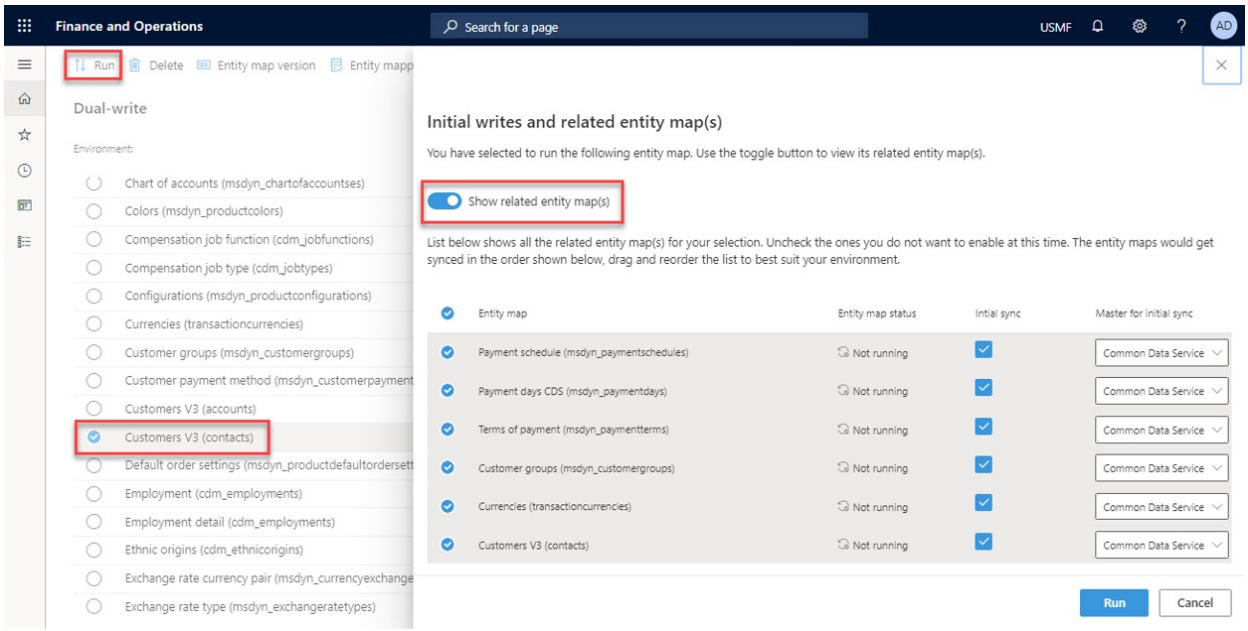
Dependent or Related Entities

Because these are relational apps that have relational data, if you don't enable the dependent tables, you might encounter errors later. To help prevent these errors, before you enable a table map, you're provided with a list of the related tables that we recommend that you enable.

Example: Enabling the Customers V3—Contacts table map

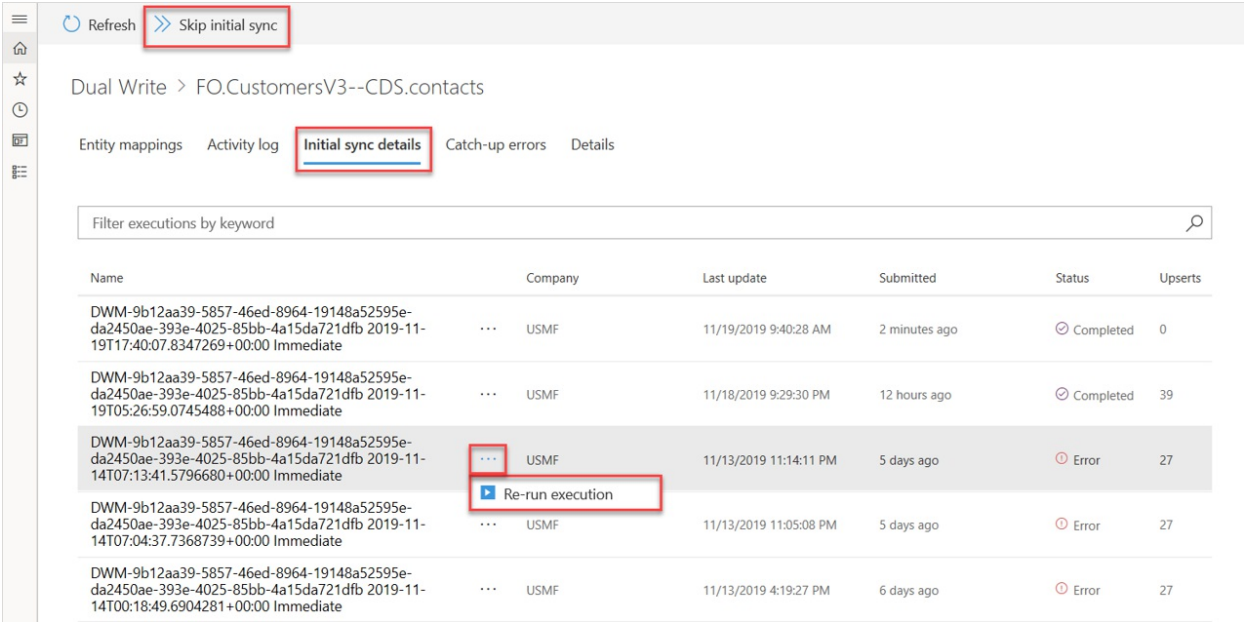
When you select a table map (for example, **Customers V3—Contacts**) and select **Run**, a dialog box appears before the table map is enabled. This dialog box lists all the dependent tables. You can select the **Show related table map(s)** option to show all the related table maps. To enable the selected table map and all its related tables, select **Run** in the dialog box.

> **NOTE**
>
> The behavior is similar when you pause a table. In that case, you have the option to pause all the related tables too.



You can further customize this by specifying a different master that should be used to resolve conflicts. (By default, Dataverse is used.) If you don't want to copy pre-existing data, skip the initial synchronization by clearing the **Initial Sync** check box. Alternatively, remove one or more of the related tables by canceling the selection of them. You can also drag the table maps to change the order that they will be synced in.

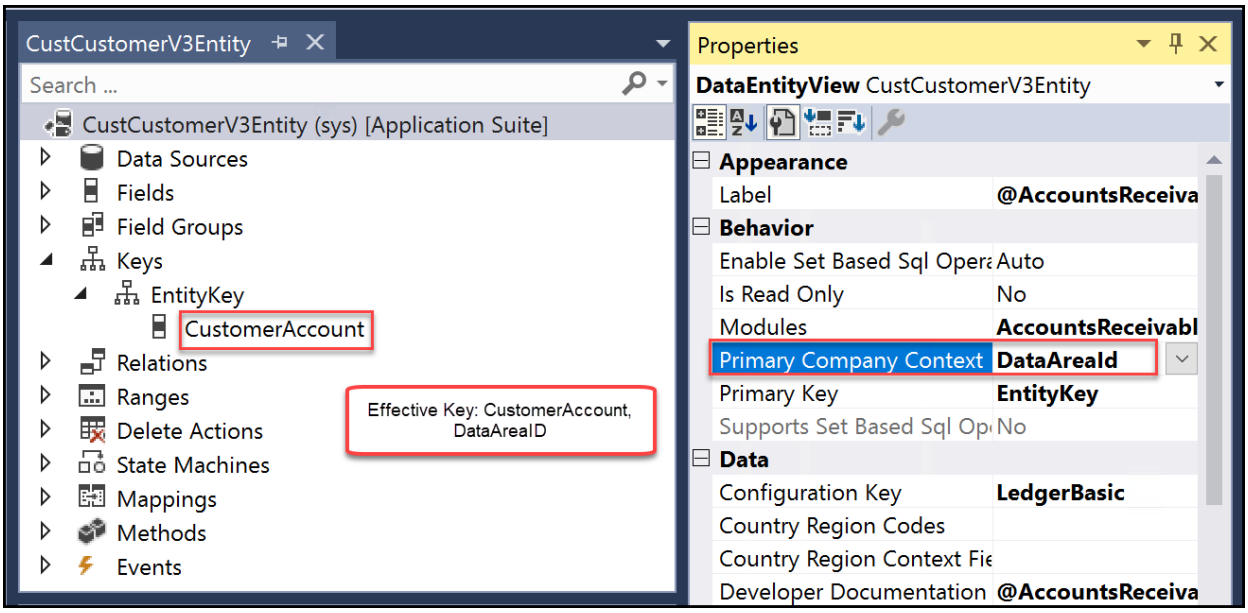
After you've finished making your selections in the dialog box, and you select **Run**, the table map and all its related tables go through the initial write phase. You're redirected to the table map list page. If any errors occur, you can view the details on the **Initial sync details** tab. This tab provides details about all the errors that occur while pre-existing data is being copied. After you fix the underlying errors, you can rerun the execution and monitor the outcome. Alternatively, if you no longer want to sync the pre-existing data, or if you experience recurring issues because of underlying data, you can skip the initial write phase. Instead, you can turn on live writes by selecting **Skip initial sync**.



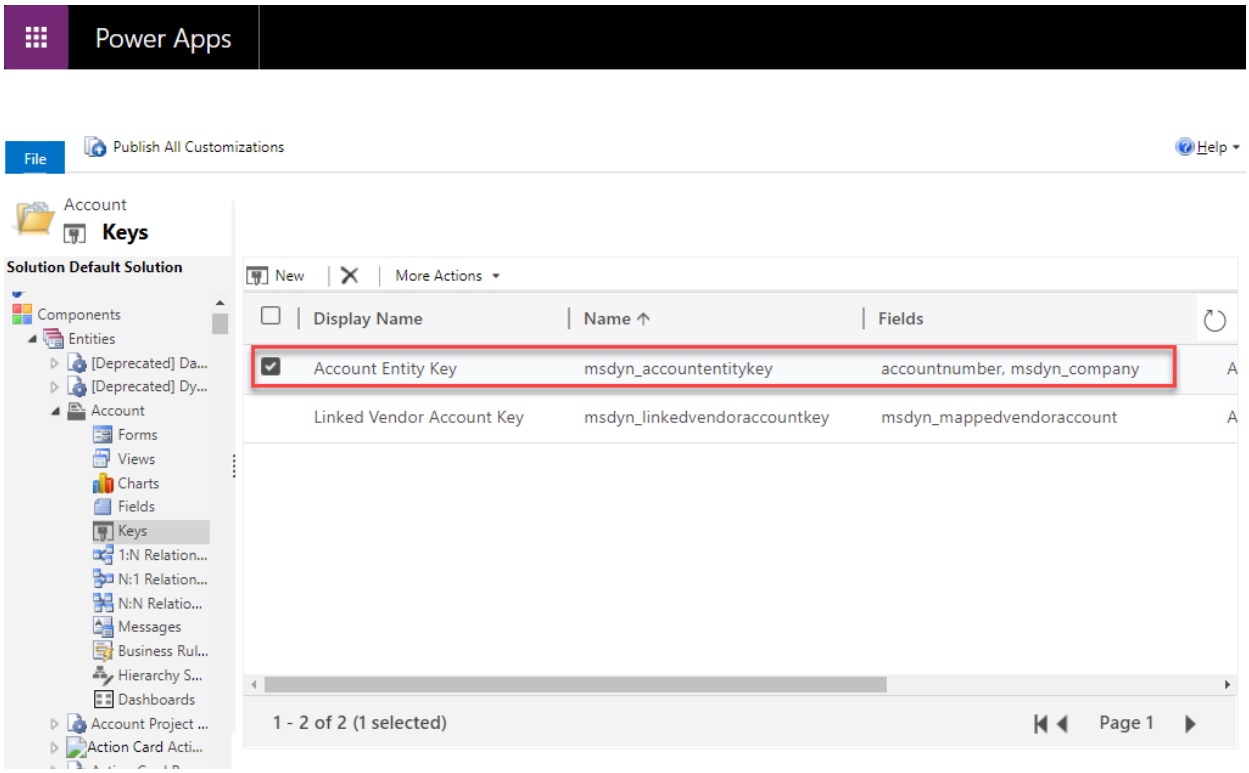
Criteria for linking tables

To enable table maps for dual-write, you must define an alternative key in Dataverse. The value of the alternative key in Dataverse must match the key that is defined in the Finance and Operations app.

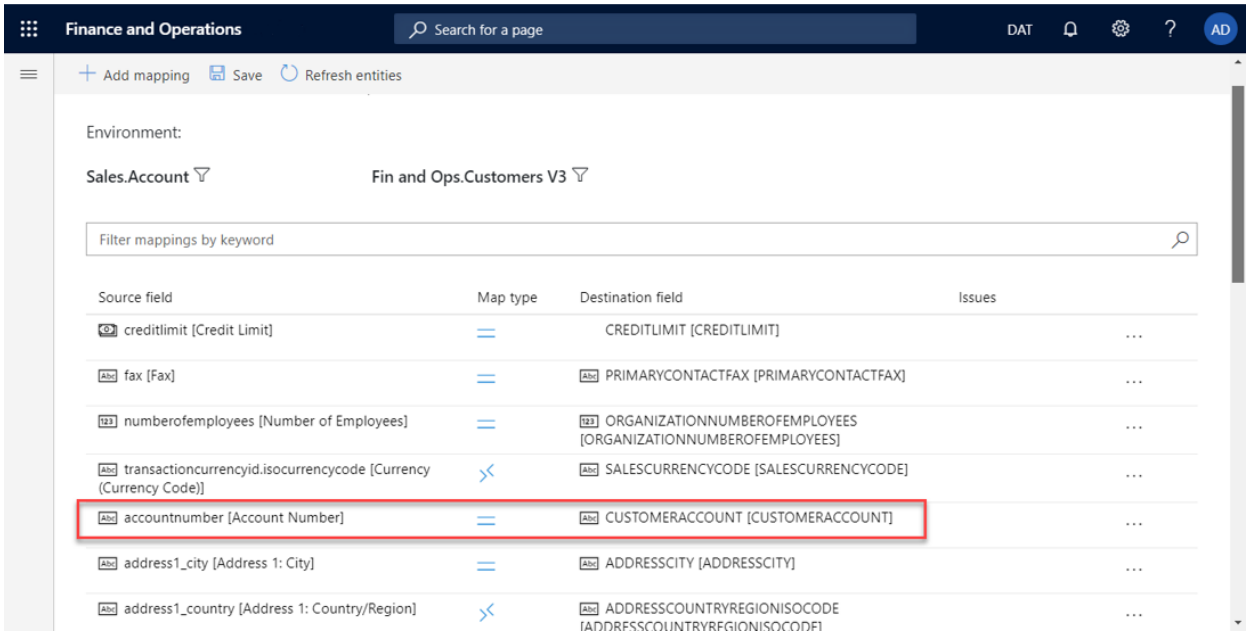
For example, in a Finance and Operations app, **CustomerAccount** is the key for the Account table.



In Dataverse, **accountnumber** is defined as the key for the Account table.



In the Customers V3 table map, you can see that **accountnumber** is mapped to **CustomerAccount**.



Next steps

Customize table and column mappings

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Currency data-type migration for dual-write

2/18/2021 • 4 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

You can increase the number of decimal places that are supported for currency values to a maximum of 10. The default limit is four decimal places. By increasing the number of decimal places, you help prevent data loss when you use dual-write to sync data. The increase in the number of decimal places is an opt-in change. To implement it, you must request assistance from Microsoft.

The process of changing the number of decimal places has two steps:

1. Request migration from Microsoft.
2. Change the number of decimal places in Dataverse.

The Finance and Operations app and Dataverse must support the same number of decimal places in currency values. Otherwise, data loss can occur when this information is synced between apps. The migration process reconfigures the way that currency and exchange rate values are stored, but it doesn't change any data. After the migration is completed, the number of decimal places for currency codes and pricing can be increased, and the data that users enter and view can have more decimal precision.

Migration is optional. If you might benefit from support for more decimal places, we recommend that you consider migration. Organizations that don't require values that have more than four decimal places don't have to migrate.

Requesting migration from Microsoft

Storage for existing currency columns in Dataverse can't support more than four decimal places. Therefore, during the migration process, currency values are copied to new internal columns in the database. This process occurs continuously until all data has been migrated. Internally, at the end of migration, the new storage types replace the old storage types, but the data values are unchanged. The currency columns can then support up to 10 decimal places. During the migration process, Dataverse can continue to be used without interruption.

At the same time, exchange rates are modified so that they support up to 12 decimal places instead of the current limit of 10. This change is required so that the number of decimal places is the same in both the Finance and Operations app and Dataverse.

Migration doesn't change any data. After the currency and exchange rate columns are converted, admins can configure the system to use up to 10 decimal places for currency columns by specifying the number of decimal places for each transaction currency and for pricing.

Request a migration

To make this feature available, email **CDSExpandDecimal@microsoft.com**, and include the following information:

- **Subject:** Request to enable expanded decimal support for <organizationID>
- **Body:** I would like to enable expanded decimal support for my org <organizationID>.

A Microsoft representative will contact you within two to three business days for the next steps.

When you request a migration, you should be aware of the following details and plan for them accordingly:

- The time that is required to migrate the data depends the amount of data in the system. Migration of large databases can take several days.
- The size of the database temporarily increases while the migration is running, because additional space is needed for indexes. Most of the additional space is freed when the migration is completed.
- During the migration process, if errors occur that prevent the migration from being completed, the system raise alerts to Microsoft Support, so that Support staff can intervene. However, even if errors occur during the migration, Dataverse remains fully available for regular use.
- The migration process isn't reversible.

Changing the number of decimal places

After the migration is completed, Dataverse can store numbers that have more decimal places. Admins can choose how many decimal places are used for specific currency codes and for pricing. Users of Microsoft Power Apps, Power BI, and Power Automate can then view and use numbers that have more decimal places.

To make this change, you must update the following settings in Power Apps:

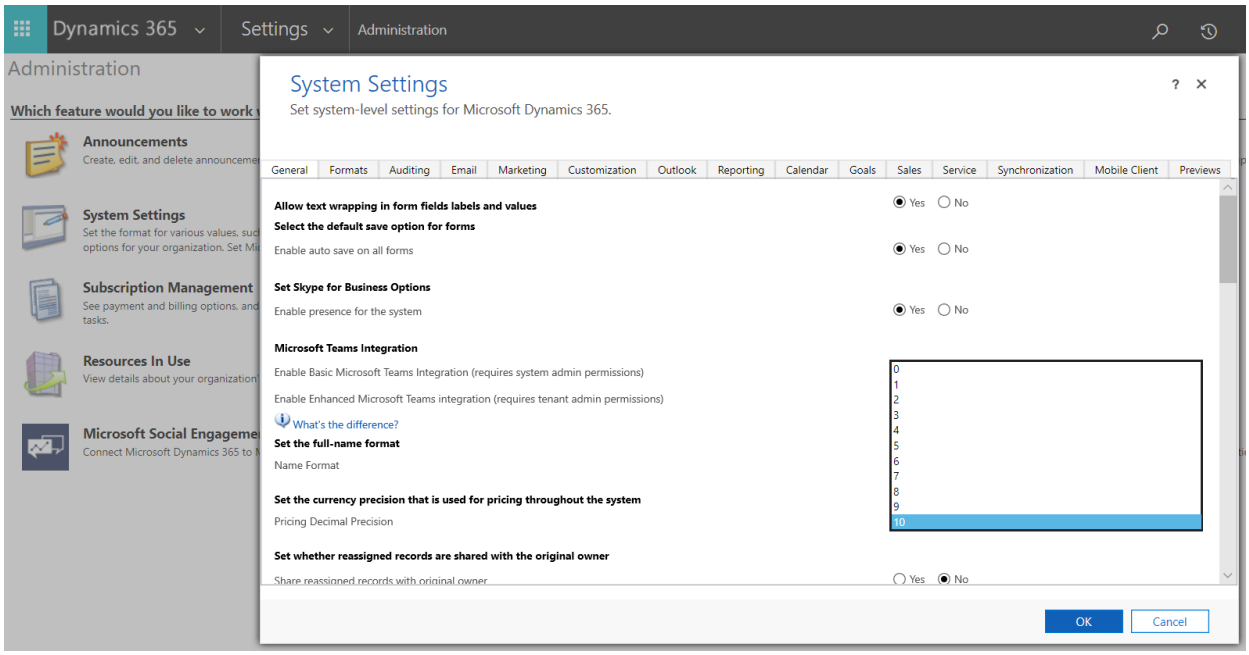
- **System Settings: Currency precision for pricing** – The **Set the currency precision that is used for pricing throughout the system** column defines how the currency will behave for the organization when **Pricing Precision** is selected.
- **Business Management: Currencies** – The **Currency Precision** column lets you specify a custom number of decimal places for a specific currency. There is a fallback to the organization-wide setting.

There are some limitations:

- You can't configure the currency column on a table.
- You can specify more than four decimal places only at the **Pricing** and **Transaction Currency** levels.

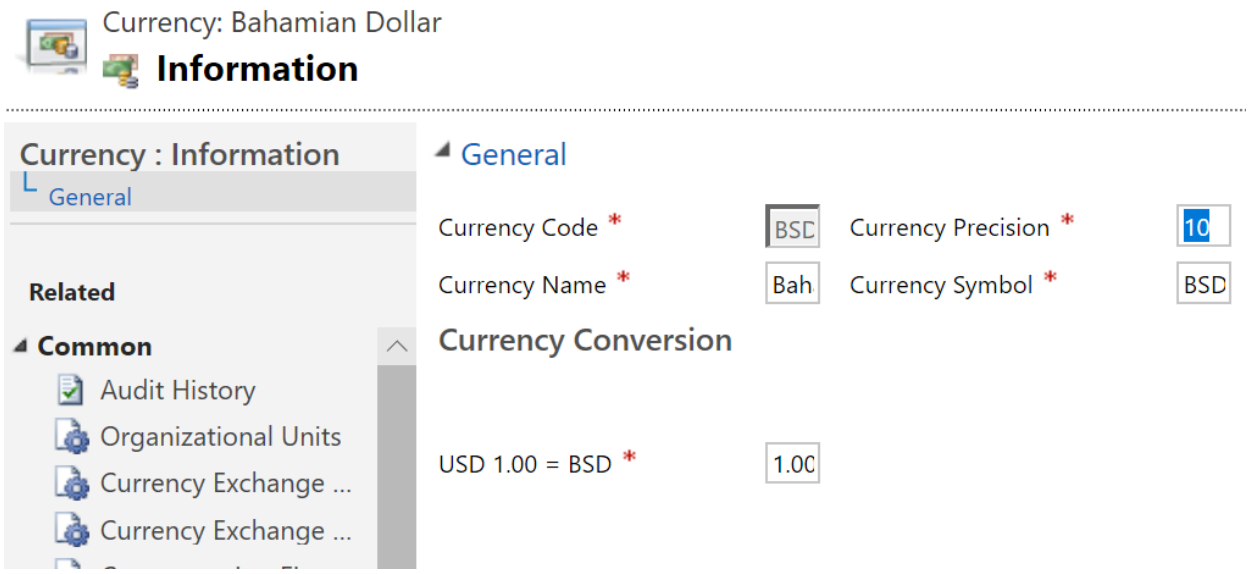
System Settings: Currency precision for pricing

After migration is completed, admins can set the currency precision. Go to **Settings > Administration**, and select **System Settings**. Then, on the **General** tab, change the value of the **Set the currency precision that is used for pricing throughout the system** column, as shown in the following illustration.



Business Management: Currencies

If you require that the currency precision for a specific currency differ from the currency precision that is used for pricing, you can change it. Go to **Settings > Business Management**, select **Currencies**, and select the currency to change. Then set the **Currency Precision** column to the number of decimal places that you want, as shown in the following illustration.



tables: Currency column

The number of decimal places that can be configured for specific currency columns is limited to four.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Set up the mapping for the sales order status columns

2/18/2021 • 2 minutes to read • Edit Online

The columns that indicate sales order status have different enumeration values in Microsoft Dynamics 365 Supply Chain Management and Dynamics 365 Sales. Additional setup is required to map these columns in dual-write.

columns in Supply Chain Management

In Supply Chain Management, two columns reflect the status of the sales order. The columns that you must map are **Status** and **Document Status**.

The **Status** enumeration specifies the overall status of the order. This status is shown on the order header.

The **Status** enumeration has the following values:

- Open Order
- Delivered
- Invoiced
- Cancelled

The **Document Status** enumeration specifies the most recent document that was generated for the order. For example, if the order is confirmed, this document is a sales order confirmation. If a sales order is partially invoiced, and then the remaining line is confirmed, the document status remains **Invoice**, because the invoice is generated later in the process.

The **Document Status** enumeration has the following values:

- Confirmation
- Picking List
- Packing Slip
- Invoice

columns in Sales

In Sales, two columns indicate the status of the order. The columns that you must map are **Status** and **Processing Status**.

The **Status** enumeration specifies the overall status of the order. It has the following values:

- Active
- Submitted
- Fulfilled
- Invoiced
- Cancelled

The **Processing Status** enumeration was introduced so that the status can be mapped more accurately with Supply Chain Management.

The following table shows the mapping of **Processing Status** in Supply Chain Management.

PROCESSING STATUS	STATUS IN SUPPLY CHAIN MANAGEMENT	DOCUMENT STATUS IN SUPPLY CHAIN MANAGEMENT
Active	Open Order	None
Confirmed	Open Order	Confirmation
Picked	Open Order	Picking List
Partially Delivered	Open Order	Packing Slip
Delivered	Delivered	Packing Slip
Partially Invoiced	Delivered	Invoice
Invoiced	Invoiced	Invoice
Cancelled	Cancelled	Not applicable

The following table shows the mapping of **Processing Status** between Sales and Supply Chain Management.

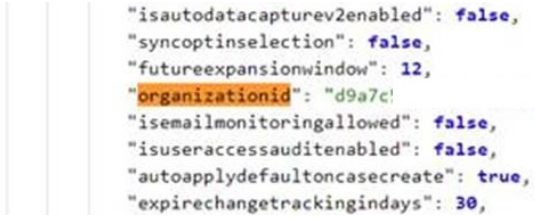
PROCESSING STATUS	STATUS IN SALES	STATUS IN SUPPLY CHAIN MANAGEMENT
Active	Active	Open Order
Confirmed	Submitted	Open Order
Picked	Submitted	Open Order
Partially Delivered	Active	Open Order
Partially Invoiced	Active	Open Order
Partially Invoiced	Fulfilled	Delivered
Invoiced	Invoiced	Invoiced
Cancelled	Cancelled	Cancelled

Setup

To set up the mapping for the sales order status columns, you must enable the **IsSOPIntegrationEnabled** and **isIntegrationUser** attributes.

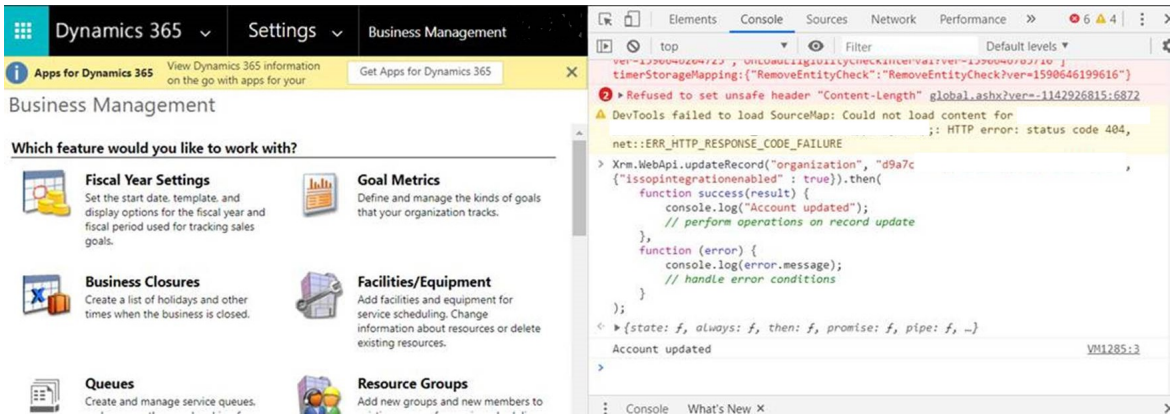
To enable the **IsSOPIntegrationEnabled** attribute, follow these steps.

1. In a browser, go to `https://<test-name>.crm.dynamics.com/api/data/v9.0/organizations`. Replace **<test-name>** with your company's link to Sales.
2. On the page that is opened, find **organizationid**, and make a note of the value.

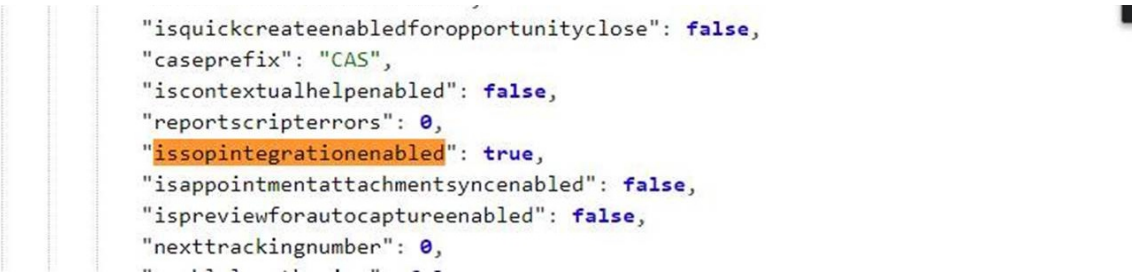


3. In Sales, open the browser console, and run following script. Use the **organizationid** value from step 2.

```
Xrm.WebApi.updateRecord("organization",
"d9a7c5f7-acbf-4aa9-86e8-a891c43f748c", {"issopintegrationenabled" :
true}).then(
    function success(result) {
        console.log("Account updated");
        // perform operations on row update
    },
    function (error) {
        console.log(error.message);
        // handle error conditions
    }
);
```

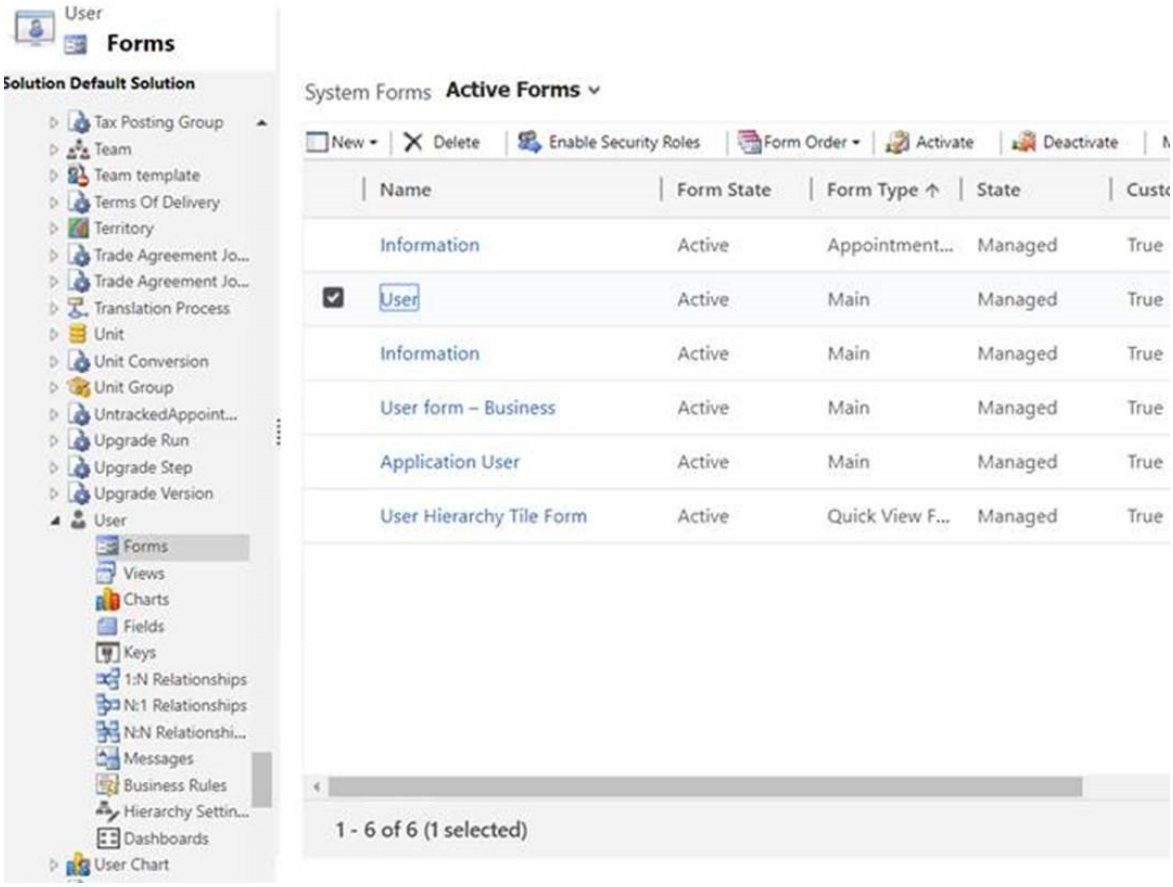


4. Verify that **IsSOPIntegrationEnabled** is set to **true**. Use the URL from step 1 to check the value.

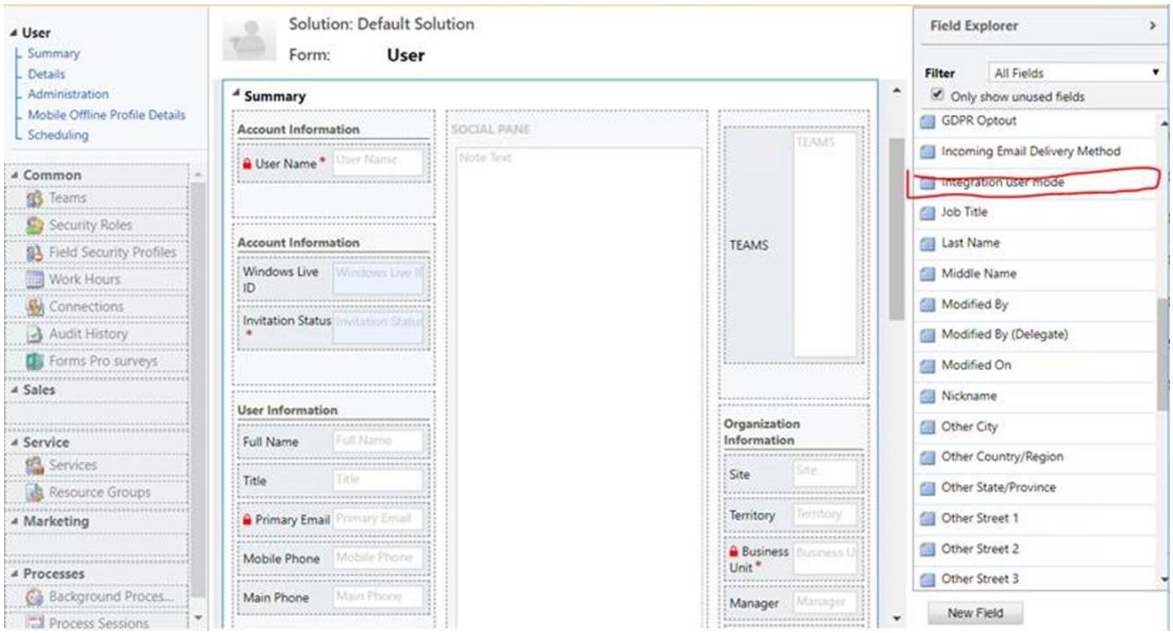


To enable the **isIntegrationUser** attribute, follow these steps.

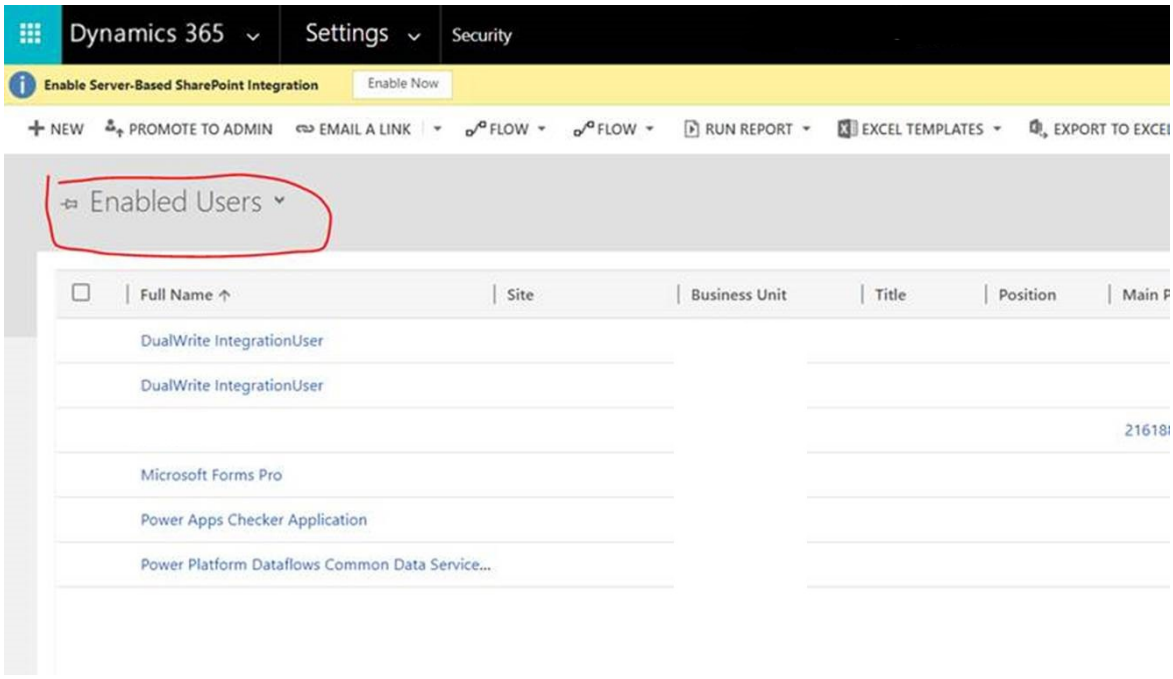
1. In Sales, go to **Setting > Customization > Customize the System**, select **User table**, and then open **Form > User**.



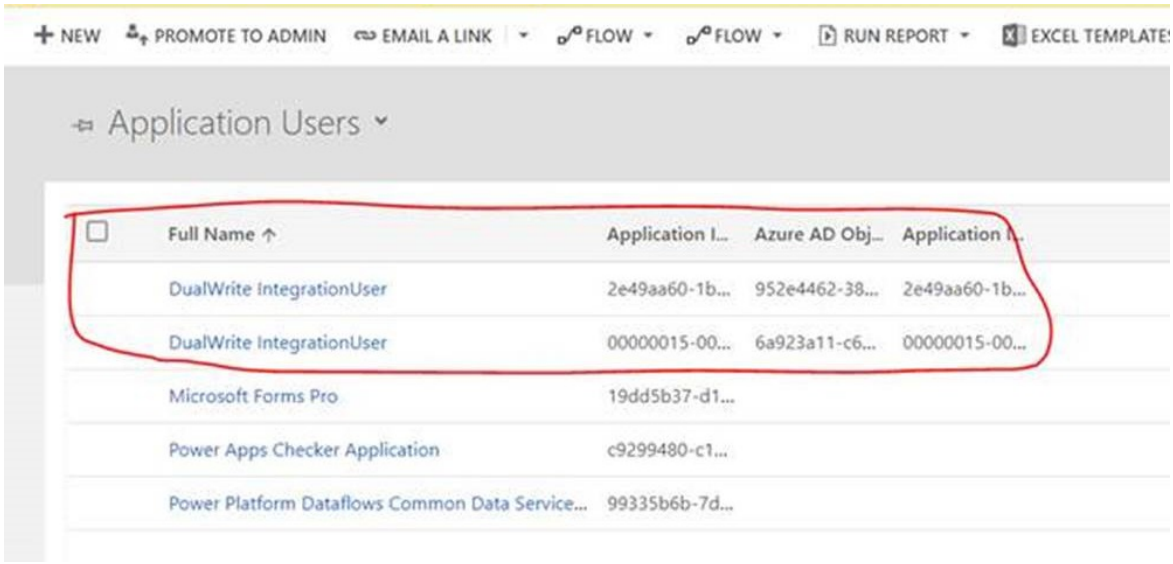
2. In Field Explorer, find **Integration user mode**, and double-click it to add it to the form. Save your change.



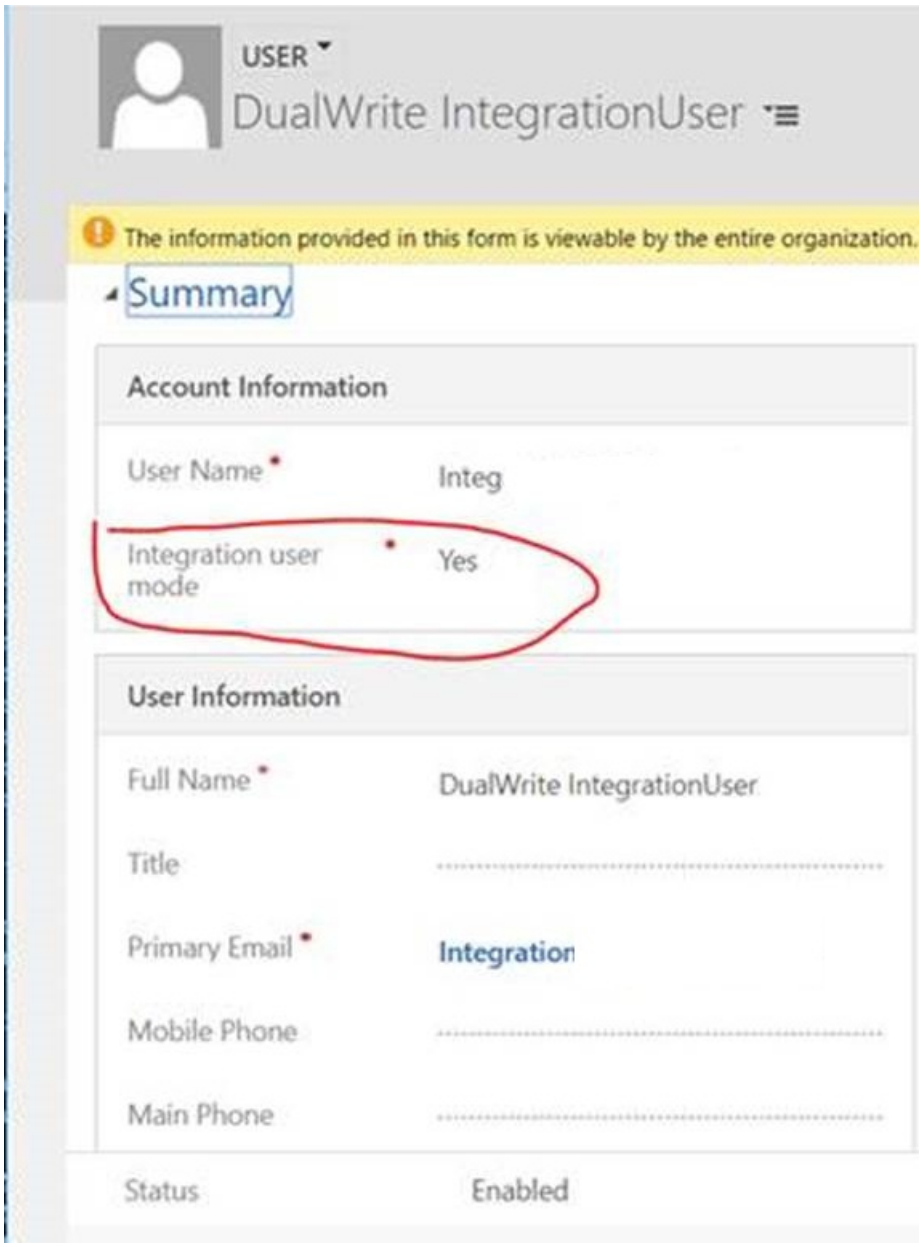
3. In Sales, go to **Setting > Security > Users**, and change the view from **Enabled Users** to **Application Users**.



4. Select the two entries for **DualWrite IntegrationUser**.



5. Change the value of the **Integration user mode** column to **Yes**.



Your sales orders are now mapped.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

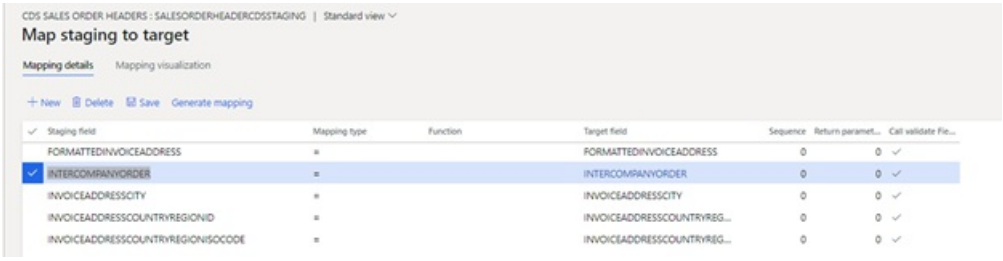
Filter intercompany orders to avoid syncing Orders and OrderLines

2/18/2021 • 2 minutes to read • Edit Online

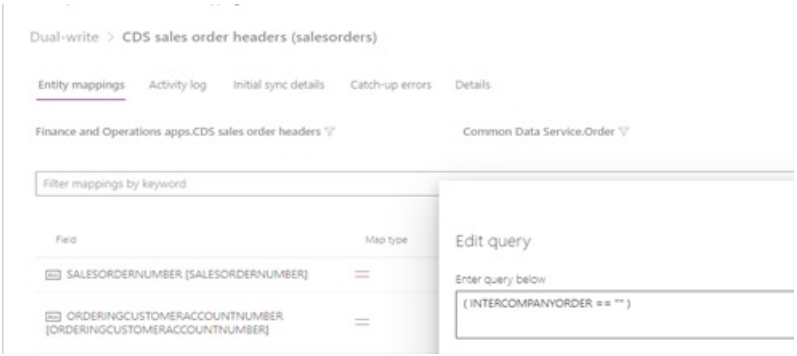
You can filter intercompany orders so that the **Orders** and **OrderLines** tables aren't synced. In some scenarios, the intercompany order details aren't required in a customer engagement app.

Each standard Dataverse table is extended through references to the **IntercompanyOrder** column, and the dual-write maps are modified so that they refer to the additional columns in the filters. Therefore, the intercompany orders are no longer synced. This process helps prevent unnecessary data in the customer engagement app.

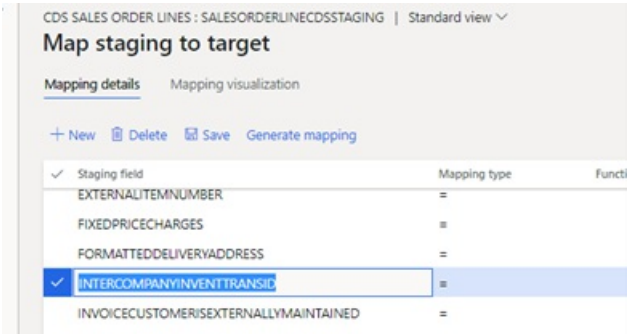
1. Extend the **CDS Sales Order Headers** table by adding a reference to the **IntercompanyOrder** column. This column is filled in only on intercompany orders. The **IntercompanyOrder** column is available in the **SalesTable** table.



2. After **CDS Sales Order Headers** is extended, the **IntercompanyOrder** column is available in the mapping. Apply a filter that has `INTERCOMPANYORDER == ""` as the query string.

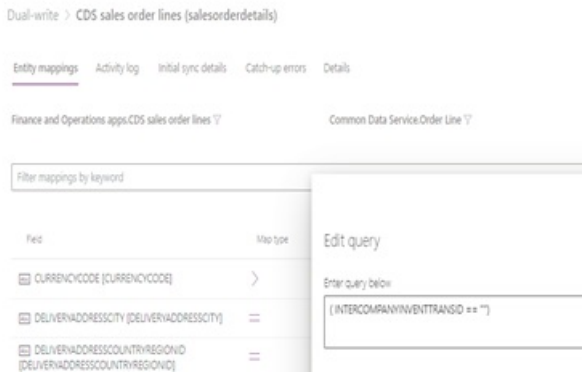


3. Extend the **CDS Sales Order Lines** table by adding a reference to the **IntercompanyInventTransId** column. This column is filled in only on intercompany orders. The **InterCompanyInventTransId** column is available in the **SalesLine** table.

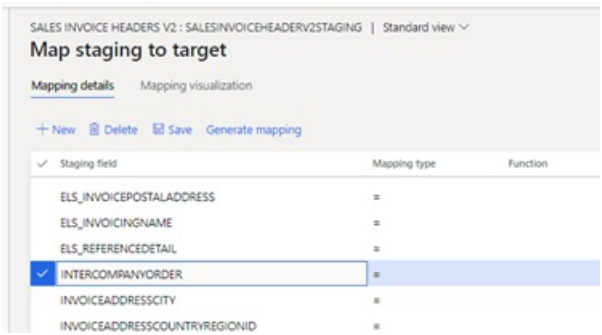


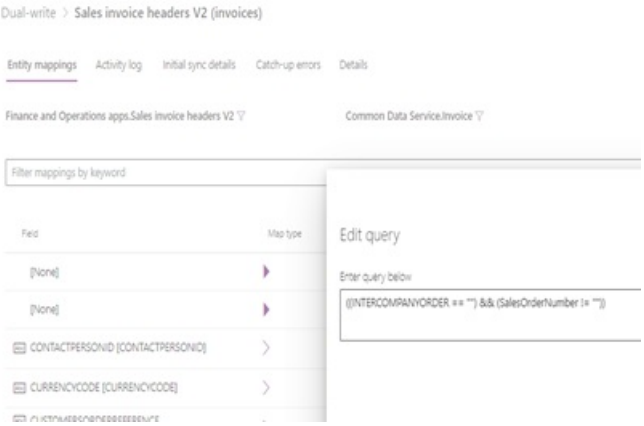
4. After **CDS Sales Order Lines** is extended, the **IntercompanyInventTransId** column is available in the

mapping. Apply a filter that has `INTERCOMPANYINVENTTRANSID == ""` as the query string.

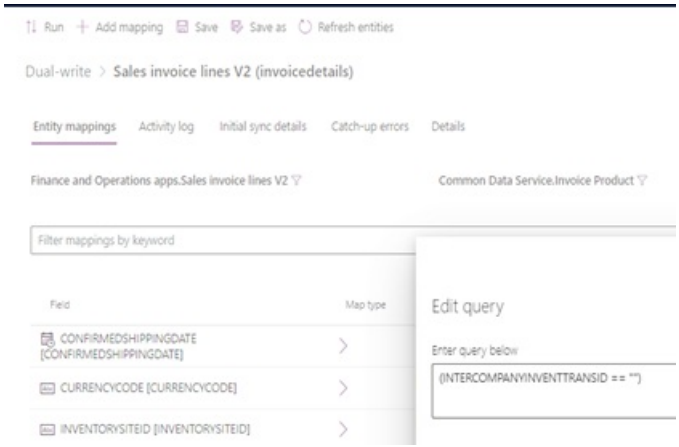


5. Repeat steps 1 and 2 to extend the **Sales Invoice Header V2** table and add a filter query. In this case, use `(INTERCOMPANYORDER == "") && (SALESORDERNUMBER != "")` as the query string for the filter.



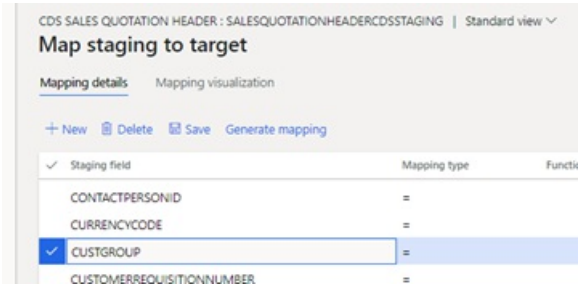


6. Repeat steps 3 and 4 to extend the **Sales Invoice Lines V2** table and add a filter query. In this case, use `INTERCOMPANYINVENTTRANSID == ""` as the query string for the filter.

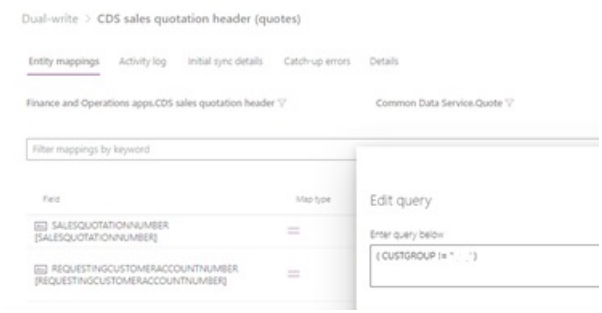


7. The **Quotations** table doesn't have an intercompany relationship. If someone creates a quotation for one of your intercompany customers, you can use the **CustGroup** column to put all those customers into

one customer group. You can extend the header and lines by adding the **CustGroup** column, and then filter so that the group isn't included.



8. After **Quotations** is extended, apply a filter that has `CUSTGROUP != "<company>"` as the query string.



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Customize table and column mappings

2/18/2021 • 3 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

The out-of-box table maps have predefined table and column mappings that enable the flow of data between two apps. In this way, they serve as "blueprints." However, because every business is different, the default table maps might sometimes not be enough. Therefore, dual-write fully supports customization by providing ways to change table maps and column mappings.

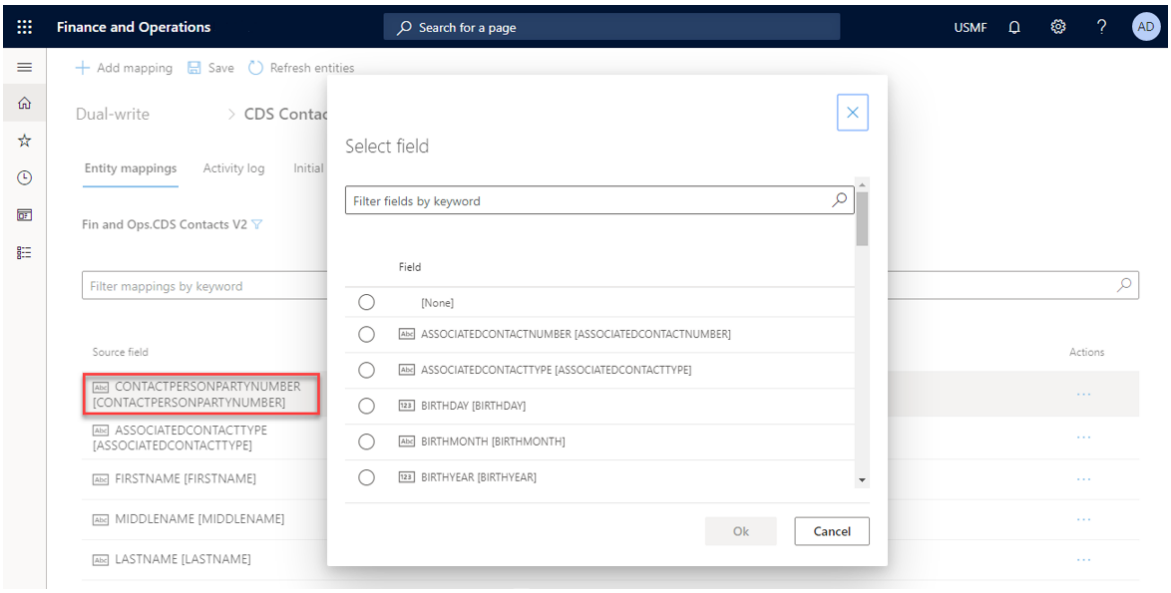
Customize column mappings, add transforms, and enable filtering

1. In your Finance and Operations app, on the **Dual-write** page, on the **Table mappings** tab, select the table map to customize.

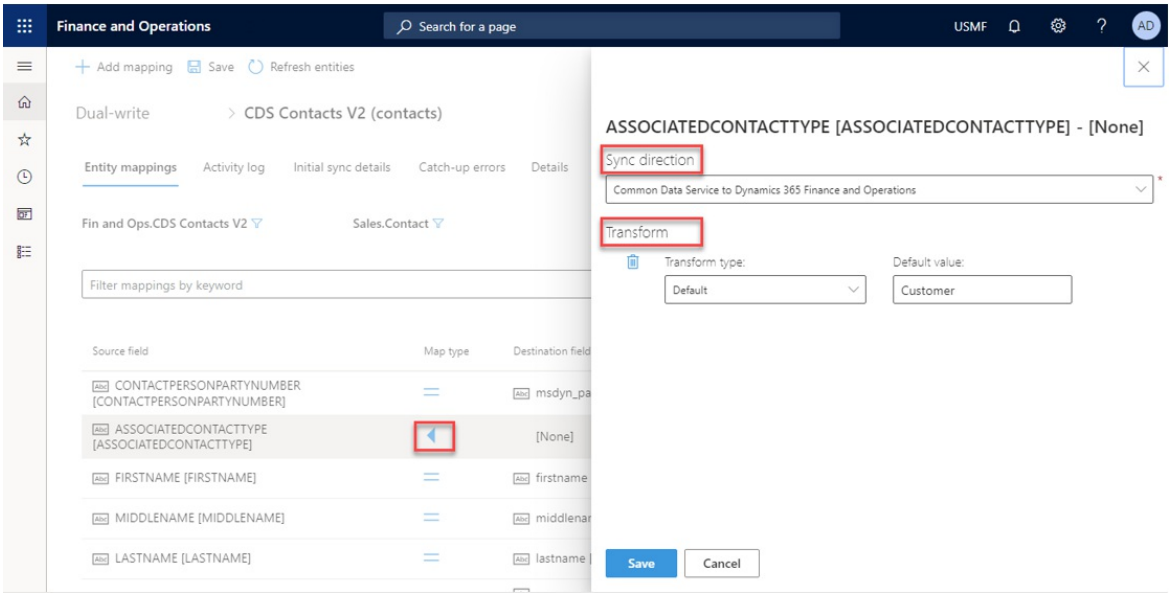
   NOTE

   Before you change table mappings, they must be stopped (not running). Otherwise, your changes won't be saved.

2. On the **Table mappings** tab, you can customize a column by selecting a new or custom column from either the Finance and Operations app or Dataverse.



3. You can customize the synchronization direction (unidirectional or bidirectional) and add transforms by selecting the map type.



The following table describes the available synchronization directions.

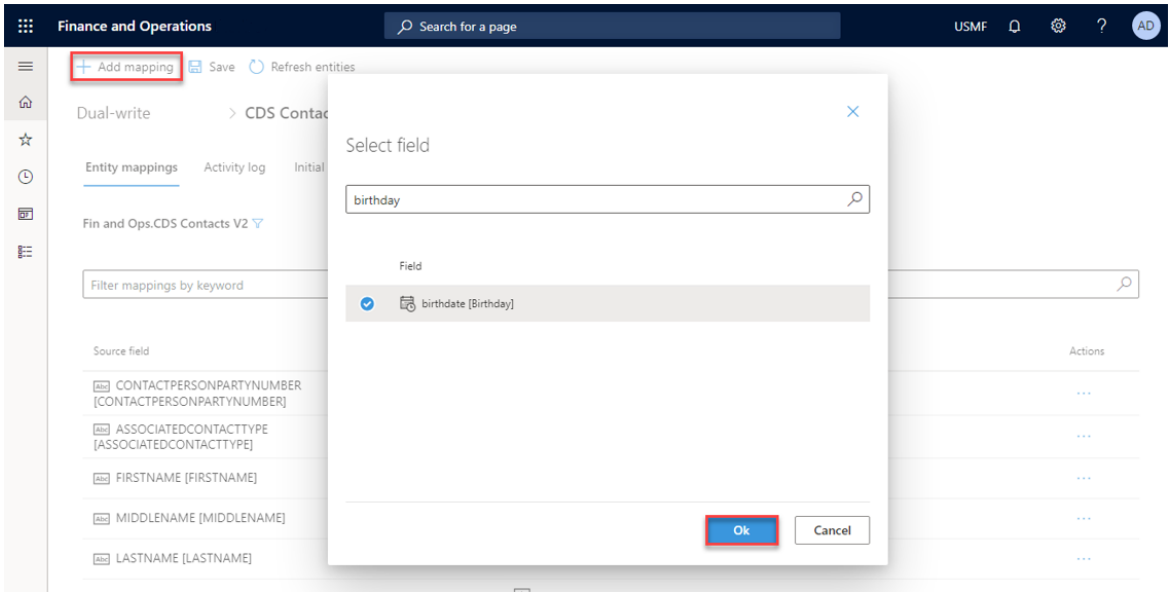
SYMBOL	DESCRIPTION
=	Bidirectional column assignment
><	Bidirectional column assignment that uses transforms
>	Unidirectional column assignment (left to right)
<	Unidirectional column assignment (right to left)
▶	Unidirectional column assignment that uses transforms (left to right)
◀	Unidirectional column assignment that uses transforms (right to left)

The following table describes the available transform types.

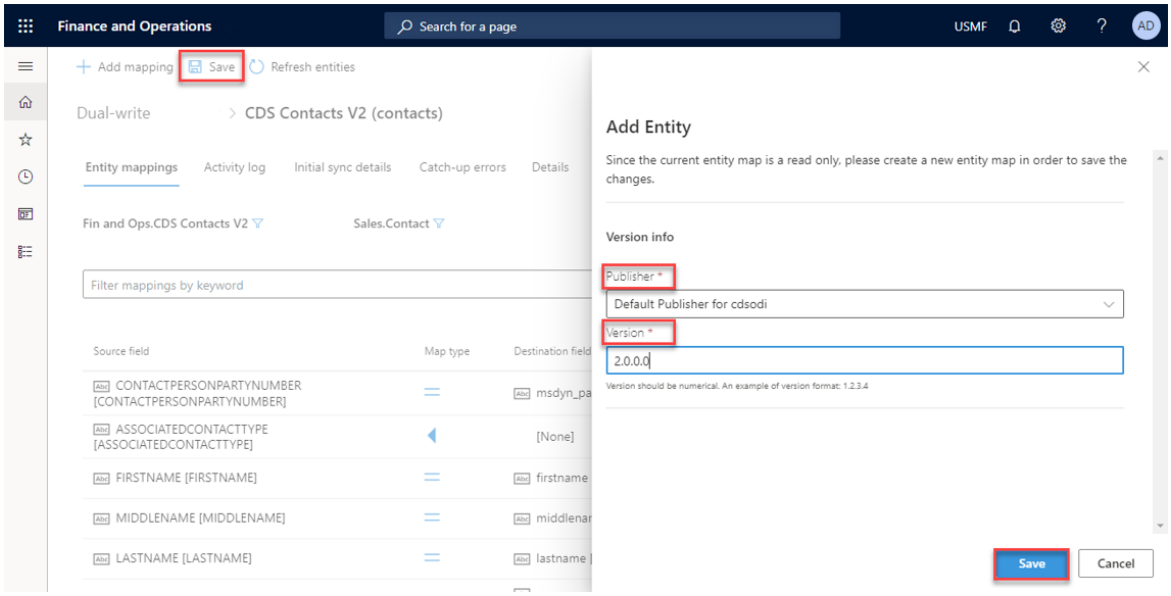
TRANSFORM TYPE	DESCRIPTION
Default	Default values are values that are applied to destination columns when no source column value is available. Use default values for columns that are required on the destination table when you have no corresponding source column.
Value map	Value maps define how values that are present in one table should be mapped to values in the other table.

4. You can add a new column by selecting **Add mapping** and then selecting an existing or custom column in the list.

   The following illustration shows an example where a new **birthdate** column is being added.



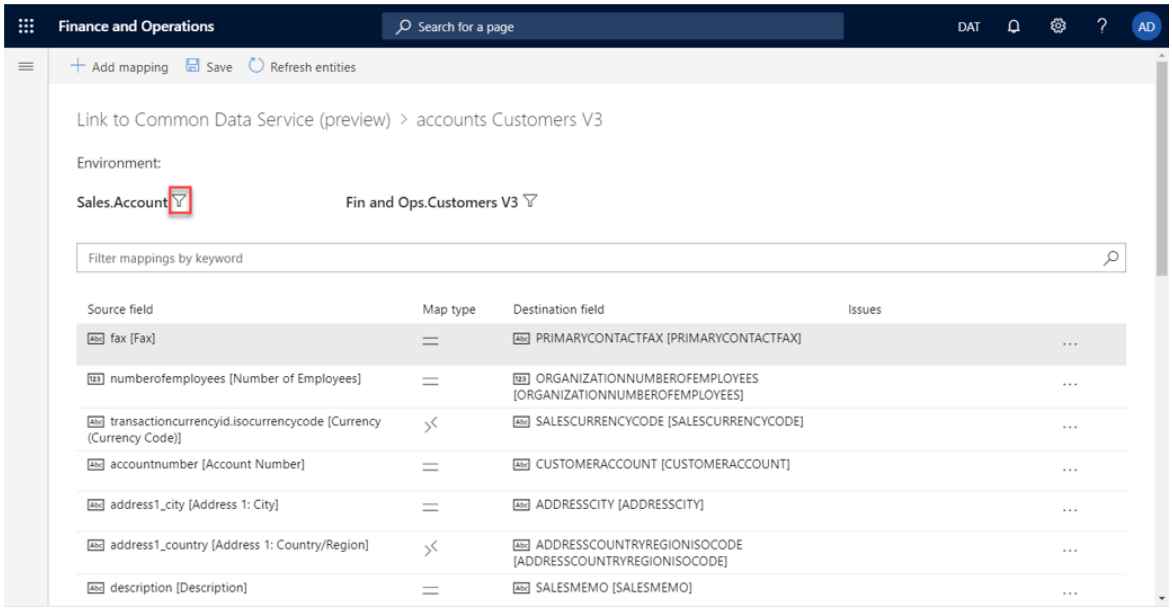
5. When you've finished customizing the column mappings, select **Save**. Then follow the prompts to specify a publisher and a version number.



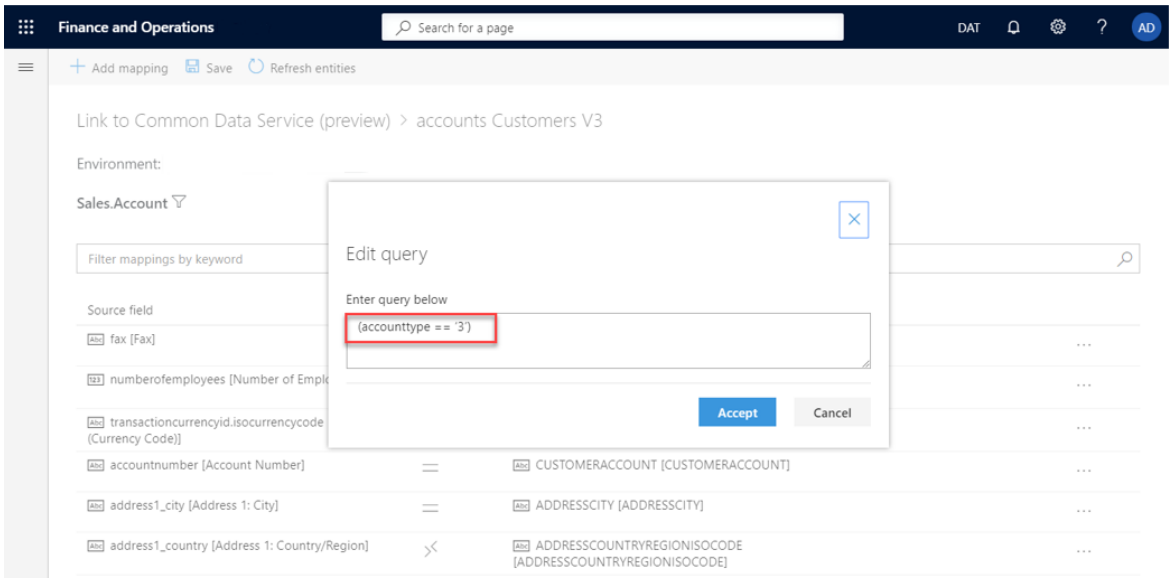
Filter your data

Dual-write lets you filter data by using Open Data Protocol (OData) filter expressions for Dataverse. For the Finance and Operations app, filtering resembles range expressions that are used in the query range.

1. On the table mapping page, select the filter button (funnel symbol).



2. In the **Edit query** dialog box, specify your filters. In this example, the filter that is specified will return only accounts where the account type equals **3**.



The following table shows some examples of filter expressions.

DATAVERSE	FINANCE AND OPERATIONS APPS
Accounttype eq '3'	(accounttype == '3')
numberofemployees gt 1000 and numberofemployees le 2000	((numberofemployees > 1000) && (numberofemployees <= 2000))

For more examples that show how to use expressions in query ranges, see Using Expressions in Query Ranges.

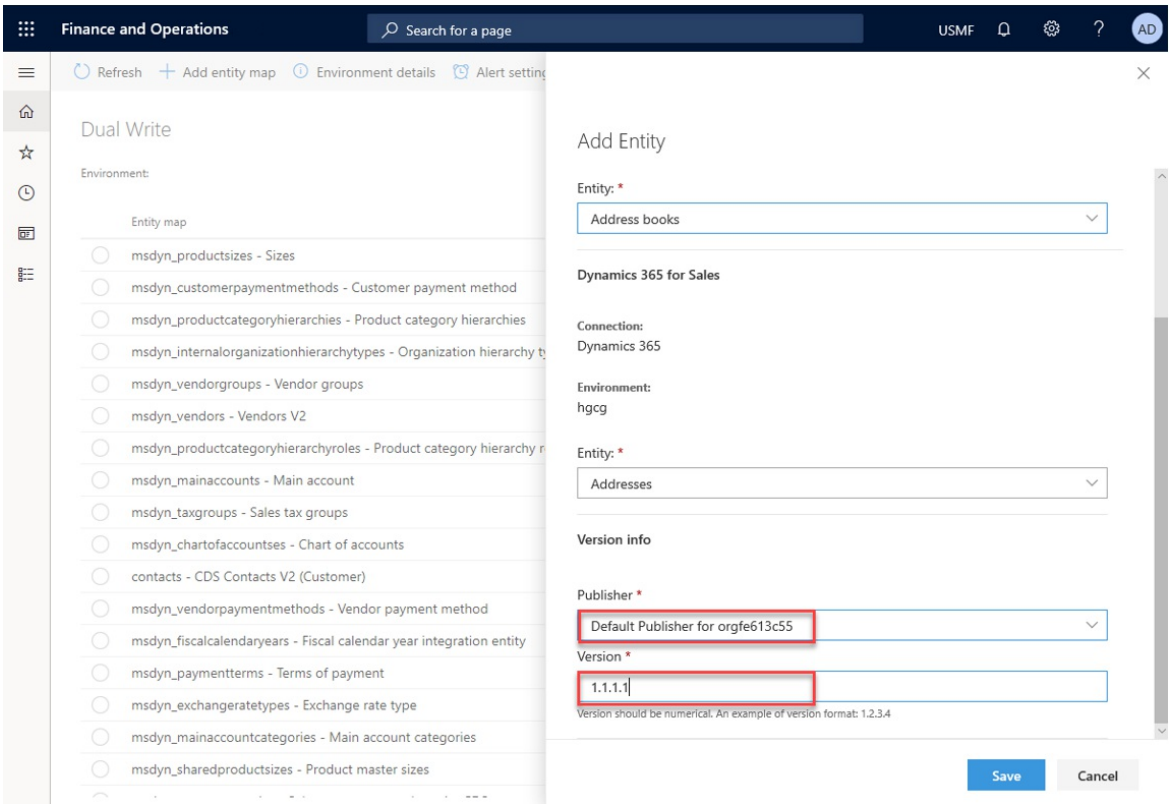
Currently, we do not support nested lookups in dual-write source filter. Only standard filter operators directly against table columns are supported. For more examples, see Standard filter operators.

Add new table maps

Although Microsoft is continuing to add new tables, you can also add standard or custom table maps.

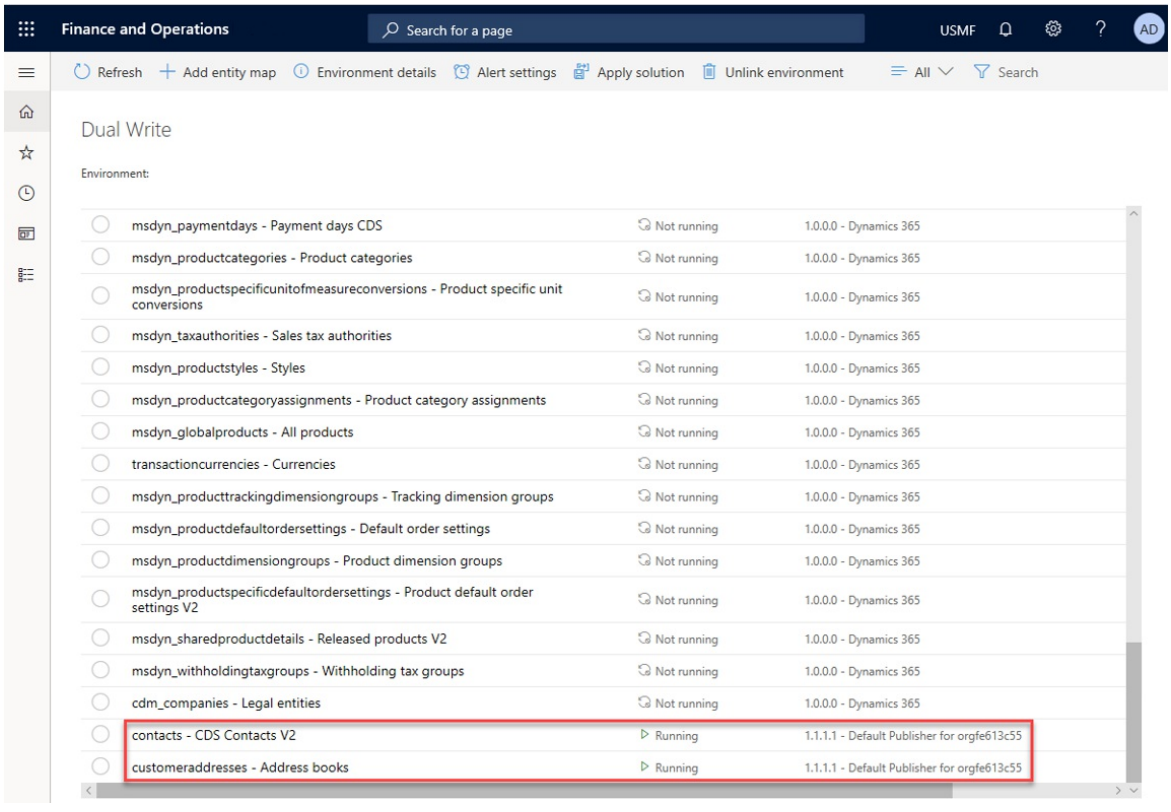
The following example shows how to add a new table map that is named **Address books**.

1. In the Finance and Operations app, on the **Dual-write** page, select **Add table map**.



> **NOTE**
>
> When you create a new solution that uses these modified table maps, you must specify the same publisher.

2. Confirm the table maps that you just modified and added. Be sure to enable and test them, to ensure that they work as you expect.



Next steps

Error management and alert notifications

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

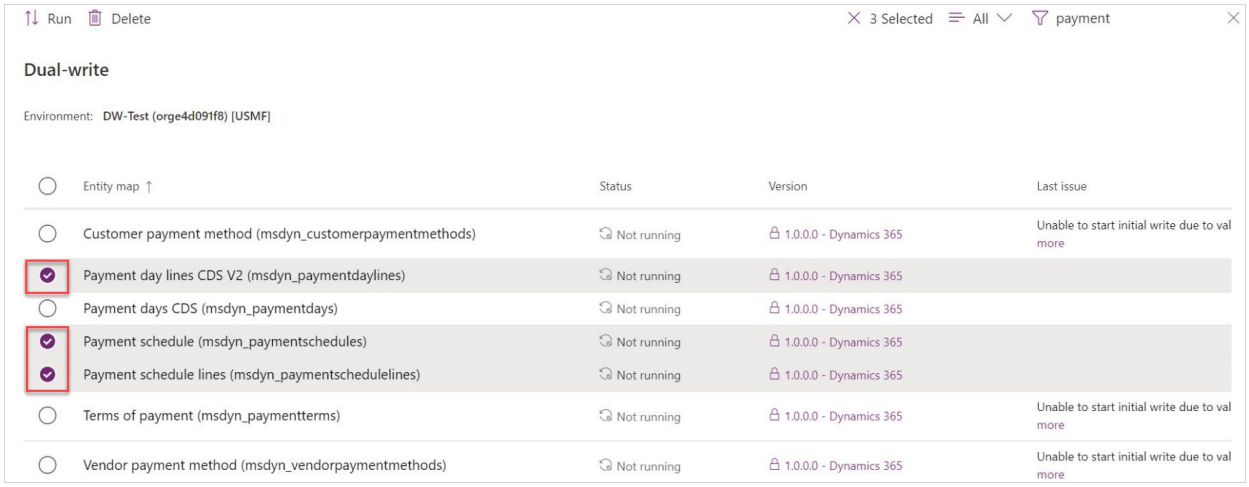
The survey will take about seven minutes. No personal data is collected (privacy statement).

Manage multiple table maps

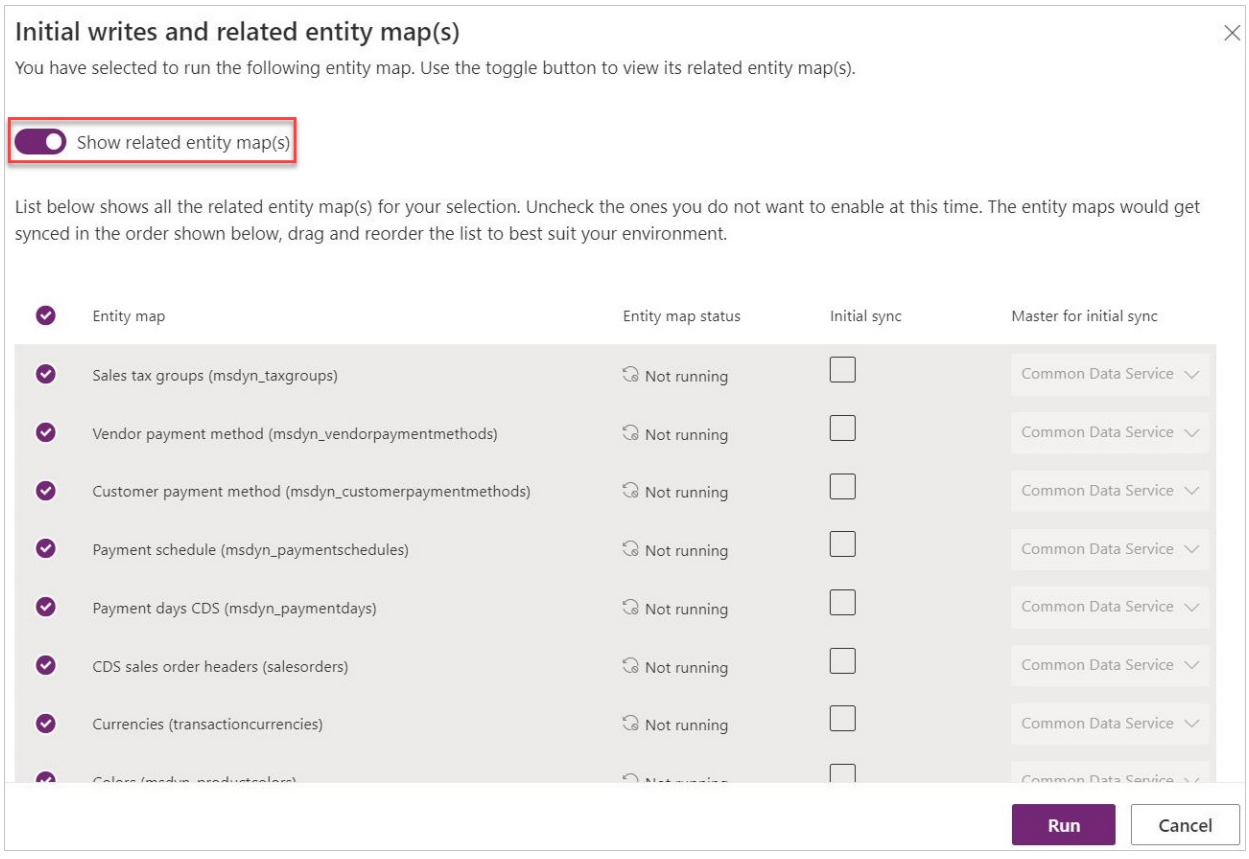
2/18/2021 • 2 minutes to read • Edit Online

This article describes how to select multiple table maps, view a list of dependent table maps, enable the table maps and all of its related tables, and copy pre-existing data.

As part of day to day operations, there may be a need to bulk handle table maps. For example, you may want to simultaneously enable or pause a set of table maps. Instead of doing this one by one, which is cumbersome and time consuming, you can now enable, pause, resume, or stop more than one table map at the same time in the dual-write list page.



As part of enabling multiple table maps, you also get to view the list of all the dependent table maps by selecting **Show related table map(s)**.



To enable the selected table map and all its related tables, select **Run**. If you want to copy the pre-existing data

for the selected table maps or its dependents, select the corresponding **Initial sync** check box. Alternatively, remove one or more of the related tables by clearing the corresponding check box. You can also drag and drop the table maps to change the order in which the maps will be synced.

Edit a legal entity after dual-write setup

2/18/2021 • 2 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

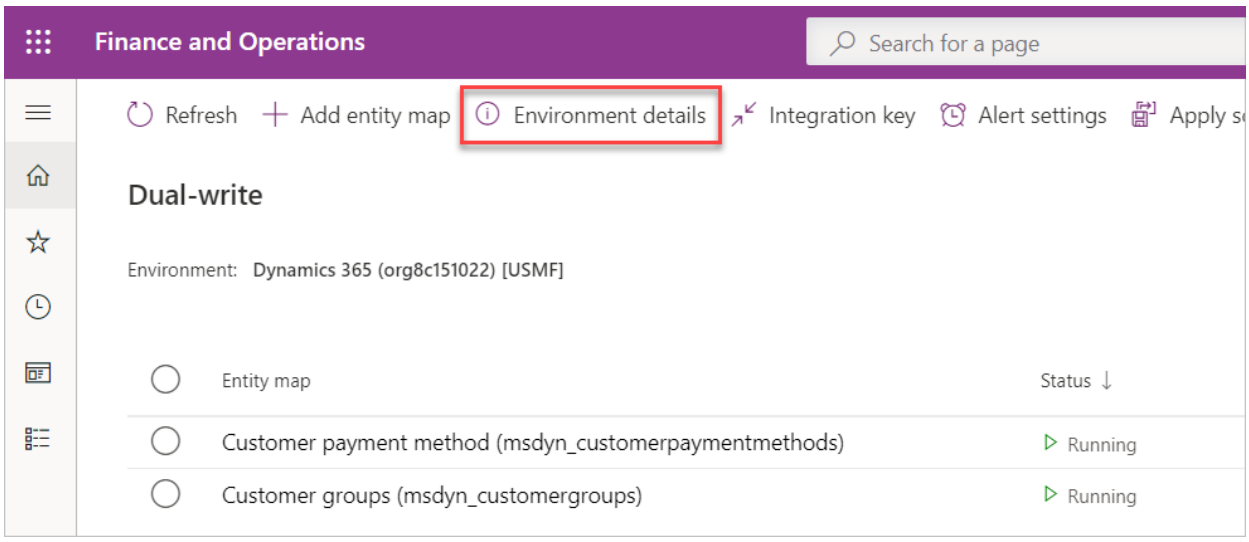
The dual-write wizard enables you to add or remove a company or legal entity after dual-write has been set up. You can do this without having to unlink and relink your dual-write environment.

The wizard enables you to link your Finance and Operations apps to Dataverse environments. As part of this wizard, you also can select one or more companies or legal entities. The company or legal entity list doesn't remain static and is constantly changing. This is because you may need to add new companies, especially as part of a phased rollout or acquisitions. Until now, you were unable to add a company or legal entity without system down-time, which required you to unlink and relink your environment. All of this can be expensive, especially because of pre-existing data. With this feature, you can add a company in a live environment without the need to unlink your existing dual-write environment.

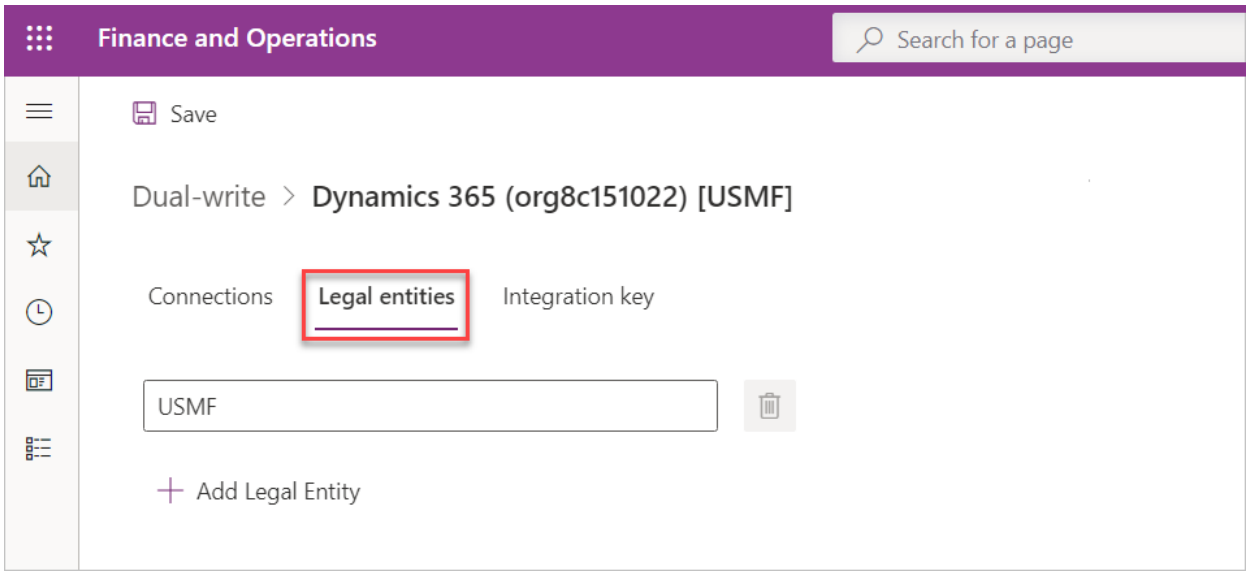
Add a company or legal entity after dual-write has been set up

Follow these steps to add a company or legal entity after dual-write has been set up.

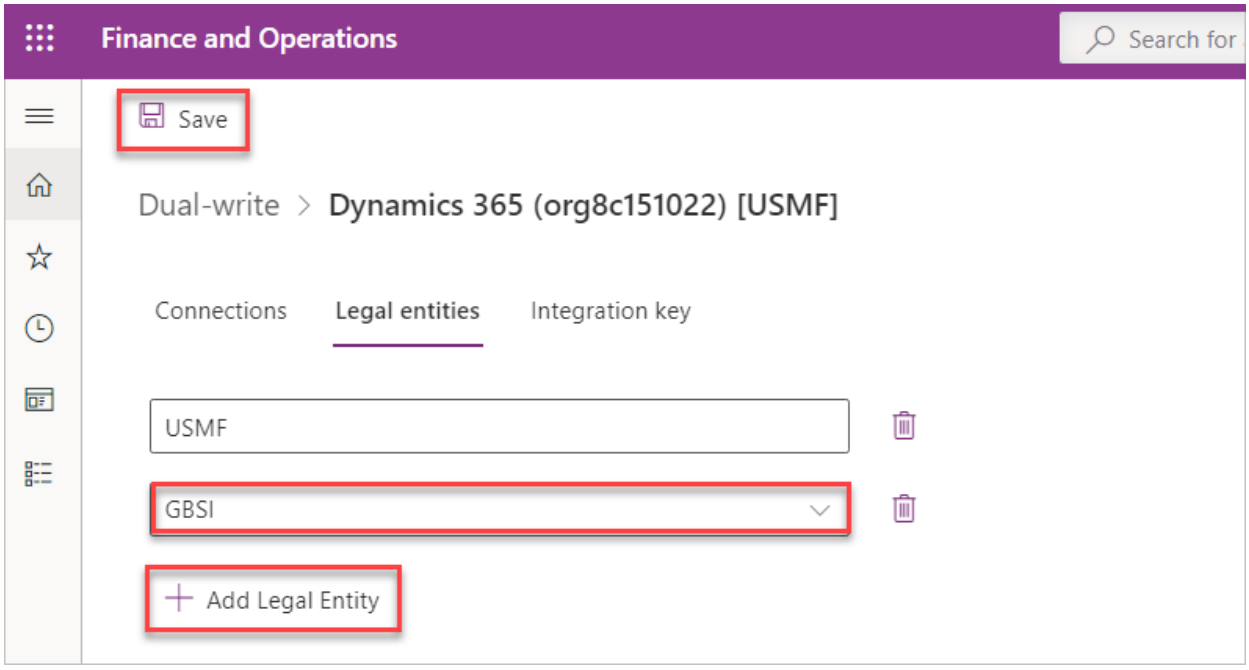
1. On the **Dual-write table map** list page, select the **Environment details** button.



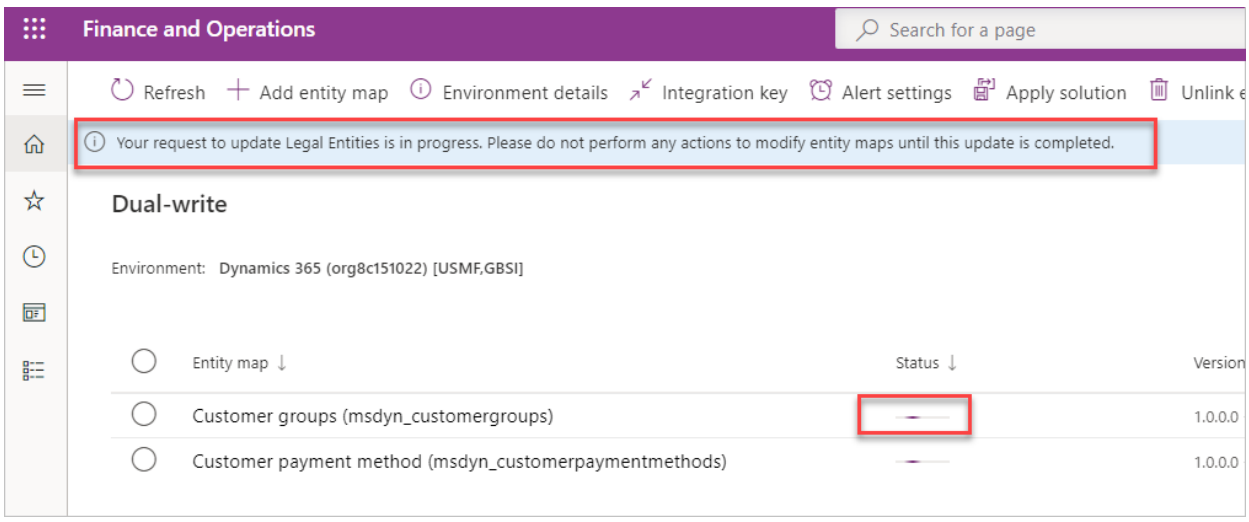
2. On the **Legal entities** tab, you see the company that you selected as part of the dual-write wizard to link environments. In this example, the company is USMF.



3. Select **Add legal entity** to add one or more companies to dual-write. In this example, GBSI. Select **Save**.



At this point, the legal entities start updating. The table maps that are currently running or paused go through the initial write process by copying pre-existing data. Until the process is completed, we recommend that you do not perform any actions to modify your table maps.

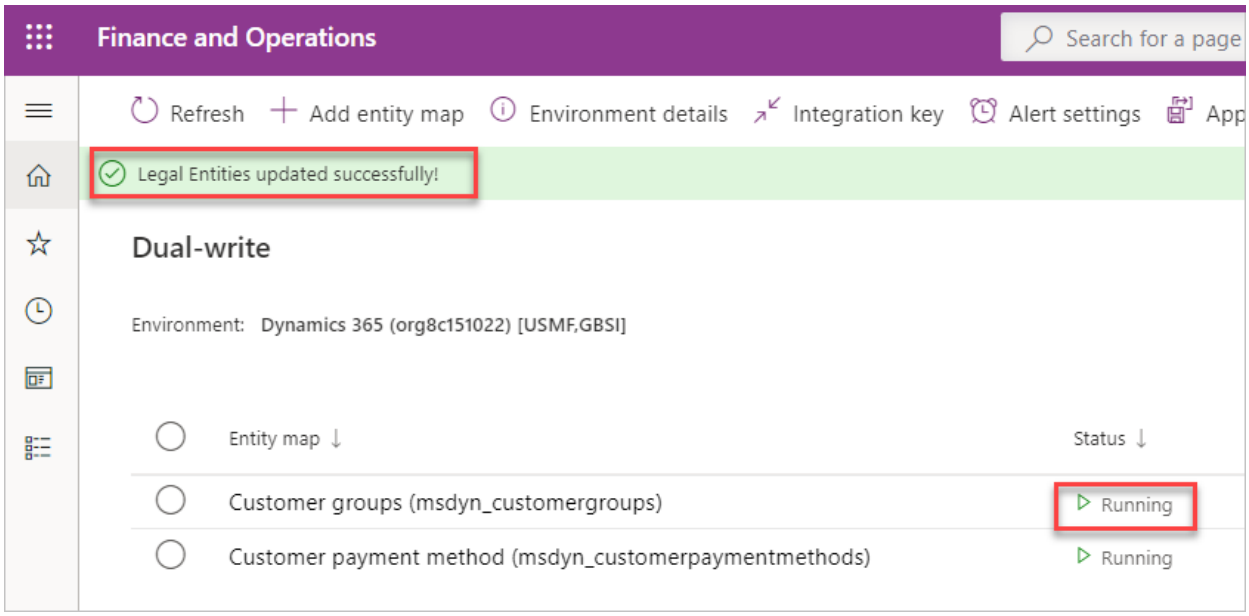


NOTE

This operation may fail if either of the following conditions are true:

- You add or remove a new company when one or more table maps is already in the Initial writes state. This the process where the system is copying pre-existing data.
- You remove a company when one or more table maps is in the Paused state.

4. After the process is complete, a banner displays informing you that the legal entities have been updated successfully. You can now resume updates to your table maps.



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Error management and alert notifications

2/18/2021 • 2 minutes to read • Edit Online

NOTE

Effective November 2020:

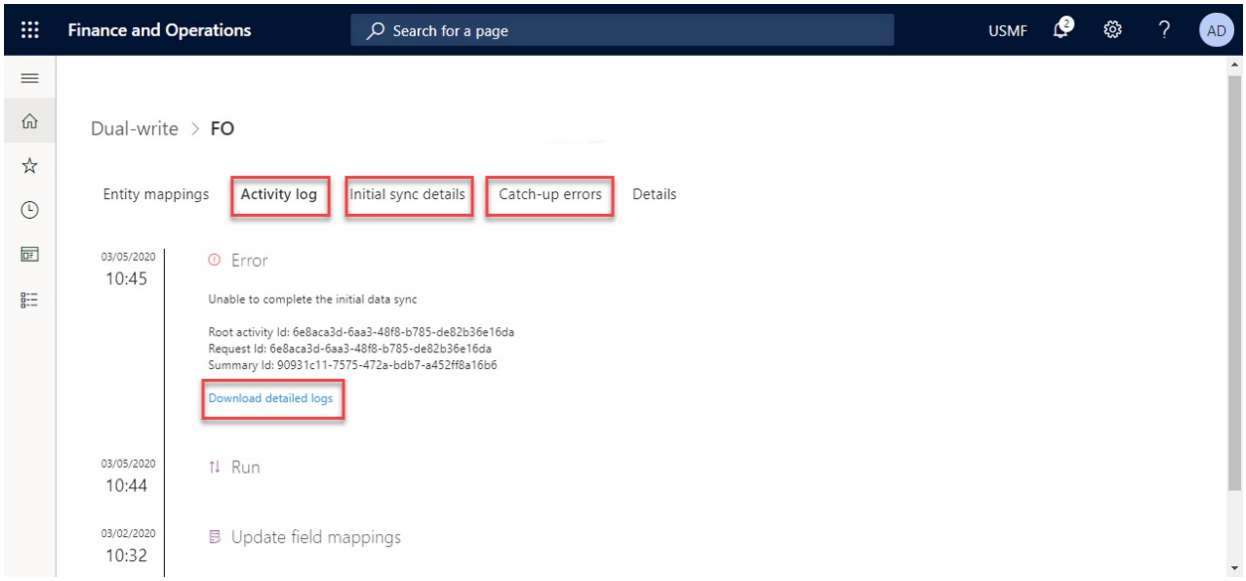
- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

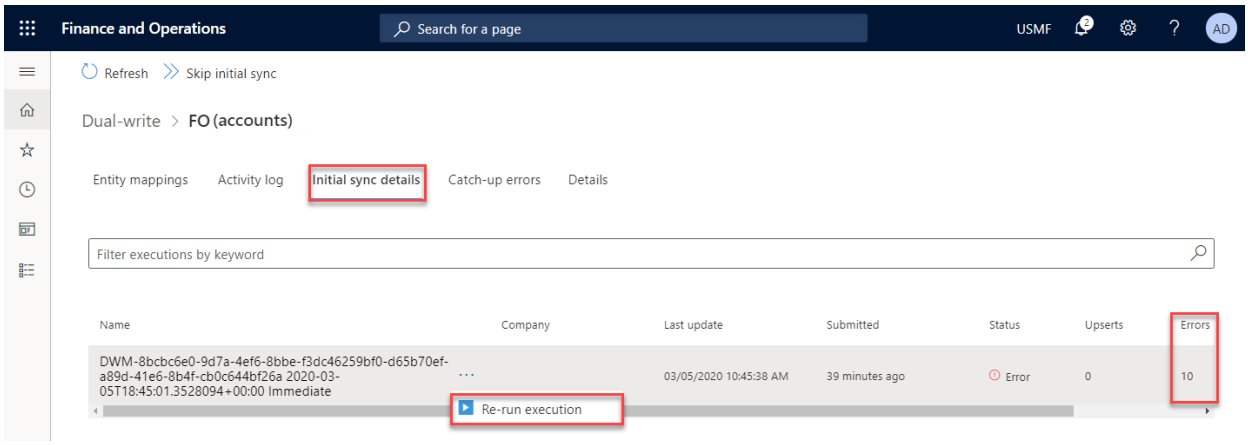
Microsoft has invested lots of time and effort into making dual-write resilient to errors. However, if you encounter an issue while or after you enable table maps for dual-write, you can select specific table maps to get a consolidated view of all the activities and errors for them. This consolidated view includes error logs. The goal is to help you during troubleshooting by providing a single view of the activities for a table map.

Consolidated error management

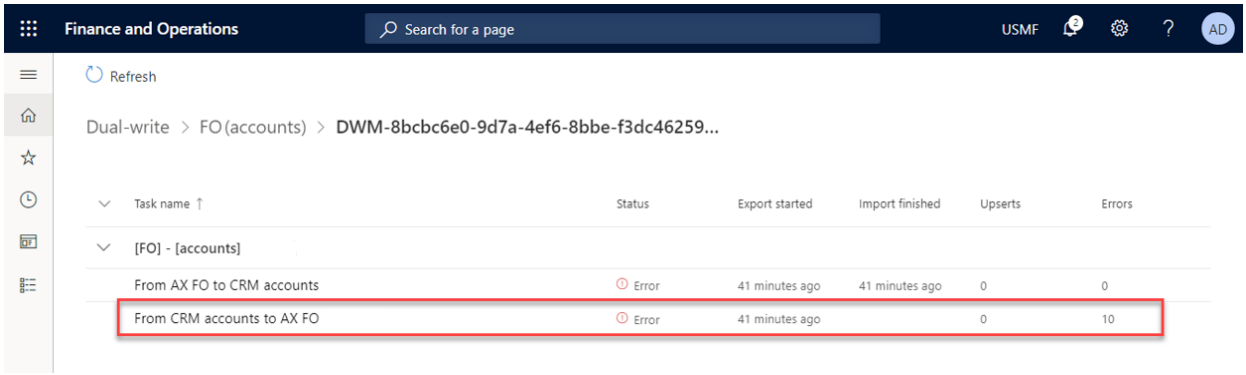
The activity log provides a chronological list of events that a specific table map goes through from the **Not Running** status to the **Running** status. For example, the list can include mappings that are created, updates of column mappings, and mappings that are run. Additionally, if errors occur, you can download the logs to get the next level of details.



If you encounter issues while you copy pre-existing data between Finance and Operations apps and Dataverse, the **Initial sync details** tab provides a count of the errors. It also lets you rerun the execution after you fix the underlying errors.



You can drill down further to view the synchronization direction where the error occurred. This information can help you narrow down the scope for troubleshooting.

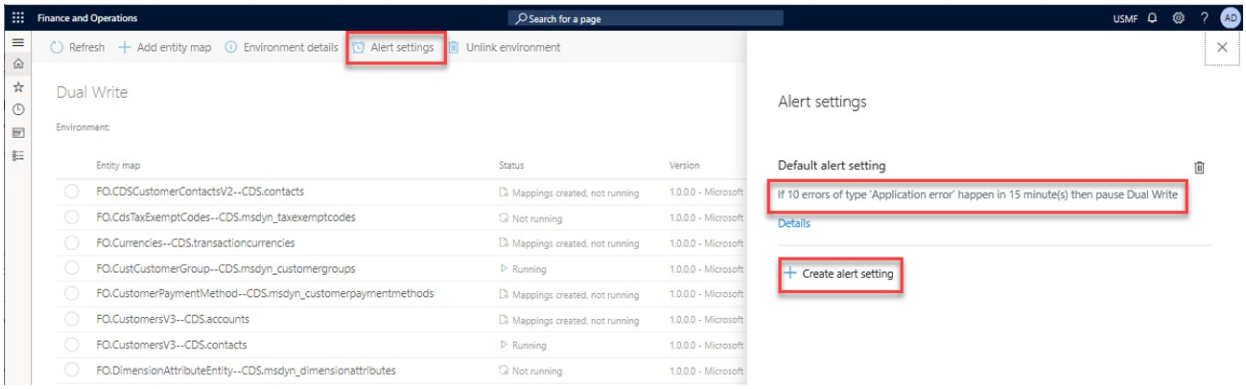


In a similar way, the **Catch-up errors** tab can help you troubleshoot issues when you resume from a paused state.

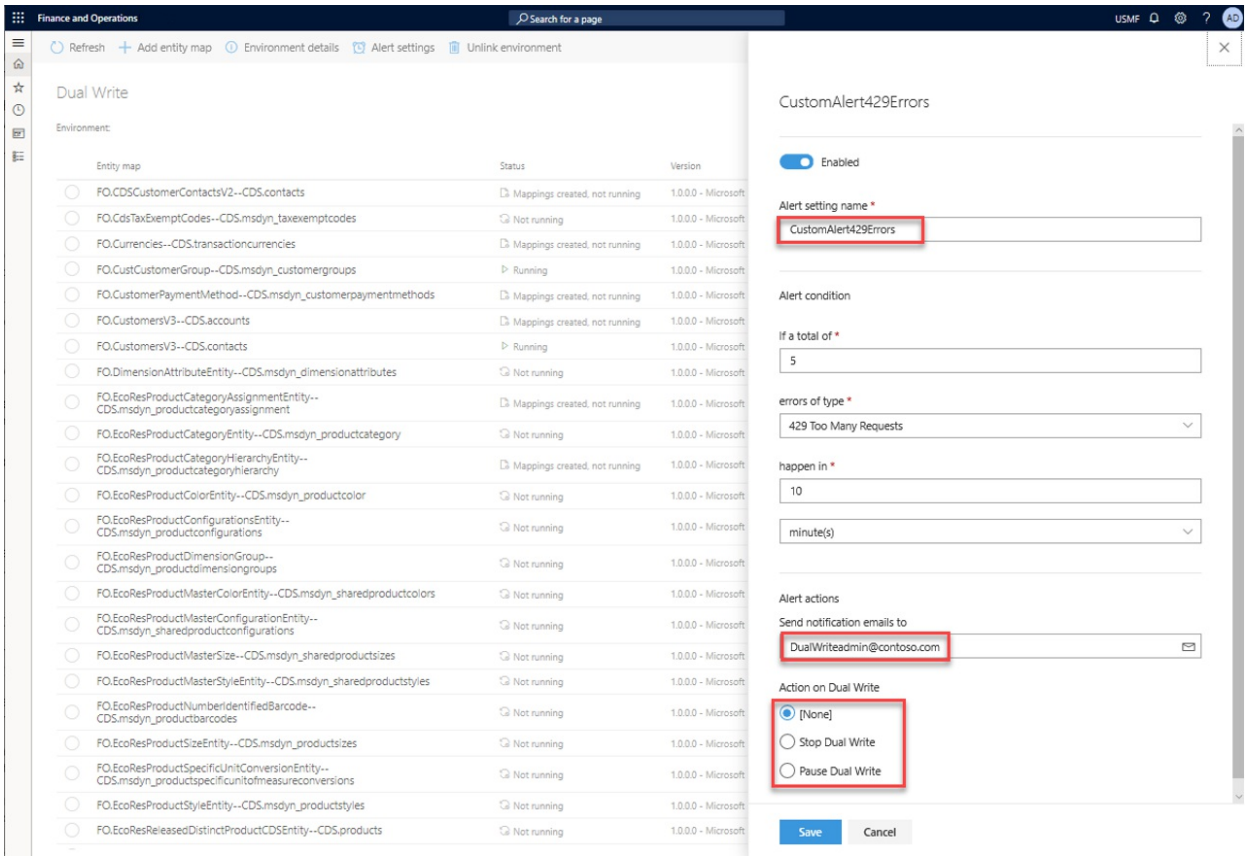
Alert notifications

As an admin, you can create one or more alert settings to handle cases of planned or unplanned maintenance. For example, you can set up the dual-write system to notify you by email if a specific error threshold is reached because of, for example, network errors. The dual-write system can also take action on your behalf. For example, it can pause or stop dual-write.

The following illustration shows an example where dual-write will be paused if 10 errors of the **Application error** type occur within 15 minutes.



By selecting **Create alert settings**, you can create more alerts. You can also select whether notifications should be sent to an individual or a group, and whether the dual-write system should take any action on your behalf.



This feature is especially useful if there is unplanned maintenance. For example, one of the apps becomes unavailable and, based on your defined thresholds, dual-write goes into a paused state where all new requests are queued (that is, they aren't lost). After you fix the underlying issue, and both apps are running smoothly, you can resume from the paused state. The updates will then be read back from the queue and written to the recovered app.

Next steps

Application lifecycle management

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Application lifecycle management

2/18/2021 • 7 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

By making dual-write solution-aware, you enable basic application lifecycle management (ALM) capabilities, such as transportation and backup/restore of dual-write table maps across environments. You also enable scenarios where you can get solutions that are published by Microsoft or an independent software vendor (ISV) from AppSource.

What is a dual-write solution?

A dual-write solution can contain one or more dual-write table maps. These maps can be imported into your environment (by selecting **Solutions** in Microsoft Power Apps). They can also be exported to other environments as a package. You can import Microsoft-published or ISV-published table maps from AppSource, modify them in your test environment, test them, and then, when they are ready, export them to your production environment. Additionally, you can publish your solution through AppSource, so that other people can use it.

There two types of solutions: managed and unmanaged.

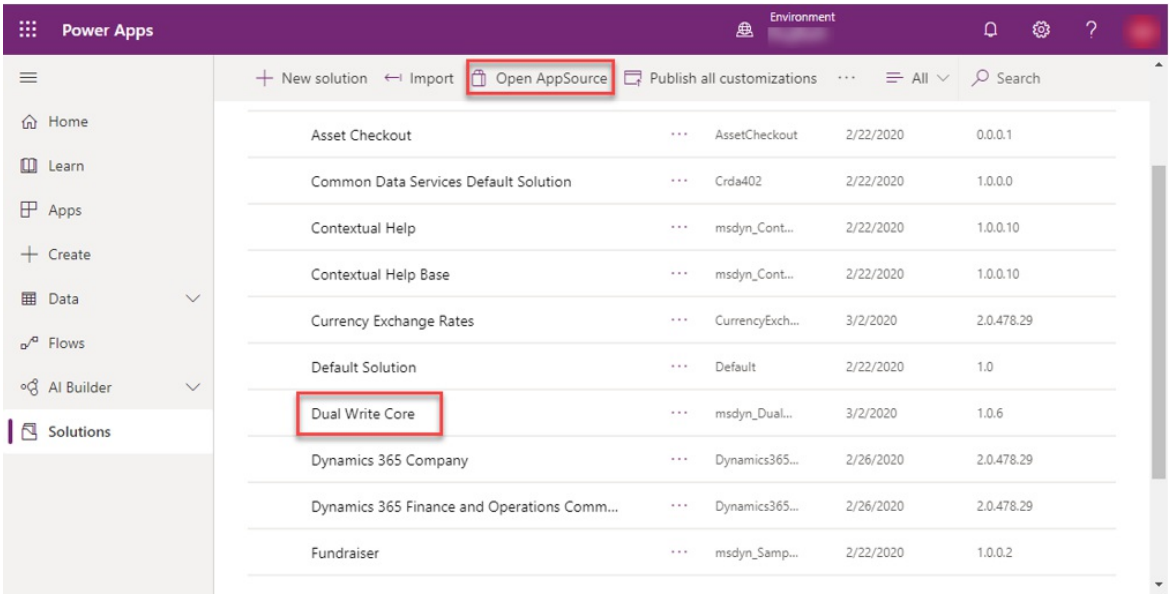
A managed solution can't be modified, and it can be uninstalled after it's imported. When you import an unmanaged solution, you add all the components of that solution into your environment. When you import an unmanaged solution that contains components that you've already customized, your customizations are overwritten by the customizations in the imported unmanaged solution.

For more information about solutions, see the solutions overview.

Install the dual-write core solution

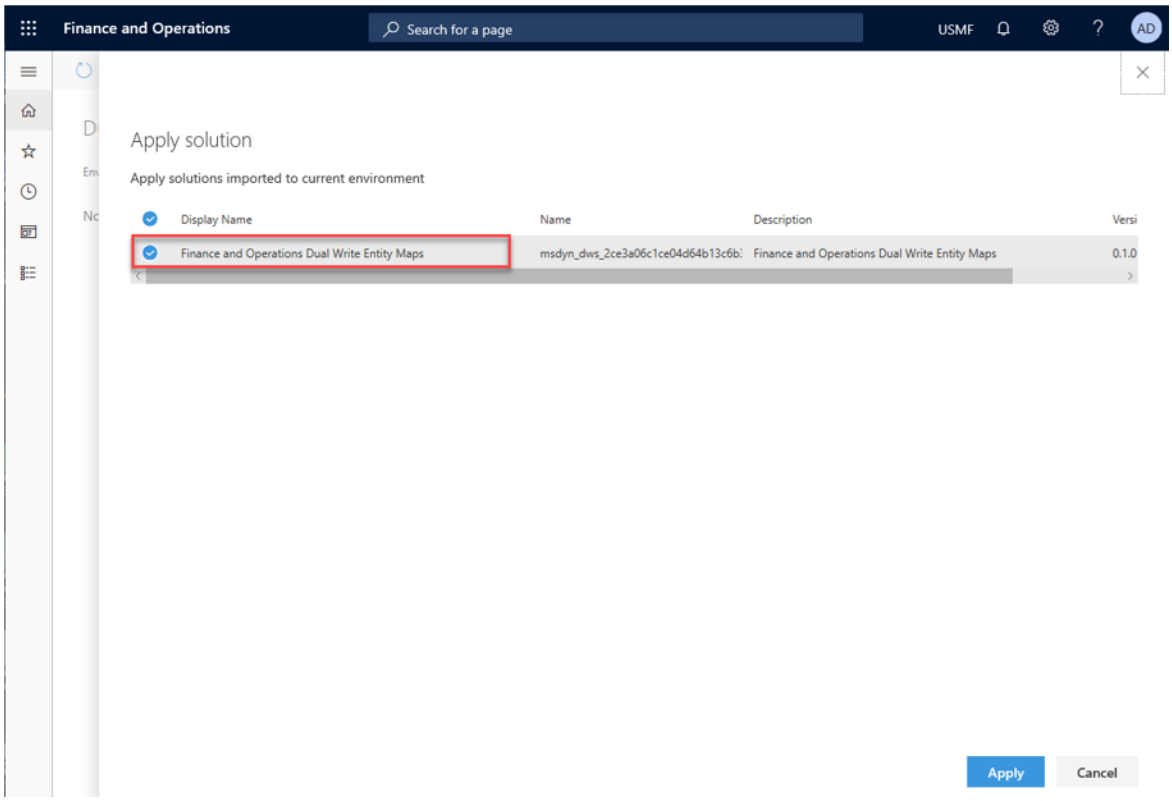
The dual-write core solution contains metadata for your table maps and must be installed in your environments.

1. In Power Apps, in the left pane, select **Solutions**.
2. Select **Open AppSource**, and search for the solution that is named **Dual Write Core**.
3. Follow the prompts to import the solution.



Install the dual-write table maps solution

1. In Power Apps, in the left pane, select **Solutions**.
2. Select **Open AppSource**, and search for the solution that is named **Dataverse Add-in for Finance and Operations package**.
3. Follow the prompts to import the solution.
4. In the Finance and Operations app, on the **Dual-write** page, select **Apply Solution** to apply the table maps that you downloaded and installed. After you apply the solution, you will see that the default table maps are published.

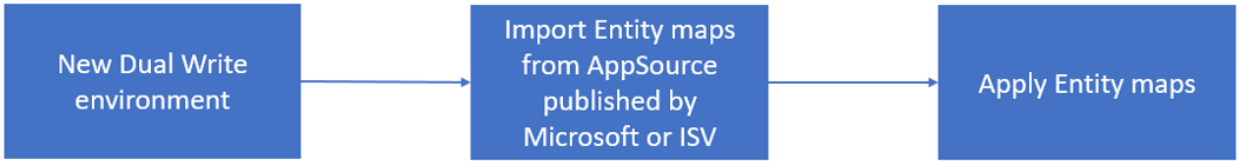


You've now successfully imported and applied a Microsoft-published dual-write table maps solution to your environment.

Import table maps through a dual-write solution and apply them to

your environment (New environments)

This section explains how to import table maps from AppSource and apply them to your environment.

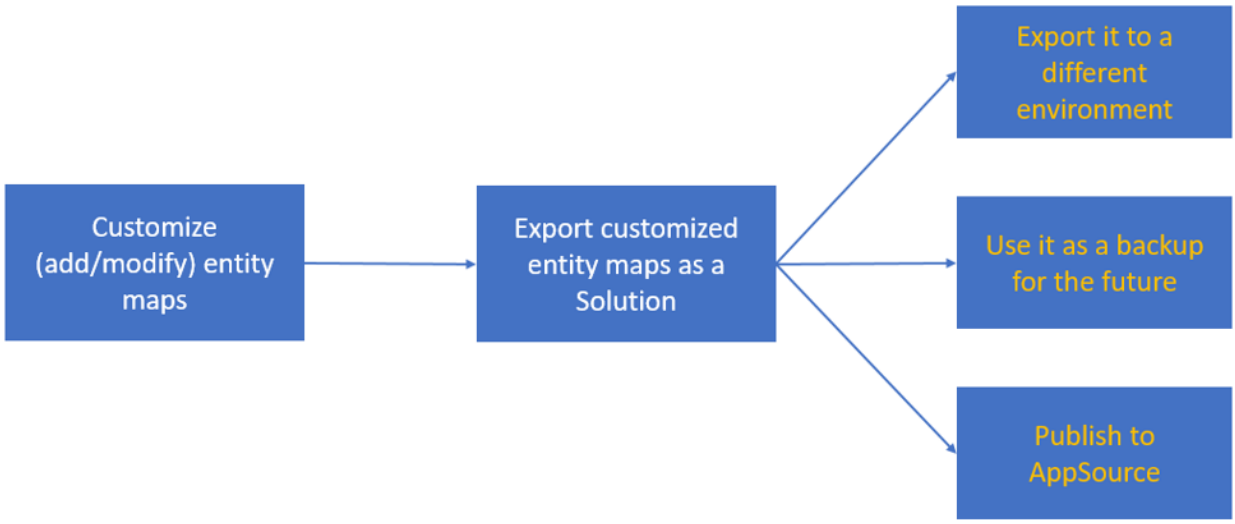


1. Import the dual-write core solution.
    a. Create a new dual-write environment (a Finance and Operations app environment and a Dataverse environment).
    b. Follow the instructions in the Install the dual-write core solution section earlier in this topic to install the dual-write core solution from AppSource in Power Apps.
    c. Verify that the dual-write core solution is listed under **Solutions** in Power Apps.
2. Import the Microsoft-published or ISV-published table maps solution.
    a. Follow the instructions in the Install the dual-write table maps solution section to download and install the Microsoft-published or ISV-published table maps from AppSource in Power Apps.
    b. Verify that the table maps solution is listed under **Solutions** in Power Apps.
3. Apply the dual-write table maps solution to your Finance and Operations app environment.

    Apply the solution that you downloaded by selecting **Apply Solutions** on the **Dual-write** page in the Finance and Operations app, as described in the Install the dual-write table maps solution section.

Update table maps and export them to other environments as a solution

This section explains how to export your customized table maps as a solution, use it as a backup, and move the artifacts across environments and/or publish them to AppSource.



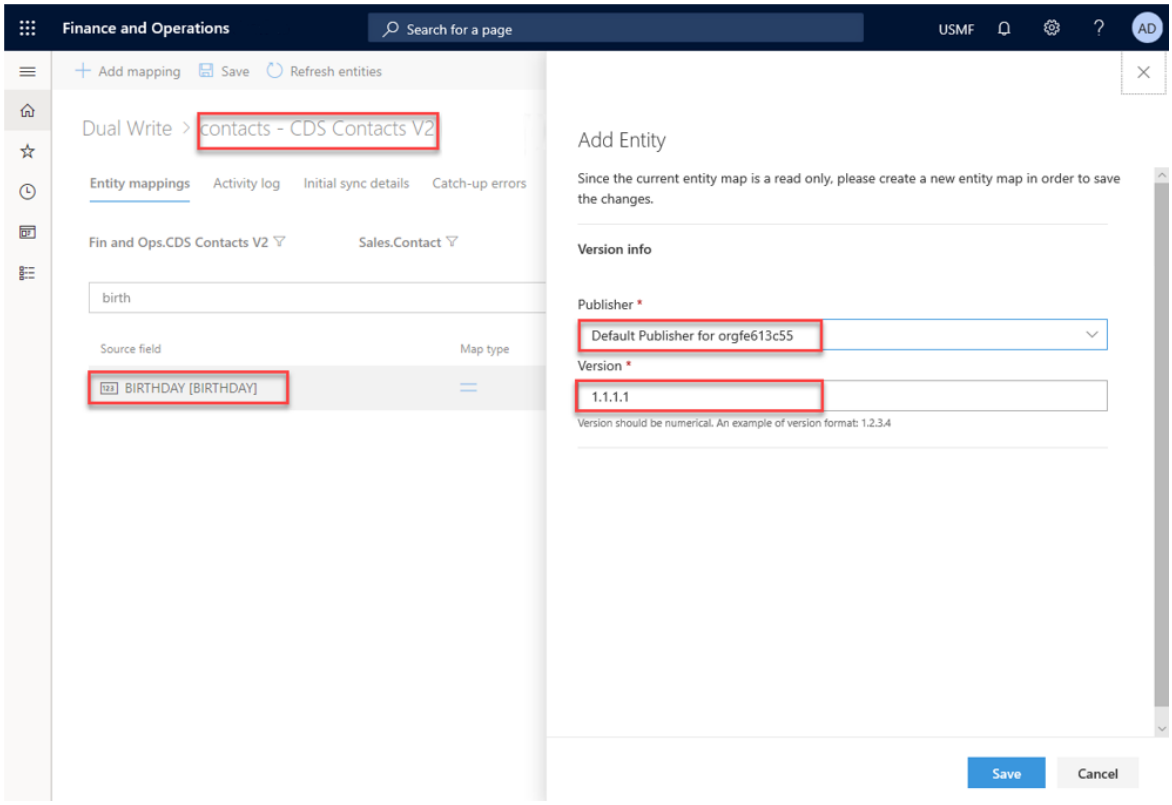
Customize your table maps

The first step is to customize your table maps by modifying existing table maps and adding a new table map.

1. In the Finance and Operations app, on the **Table mappings** tab, customize the mappings for the default table map that you just installed by using a solution. To add a new table map, select **Add Table**. In both cases, when you save the table map, you're prompted to specify the publisher and the version number.

    The following figure shows how to add a new column that is named **birthday** to the contacts - CDS

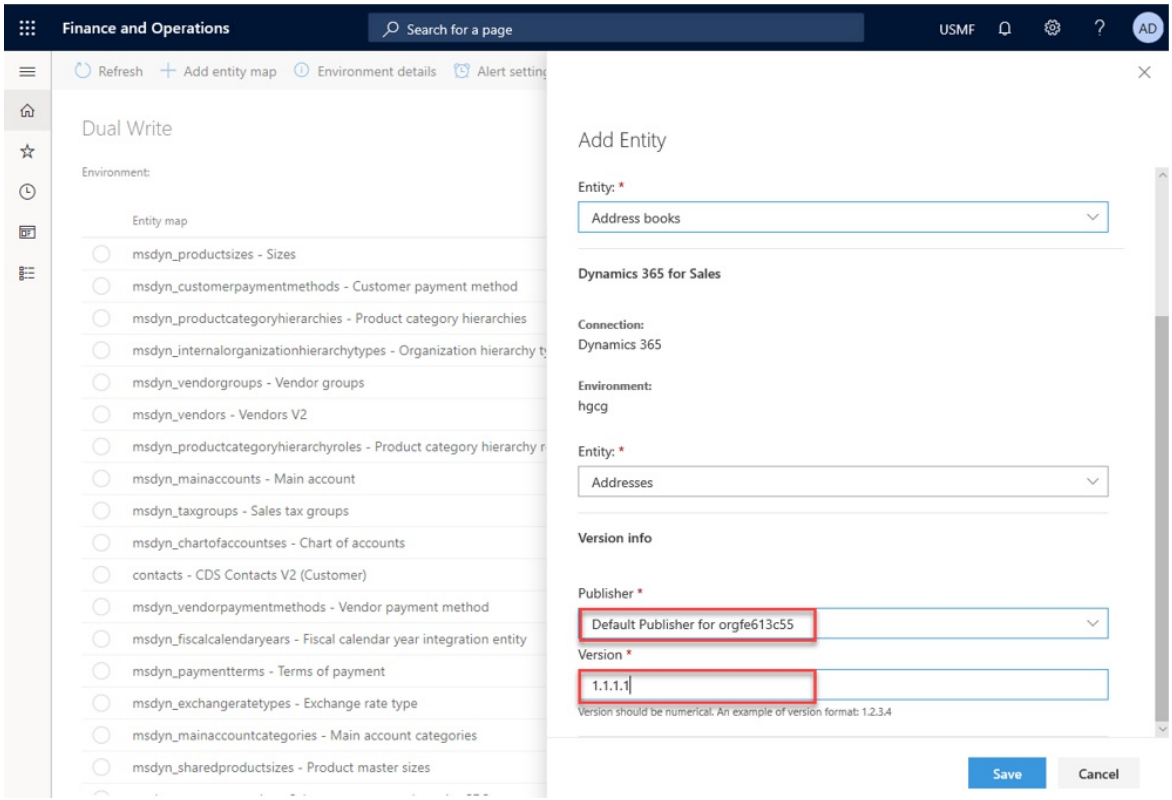
Contacts V2 table map and select the default publisher.



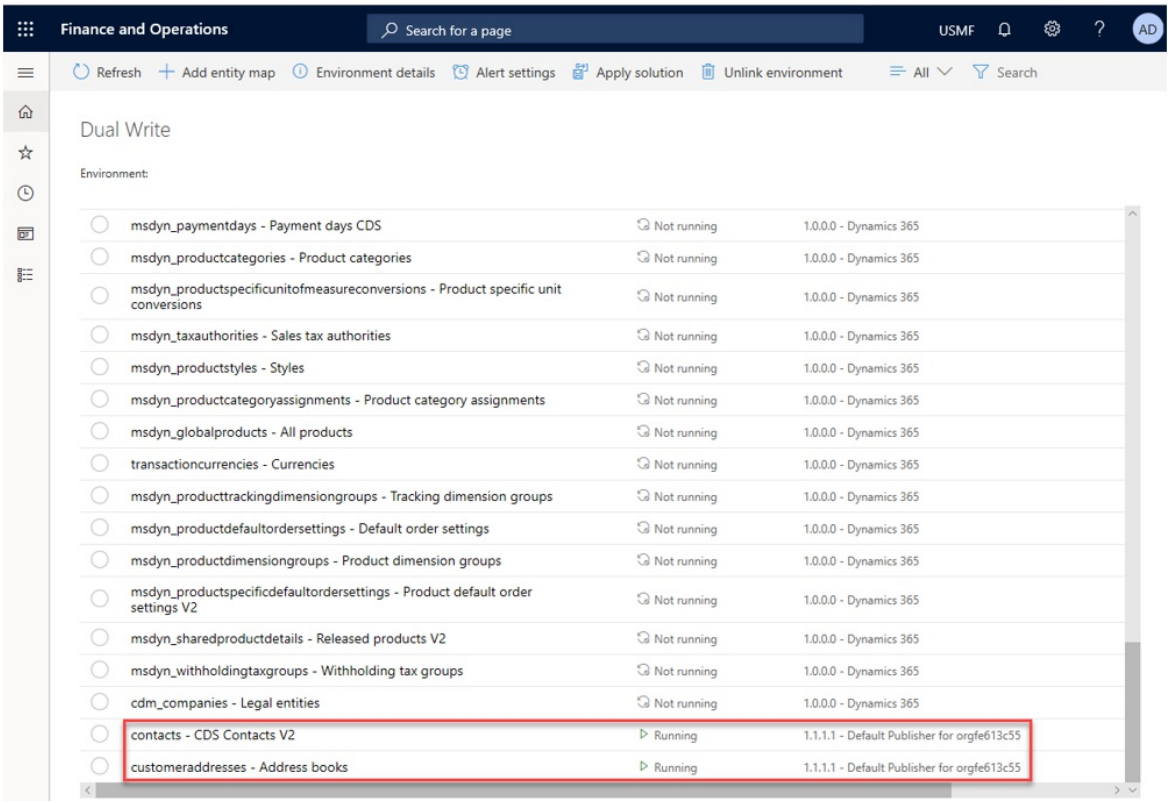
NOTE

When you create a new solution by using these modified table maps, you must specify the same publisher.

The following figure shows how to add a new table map that is named **Address books**.



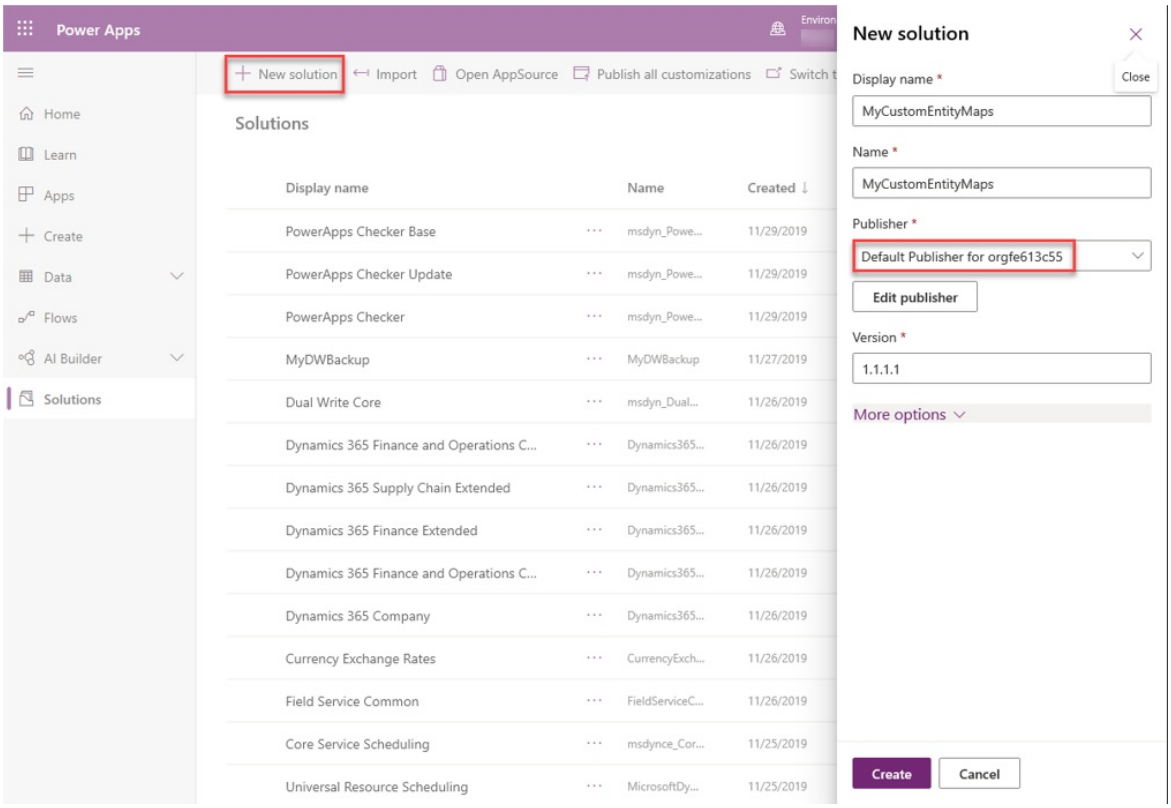
2. Confirm the table maps that you just modified and added. Be sure to enable and test them, to ensure that they work as you expect.



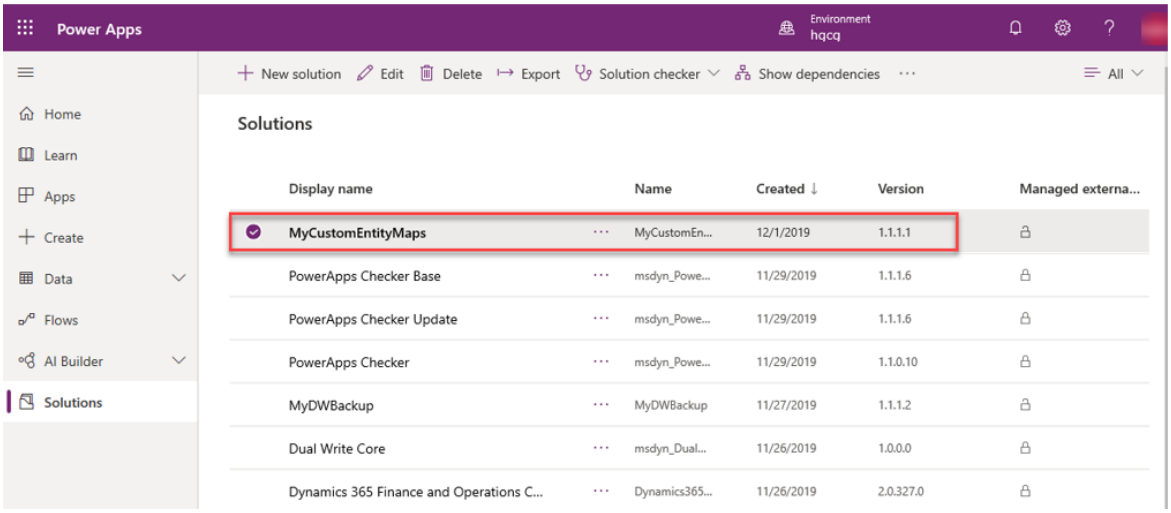
Create a new dual-write solution and add your components (Customized table maps)

Now that you've customized your mappings and added new mappings, the next step is to create a new dual-write solution and add the table maps to it.

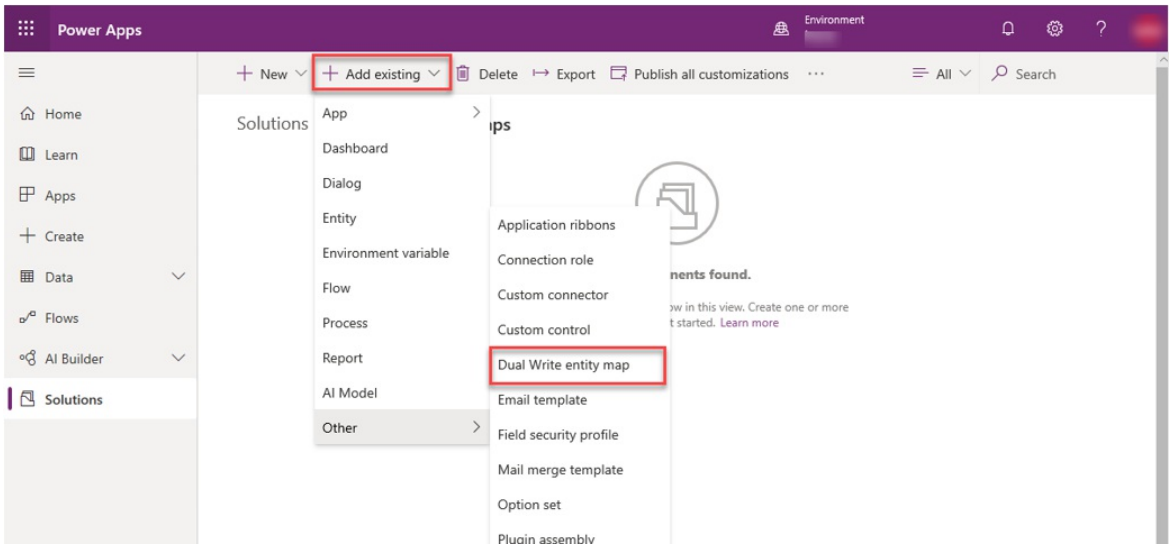
1. In Power Apps, in the left pane, select **Solutions**, and then select **New solution** to create a solution. For this example, the solution is named **MyCustomTableMaps**. Be sure to select the same publisher that you selected in previous steps.



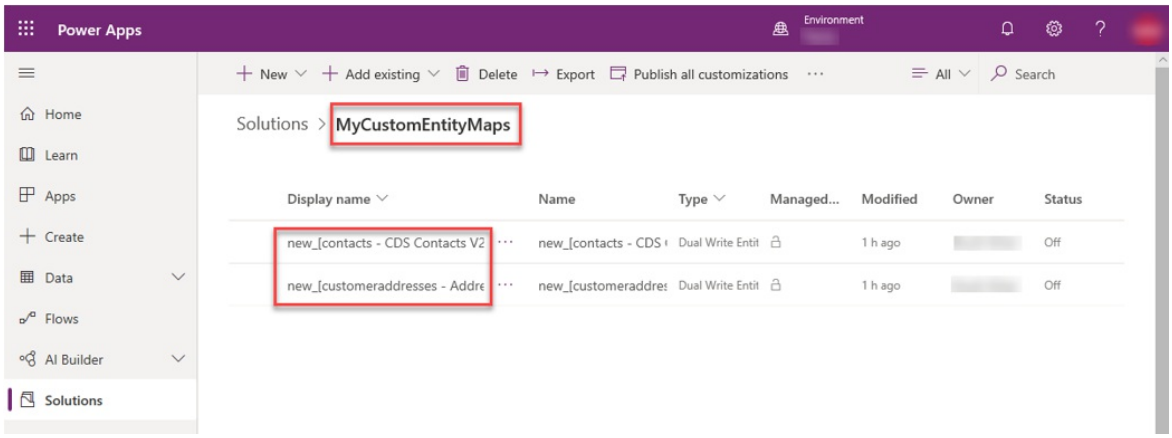
2. Select **Create**. The new solution appears on the **Solutions** list page.



3. Now that you've created your dual-write solution, you can add the customized table maps that you created in previous steps. Select the **MyCustomTableMaps** solution that you just created, select **Add existing**, point to **Other**, and then select **Dual Write table map**.



4. In the list, select the customized table maps, and add them to the solution. The solution should now contain your customized tables.



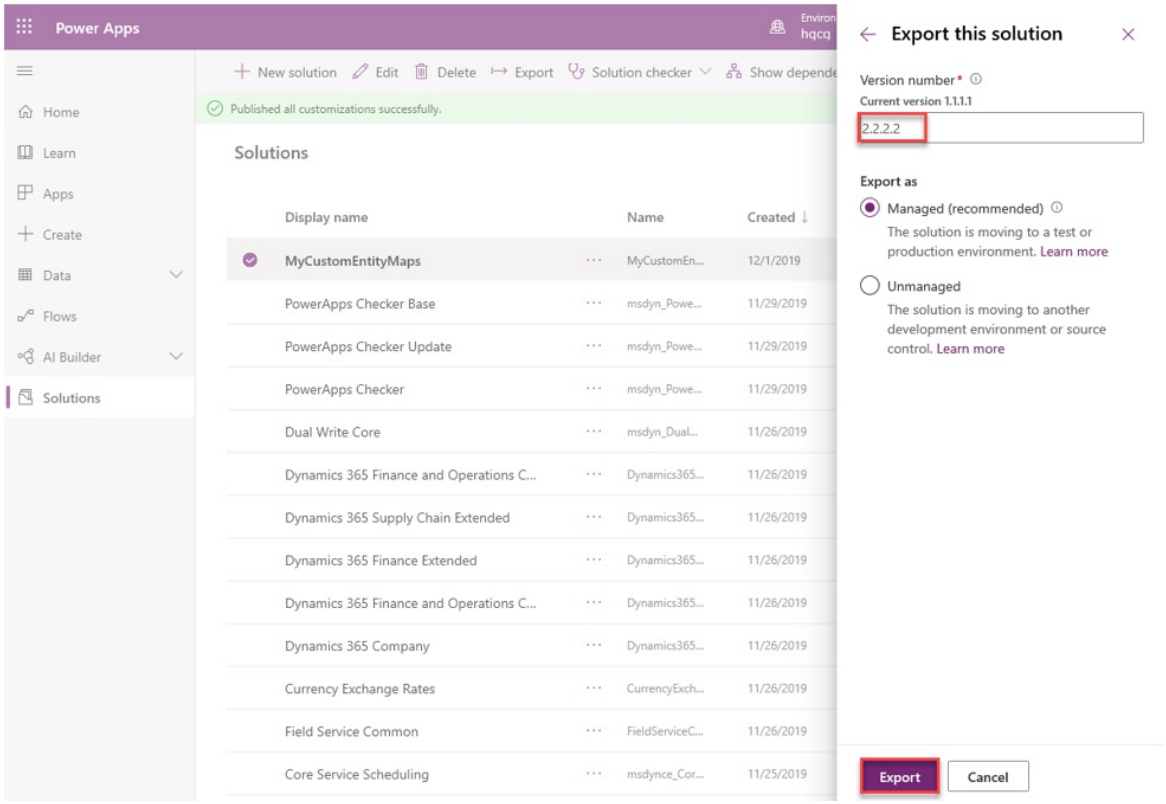
You've now customized your tables and put them into a solution.

Export and publish your solution

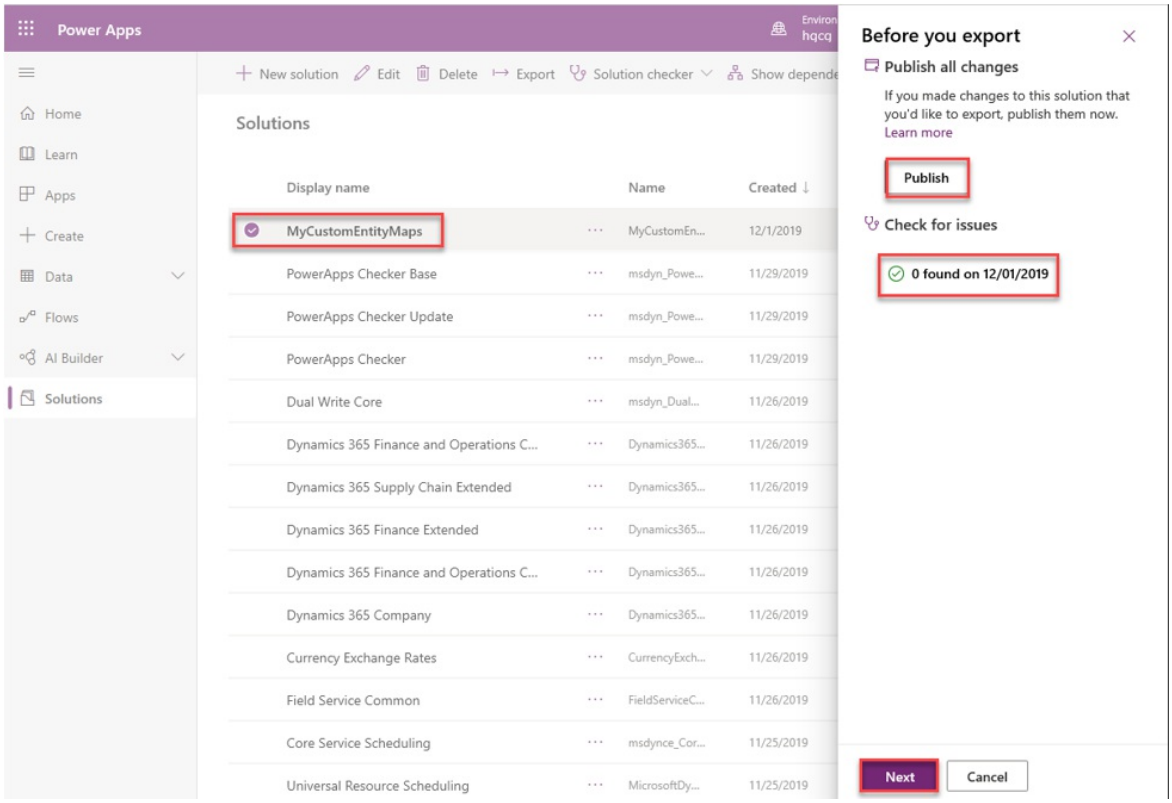
After you run the solution checker and make sure that there are no issues, you export the solution that you created and publish the changes.

1. In the list of solutions, select your solution, and then select **Export**.

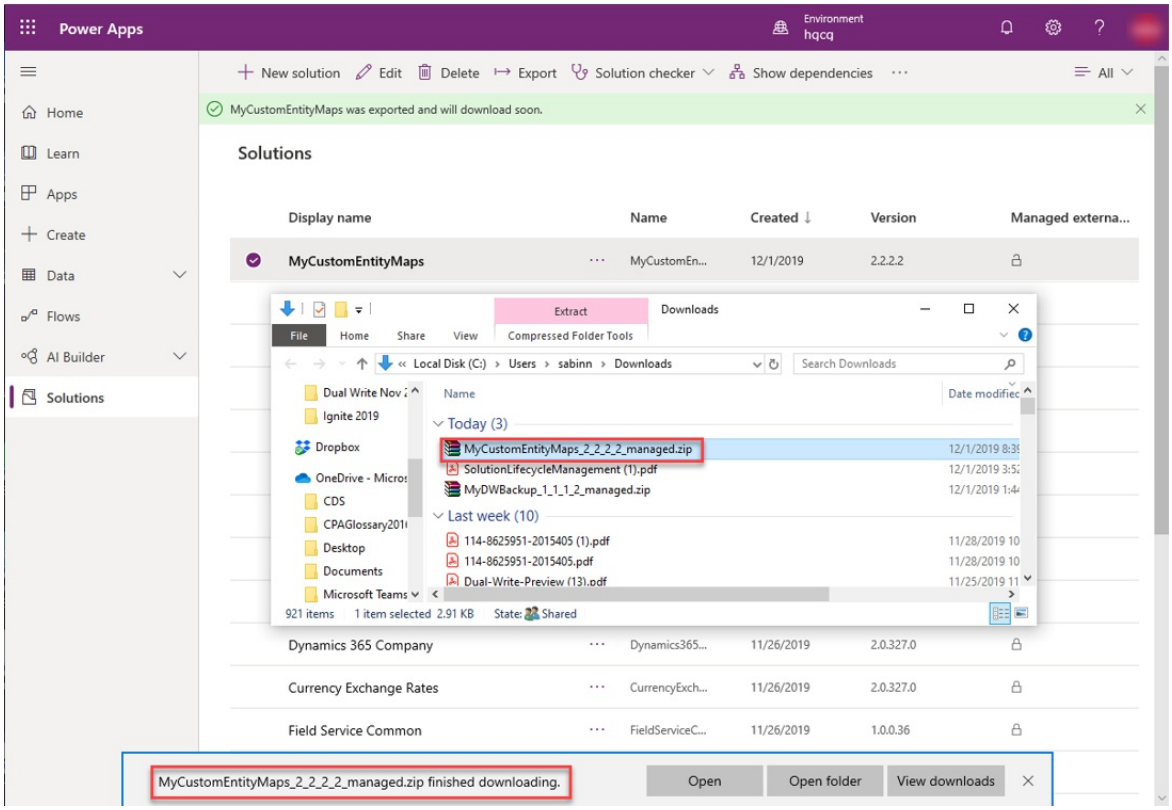
2. Update the version number, and select whether you want to export the solution as a managed or unmanaged solution. (We recommend that you export it as a managed solution.) Then select **Export**.



3. Before you export, select **Publish all changes**, and then select **Check for issues**. When you've finished, select **Next** to publish all your changes.



The solution, together with all its components, is exported to a zip file.



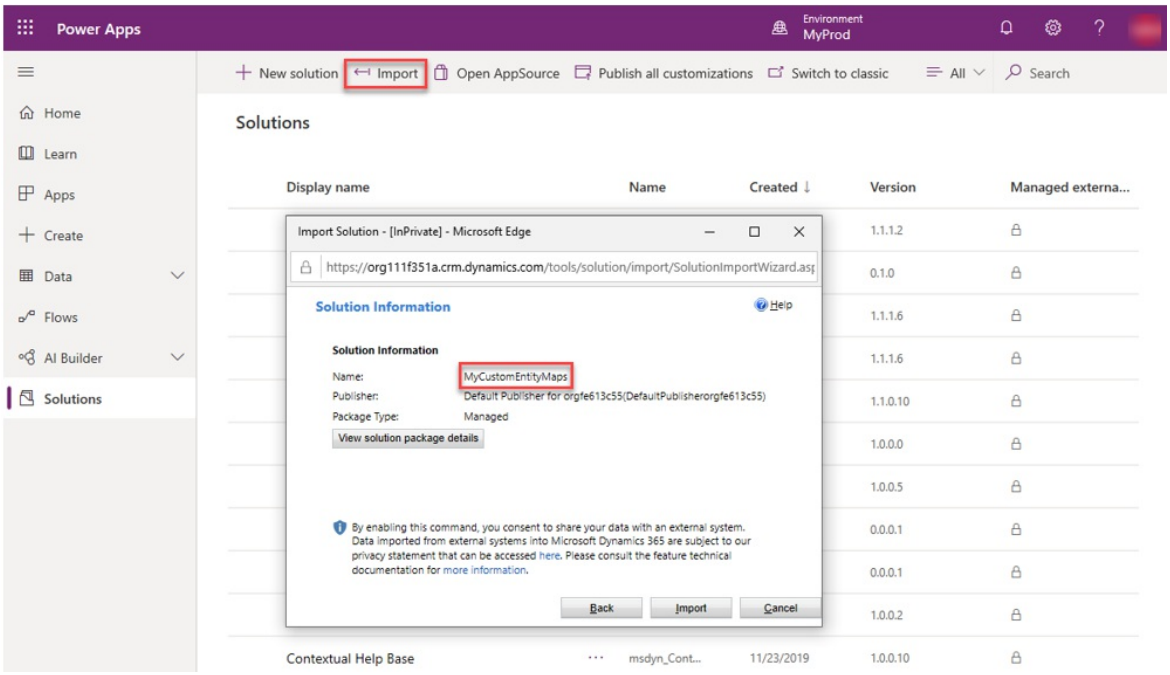
You've now customized your tables, added them to a new solution, and created a solution file that can be imported and applied to other environments. (This capability can be useful if you want to move table maps between test and production environments.) In a similar way, you can create a backup of all your table maps by adding them to a solution and exporting the solution as a package. That package can then be imported into to any environment to restore the table maps.

For information about how to publish the package to AppSource, see Publish your app on AppSource.

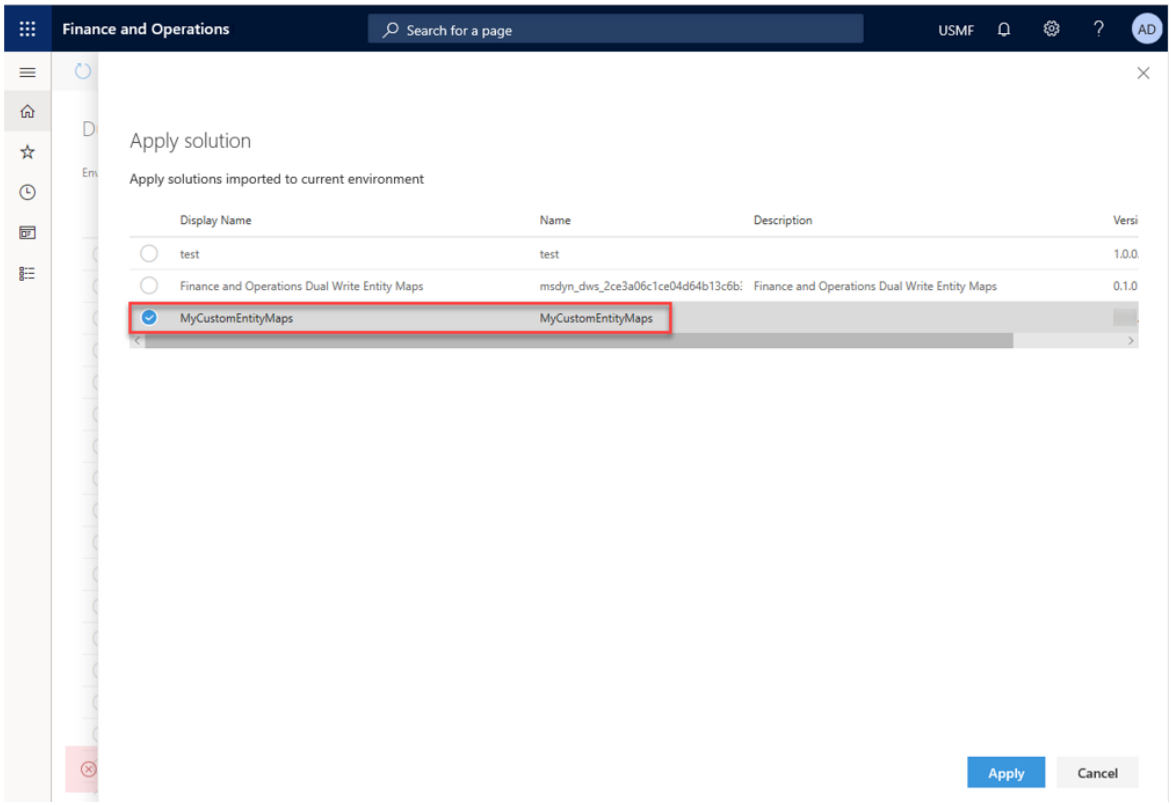
Test your exported solution package

You can test your exported solution package by importing and applying it to another environment.

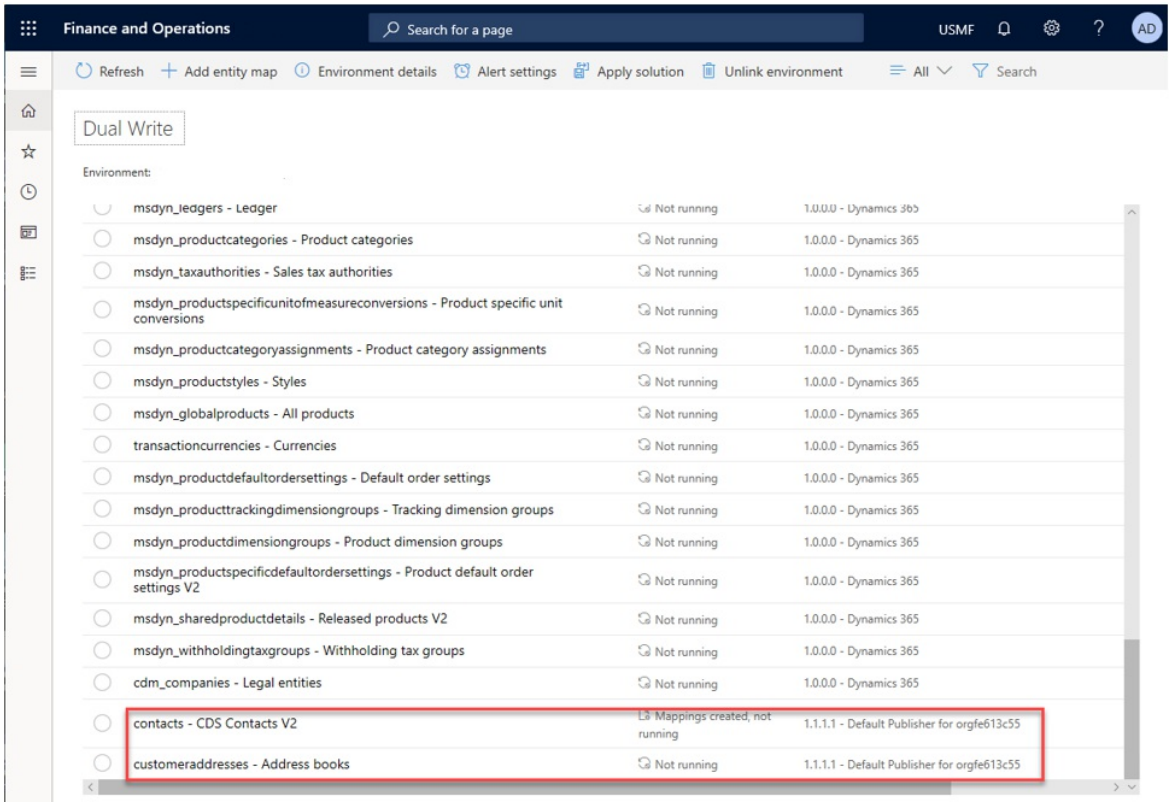
1. In Power Apps, select **Import** to import the package into a new environment.



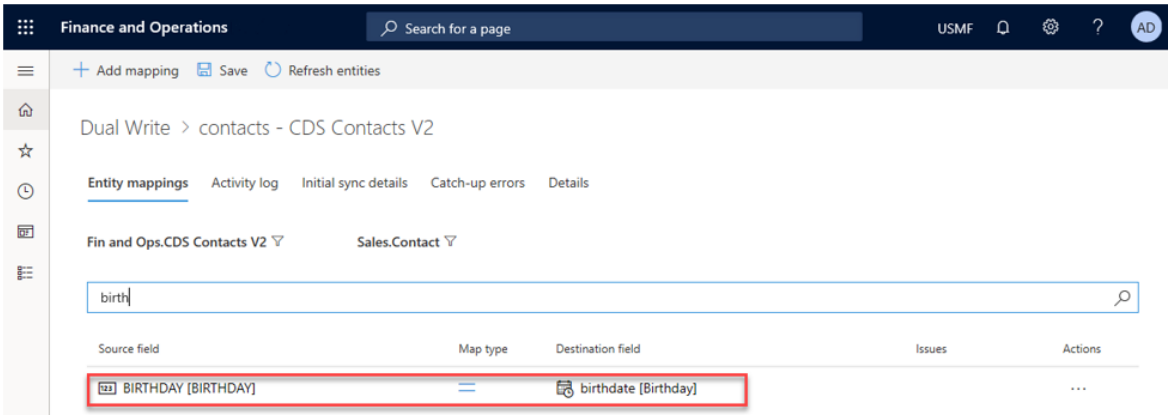
2. Apply the solution that you just imported to the environment.



3. Verify that the two customized table maps appear on the dual-write table maps list page.

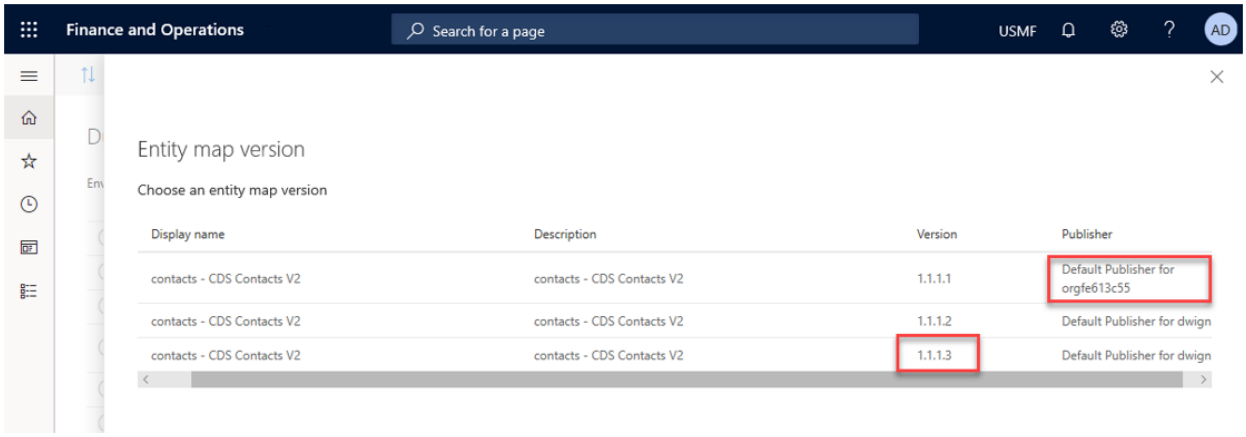


4. Make sure that the customizations from previous steps are preserved.



Use the table map version

Sometimes, a solution might contain different implementations of a table map. For example, the version of the contacts - CDS Contacts V2 table map might have a different publisher or a newer version number. In these cases, you can use the **Table Map version** button to select which table map you want to use in your environment.



Upgrade existing dual-write environments for solution awareness (Existing environments)

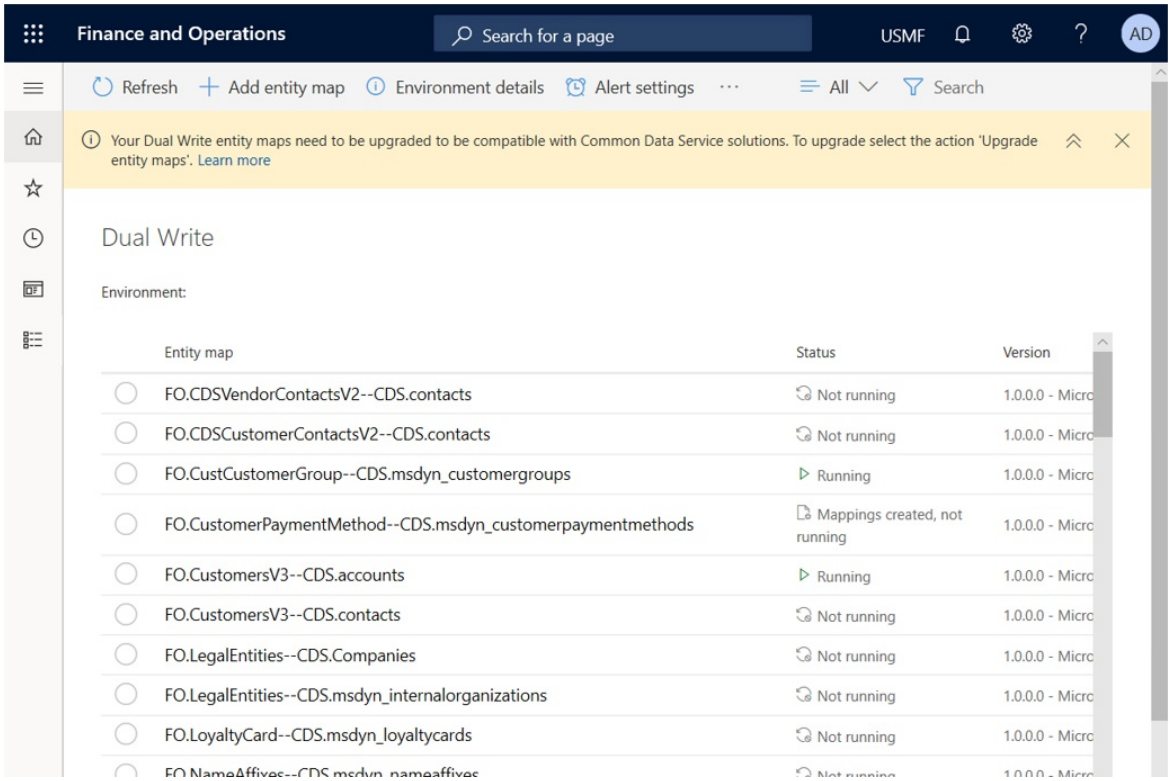
1. Import the dual-write core solution.

    a. Follow the instructions in the Install the dual-write core solution section earlier in this topic to import the dual-write core solution from AppSource into Power Apps.

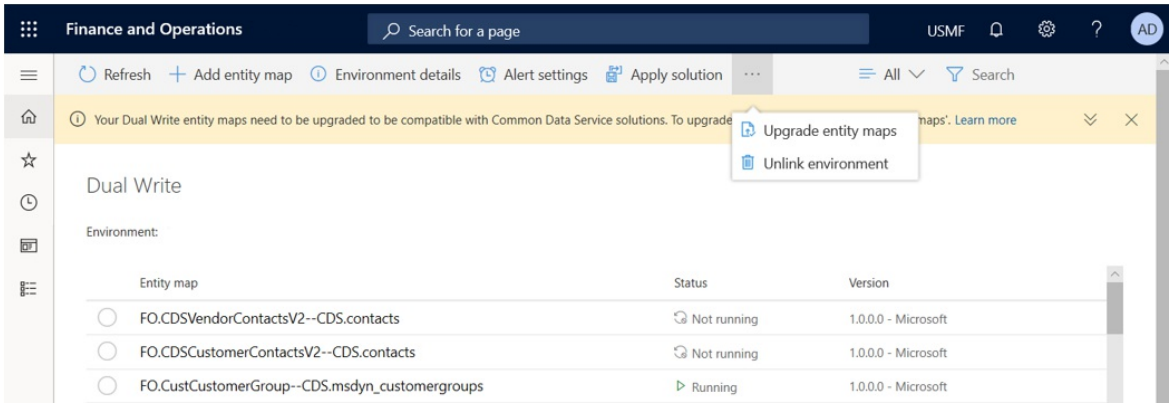
    b. Verify that the dual-write core solution is listed under **Solutions** in Power Apps.

2. Upgrade the table maps.

    You'll see a notification prompting you to upgrade.



Select **Upgrade table maps** at the top of the page.



The upgrade takes a few minutes. When it's completed, you receive a notification.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Integrated customer master

2/18/2021 • 8 minutes to read • Edit Online

NOTE

Effective November 2020:

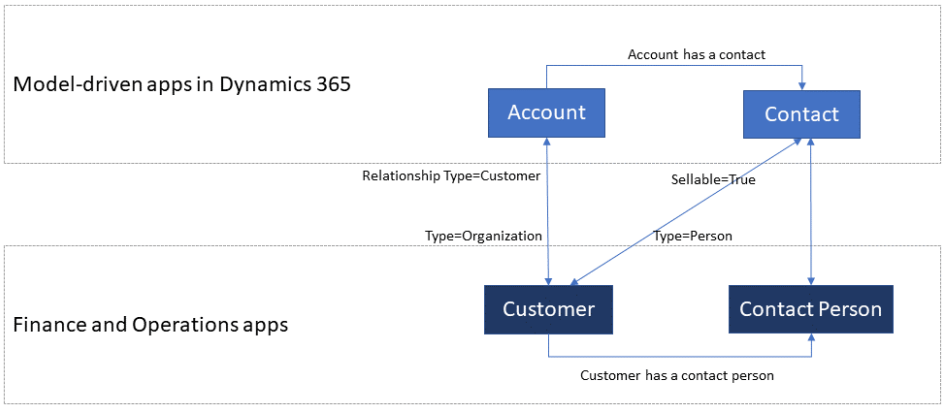
- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

Customer data can be mastered in more than one Dynamics 365 application. For example, a customer row can originate though sales activity in Dynamics 365 Sales (a model-driven app in Dynamics 365), or a row can originate through retail activity in Dynamics 365 Commerce (a Finance and Operations app). No matter where where the customer data originates, it is integrated behind the scenes. Integrated customer master gives you the flexibility to master customer data in any Dynamics 365 application and provides a comprehensive view of the customer across the Dynamics 365 application suite.

Customer data flow

Customer is a well-defined concept in applications. Therefore, the integration of customer data just involves harmonizing the customer concept between the two applications. The following illustration shows the customer data flow.



Customers can be broadly classified into two types: commercial/organizational customers and consumers/end users. These two types of customers are stored and handled differently in Finance and Operations and Dataverse.

In Finance and Operations, both commercial/organizational customers and consumers/end users are mastered in a single table that is named **CustTable** (CustCustomerV3Entity), and they are classified based on the **Type**

attribute. (If **Type** is set to **Organization**, the customer is a commercial/organizational customer, and if **Type** is set to **Person**, the customer is a consumer/end user.) The primary contact person information is handled through the SMMContactPersonEntity table.

In Dataverse, commercial/organizational customers are mastered in the Account table and are identified as customers when the **RelationshipType** attribute is set to **Customer**. Both consumers/end users and the contact person are represented by the Contact table. To provide a clear separation between a consumer/end user and a contact person, the **Contact** table has a Boolean flag that is named **Sellable**. When **Sellable** is **True**, the contact is a consumer/end user, and quotations and orders can be created for that contact. When **Sellable** is **False**, the contact is just a primary contact person of a customer.

When a non-sellable contact participates in a quotation or order process, **Sellable** is set to **True** to flag the contact as a sellable contact. A contact that has become a sellable contact remains a sellable contact.

Templates

Customer data includes all information about the customer, such as the customer group, addresses, contact information, payment profile, invoice profile, and loyalty status. A collection of table maps works together during customer data interaction, as shown in the following table.

FINANCE AND OPERATIONS APPS	OTHER DYNAMICS 365 APPS	DESCRIPTION
CDS Contacts V2	contacts	This template synchronizes all primary, secondary, and tertiary contact information, for both customers and vendors.
Customer groups	msdyn_customergroups	This template synchronizes customer group information.
Customer payment method	msdyn_customerpaymentmethods	This template synchronizes customer payment method information.
Customers V3	accounts	This template synchronizes customer master information for commercial and organizational customers.
Customers V3	contacts	This template synchronizes customer master data for consumers and end users.
Name affixes	msdyn_nameaffixes	This template synchronizes name affixes reference data, for both customers and vendors.
Payment day lines CDS V2	msdyn_paymentdaylines	This template synchronizes payment day lines reference data, for both customers and vendors.
Payment days CDS	msdyn_paymentdays	This template synchronizes payment days reference data, for both customers and vendors.
Payment schedule lines	msdyn_paymentschedulelines	Syncs payment schedule lines reference data, for both customers and vendors.

FINANCE AND OPERATIONS APPS	OTHER DYNAMICS 365 APPS	DESCRIPTION
Payment schedule	msdyn_paymentschedules	This template synchronizes payment schedule reference data, for both customers and vendors.
Terms of payment	msdyn_paymentterms	This template synchronizes payment terms (terms of payment) reference data, for both customers and vendors.

Mapping tables

Mapping types

There are several different mapping types. The following table explains the symbols used in the template tables.

SYMBOL	DESCRIPTION
>	One-way
>>	One-way, and data is transformed in the process.
=	Bidirectional
><	Bidirectional, and data is transformed in the process.
<<	One-way, and data is transformed in the process.

Filters

The source filter and reverse source filter determine which rows are synchronized.

Default values

If a synchronized field does not exist in either the Finance and Operations table or the other Dynamics 365 table, then a default value is assigned in the synchronized table. In some cases, the default value is an integer that is a lookup to an attribute value in the Common Data Model. For example, in the Contacts table of the Common Data Model, the default value of **address1_addresstypecode** is **3**. In the Common Data Model, for address1AddressTypeCode the value of **3** is **Primary address**.

CDS Contacts V2 to contacts

This template synchronizes data between Finance and Operations apps and Dataverse.

Source filter: `(AssociatedContactType = 1)`

Reversed source filter: msdyn_contactforvendor eq true and msdyn_sellable eq false and msdyn_contactpersonid ne ''

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
CONTACTPERSONPARTYNUMBER	=	msdyn_partynumber	
ASSOCIATEDCONTACTTYPE	<<	none	Vendor

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
FIRSTNAME	=	firstname	
MIDDLENAME	=	middlename	
LASTNAME	=	lastname	
ASSOCIATEDCONTACTNUMBER	=	msdyn_vendorcontactid.msdyn_vendoraccountnumber	
PRIMARYADDRESSCITY	=	address1_city	
PRIMARYADDRESSCOUNTRYREGIONID	=	address1_country	
PRIMARYADDRESSCOUNTYID	=	address1_county	
PRIMARYFAXNUMBER	=	fax	
PRIMARYADDRESSSTATEID	=	address1_stateorprovince	
PRIMARYADDRESSSTREET	=	address1_line1	
PRIMARYADDRESSZIPCODE	=	address1_postalcode	
PRIMARYPHONENUMBER	=	telephone1	
PRIMARYEMAILADDRESS	=	emailaddress1	
EMPLOYMENTDEPARTMENT	=	department	
NOTES	=	description	
GENDER	><	gendercode	
GOVERNMENTIDENTIFICATIONNUMBER	=	governmentid	
PRIMARYURL	=	websiteurl	
MARITALSTATUS	><	familystatuscode	
ISRECEIVINGDIRECTMAIL	><	donotemail	
EMPLOYMENTPROFESSION	=	jobtitle	
SPOUSENAME	=	spousesname	
none	>>	msdyn_contactforvendor	True

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
CONTACTPERSONID	=	msdyn_contactpersonid	

Customer groups to msdyn_customergroups

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
CUSTOMERGROUPID	=	msdyn_groupid	
DESCRIPTION	=	msdyn_description	
ISSALESTAXINCLUDEDINPRICE	><	msdyn_issalestaxincludedinprice	
PAYMENTTERMID	=	msdyn_paymenttermid.msdyn_name	
CLEARINGPERIODPAYMENTTERMNAME	=	msdyn_clearingperiodpaymenttermname.msdyn_name	

Customer payment method to msdyn_customerpaymentmethods

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
NAME	=	msdyn_name	
ACCOUNTTYPE	><	msdyn_accounttype	
DISCOUNTGRACEPERIODDAYS	=	msdyn_discountgraceperioddays	
BRIDGINGPOSTINGENABLED	><	msdyn_bridgingpostingenabled	
ISSEPA	><	msdyn_issepa	
LASTFILENUMBER	=	msdyn_lastfilenumber	
LASTFILENUMBERTODAY	=	msdyn_lastfilenumbertoday	
DESCRIPTION	=	msdyn_description	
PAYMENTTYPE	><	msdyn_paymenttype	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
CREATEANDDRAWBILLOFEXCHANGEDURINGINVOICEPOSTING	><	msdyn_invoiceupdate	
PAYMENTSTATUS	><	msdyn_paymentstatus	
SUMBYPERIOD	><	msdyn_sumbyperiod	
ENABLEPOSTDATEDCHECKCLEARINGPOSTING	><	msdyn_enablepostdatescheckclearingposting	
BILLOFEXCHANGEDRAFTTYPE	><	msdyn_billofexchangedrafttype	
DIRECTDEBIT	><	msdyn_directdebit	

Customers V3 to accounts

This template synchronizes data between Finance and Operations apps and Dataverse.

Source filter: `((PartyType == "Organization"))`

Reversed source filter: customertypecode eq 3

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
CUSTOMERACCOUNT	=	accountnumber	
INVOICEADDRESSCITY	=	address2_city	
INVOICEADDRESSCOUNTRYREGIONISOCODE	=	address2_country	
INVOICEADDRESSCOUNTY	=	address2_county	
INVOICEADDRESSLATITUDE	>	address2_latitude	
INVOICEADDRESSLONGITUDE	>	address2_longitude	
INVOICEADDRESSSTATE	=	address2_stateorprovince	
INVOICEADDRESSSTREET	=	address2_line1	
INVOICEADDRESSZIPCODE	=	address2_postalcode	
CREDITLIMIT	=	creditlimit	
DELIVERYADDRESSCITY	=	address1_city	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
DELIVERYADDRESSCOUNTRYREGIONISOCODE	=	address1_country	
DELIVERYADDRESSCOUNTY	=	address1_county	
DELIVERYADDRESSLATITUDE	>	address1_latitude	
DELIVERYADDRESSLONGITUDE	>	address1_longitude	
DELIVERYADDRESSZIPCODE	=	address1_postalcode	
ORGANIZATIONNAME	=	name	
ORGANIZATIONNUMBEROFEMPLOYEES	=	numberofemployees	
PRIMARYCONTACTEMAIL	=	emailaddress1	
PRIMARYCONTACTFAX	=	fax	
PRIMARYCONTACTPHONE	=	telephone1	
PRIMARYCONTACTTWITTER	=	primarytwitterid	
PRIMARYCONTACTURL	=	websiteurl	
SALESCURRENCYCODE	=	transactioncurrencyid.isocurrencycode	
SALESMEMO	=	description	
CREDITLIMITISMANDATORY	><	msdyn_creditlimitismandatory	
CREDITRATING	=	msdyn_creditrating	
CUSTOMERGROUPID	=	msdyn_customergroupid.msdyn_groupid	
IDENTIFICATIONNUMBER	=	msdyn_identificationnumber	
INVOICEACCOUNT	=	msdyn_billingaccount.accountnumber	
INVOICEADDRESS	><	msdyn_invoiceaddress	
ISONETIMECUSTOMER	><	msdyn_onetimecustomer	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
ONHOLDSTATUS	><	msdyn_onholdstatus	
PARTYCOUNTRY	=	msdyn_partycountry	
PARTYSTATE	=	msdyn_partystateprovince	
PAYMENTDAY	=	msdyn_paymentday.msdyn_name	
PAYMENTMETHOD	=	msdyn_customerpaymentmethod.msdyn_name	
PAYMENTSCHEDULE	=	msdyn_paymentschedule.msdyn_name	
PAYMENTTERMS	=	msdyn_paymentterm.msdyn_name	
PAYMENTTERMSBASEDAYS	=	msdyn_paymenttermsbasedays	
PRIMARYCONTACTFACEBOOK	=	msdyn_primaryfacebookid	
PRIMARYCONTACTFAXEXTENSION	=	msdyn_faxextension	
PRIMARYCONTACTLINKEDIN	=	msdyn_primarylinkedinid	
TAXEXEMPTNUMBER	=	msdyn_taxexemptnumber	
VENDORACCOUNT	=	msdyn_vendor.msdyn_vendoraccountnumber	
PRIMARYCONTACTEMAILDESCRIPTION	=	msdyn_emailaddress1description	
PRIMARYCONTACTFACEBOOKDESCRIPTION	=	msdyn_primaryfacebookdescription	
PRIMARYCONTACTFAXDESCRIPTION	=	msdyn_faxdescription	
PRIMARYCONTACTLINKEDINDESCRIPTION	=	msdyn_primarylinkedindescription	
PRIMARYCONTACTPHONEDESCRIPTION	=	msdyn_telephone1description	
PRIMARYCONTACTPHONEEXTENSION	=	msdyn_telephone1extension	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
PRIMARYCONTACTTWITTERDESCRIPTION	=	msdyn_primarytwitteriddescription	
PRIMARYCONTACTURLDESCRIPTION	=	msdyn_websiteurldescription	
LANGUAGEID	<<	none	en-us
DELIVERYADDRESSSTREET	=	address1_line1	
DELIVERYADDRESSSTATE	=	address1_stateorprovince	
none	>>	address1_addresstypecode	2
none	>>	customertypecode	3
PARTYTYPE	<<	none	Organization
PARTYNUMBER	=	msdyn_partynumber	
CONTACTPERSONID	=	primarycontactid.msdyn_contactpersonid	

Customers V3 to contacts

This template synchronizes data between Finance and Operations apps and Dataverse.

Source filter: `((PartyType == "Person"))`

Reversed source filter: msdyn_sellable eq true and msdyn_contactpersonid ne ''

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
none	>>	msdyn_sellable	True
PARTYTYPE	<<	none	Person
PARTYNUMBER	=	msdyn_partynumber	
CUSTOMERACCOUNT	=	msdyn_contactpersonid	
CUSTOMERGROUPID	=	msdyn_customergroupid.msdyn_groupid	
PERSONFIRSTNAME	=	firstname	
PERSONLASTNAME	=	lastname	
PERSONMIDDLENAME	=	middlename	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
PERSONPROFESSIONALTITLE	=	jobtitle	
PERSONGENDER	><	gendercode	
PERSONMARITALSTATUS	><	familystatuscode	
LANGUAGEID	<<	none	en-us
ADDRESSCITY	=	address1_city	
ADDRESSCOUNTRYREGIONISOCODE	=	address1_country	
ADDRESSCOUNTY	=	address1_county	
ADDRESSLATITUDE	>	address1_latitude	
ADDRESSLONGITUDE	>	address1_longitude	
ADDRESSLOCATIONROLES	<<	none	Business
ADDRESSSTATE	=	address1_stateorprovince	
ADDRESSSTREET	=	address1_line1	
ADDRESSZIPCODE	=	address1_postalcode	
ADDRESSPOSTBOX	=	address1_postofficebox	
none	>>	address1_addresstypecode	3
INVOICEADDRESSCITY	=	address2_city	
INVOICEADDRESSCOUNTRYREGIONISOCODE	=	address2_country	
INVOICEADDRESSCOUNTY	=	address2_county	
INVOICEADDRESSLATITUDE	>	address2_latitude	
INVOICEADDRESSLONGITUDE	>	address2_longitude	
INVOICEADDRESSSTATE	=	address2_stateorprovince	
INVOICEADDRESSSTREET	=	address2_line1	
INVOICEADDRESSZIPCODE	=	address2_postalcode	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
DELIVERYADDRESSCITY	=	address3_city	
DELIVERYADDRESSCOUNTRYREGIONISOCODE	=	address3_country	
DELIVERYADDRESSCOUNTY	=	address3_county	
DELIVERYADDRESSLATITUDE	>	address3_latitude	
DELIVERYADDRESSLONGITUDE	>>	address3_longitude	
DELIVERYADDRESSSTATE	=	address3_stateorprovince	
DELIVERYADDRESSSTREET	=	address3_line1	
DELIVERYADDRESSZIPCODE	=	address3_postalcode	
PRIMARYCONTACTEMAIL	=	emailaddress1	
PRIMARYCONTACTEMAILDESCRIPTION	=	msdyn_emailaddress1description	
PRIMARYCONTACTFAX	=	fax	
PRIMARYCONTACTFAXDESCRIPTION	=	msdyn_faxdescription	
PRIMARYCONTACTFAXEXTENSION	=	msdyn_faxextension	
IDENTIFICATIONNUMBER	=	msdyn_identificationnumber	
PARTYCOUNTRY	=	msdyn_partycountry	
PARTYSTATE	=	msdyn_partystateprovince	
PRIMARYCONTACTFACEBOOK	=	msdyn_primaryfacebookid	
PRIMARYCONTACTFACEBOOKDESCRIPTION	=	msdyn_primaryfacebookdescription	
PRIMARYCONTACTLINKEDIN	=	msdyn_primaryinkedinid	
PRIMARYCONTACTLINKEDINDESCRIPTION	=	msdyn_primarylinkedindescription	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
PRIMARYCONTACTPHONE	=	telephone1	
PRIMARYCONTACTPHONEDESCRIPTION	=	msdyn_telephone1description	
PRIMARYCONTACTPHONEEXTENSION	=	msdyn_telephone1extension	
PRIMARYCONTACTTWITTER	=	msdyn_primarytwitterid	
PRIMARYCONTACTTWITTERDESCRIPTION	=	msdyn_primarytwitteriddescription	
PRIMARYCONTACTURL	=	websiteurl	
PRIMARYCONTACTURLDESCRIPTION	=	msdyn_websiteurldescription	
SALESCURRENCYCODE	=	transactioncurrencyid.isocurrencycode	
SALESMEMO	=	description	

Name affixes to msdyn_nameaffixes

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
AFFIX	=	msdyn_affix	
TYPE	><	msdyn_affixtype	
DESCRIPTION	=	msdyn_description	

Payment day lines CDS V2 to msdyn_paymentdaylines

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
NAME	=	msdyn_paymentday.msdyn_name	
LINENUMBER	=	msdyn_linenumber	
FREQUENCY	><	msdyn_frequency	
DAYOFWEEK	><	msdyn_dayofweek	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
DAYOFMONTH	=	msdyn_dayofmonth	

Payment days CDS to msdyn_paymentdays

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
NAME	=	msdyn_name	
DESCRIPTION	=	msdyn_description	

Payment schedule lines to msdyn_paymentschedulelines

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
PAYMENTSCHEDULENAME	=	msdyn_paymentschedule.msdyn_name	
LINENUMBER	=	msdyn_linenumber	
PERIODSAFTERDUEDATE	=	msdyn_periodsafterduedate	
PERCENTORAMOUNT	><	msdyn_percentoramount	
PERCENTORAMOUNTVALUE	=	msdyn_percentoramountvalue	

Payment schedule to msdyn_paymentschedules

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
NAME	=	msdyn_name	
DESCRIPTION	=	msdyn_description	
ALLOCATIONMETHOD	><	msdyn_allocationmethod	
PAYMENTFREQUENCYUNITS	><	msdyn_paymentfrequencyunit	
PAYMENTFREQUENCY	=	msdyn_paymentfrequency	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
NUMBEROFPAYMENTS	=	msdyn_numberofpayments	
FIXEDPAYMENTAMOUNT	=	msdyn_fixedpaymentamount	
MINIMUMPAYMENTAMOUNT	=	msdyn_minimumpaymentamount	
SALESTAXALLOCATIONMETHOD	><	msdyn_salestaxallocationmethod	
NOTES	=	msdyn_note	

Terms of payment to msdyn_paymentterms

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
DESCRIPTION	=	msdyn_description	
NAME	=	msdyn_name	
NUMBEROFMONTHS	=	msdyn_numberofmonth	
CUTOFFDAYOFMONTH	=	msdyn_cutoffdayofmonth	
ISCASHPAYMENT	><	msdyn_iscashpayment	
NUMBEROFDAYS	=	msdyn_days	
ISCERTIFIEDCOMPANYCHECK	><	msdyn_iscertifiedcompanycheck	
ISDEFAULTPAYMENTTERM	><	msdyn_isdefaultpaymentterm	
CREDITCARDPAYMENTTYPE	><	msdyn_creditcardpaymenttype	
CREDITCARDCREDITCHECKTYPE	><	msdyn_creditcardcreditchecktype	
PAYMENTDAYNAME	=	msdyn_paymentdayname.msdyn_name	
PAYMENTMETHODTYPE	><	msdyn_paymentmethodtype	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
PAYMENTSCHEDULENAME	=	msdyn_paymentschedulename.msdyn_name	

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Integrated vendor master

2/18/2021 • 5 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

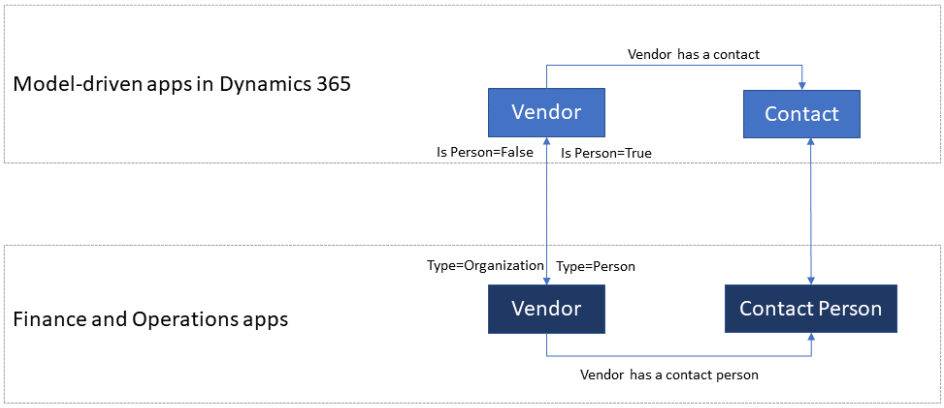
This topic will be updated soon to reflect the latest terminology.

The term *vendor* refers to a supplier organization, or a sole proprietor who supplies goods or services to a business. Although *vendor* is an established concept in Microsoft Dynamics 365 Supply Chain Management, no vendor concept exists in model-driven apps in Dynamics 365. However, you can overload the **Account/Contact** table to store vendor information. The integrated vendor master introduces an explicit vendor concept in model-driven apps in Dynamics 365. You can either use the new vendor design or store vendor data in the **Account/Contact** table. Dual-write supports both approaches.

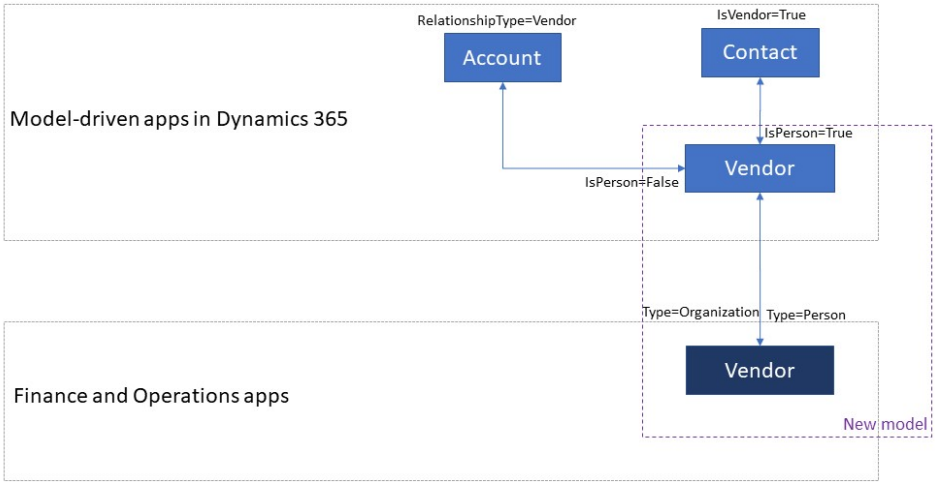
In both approaches, the vendor data is integrated between Dynamics 365 Supply Chain Management, Dynamics 365 Sales, Dynamics 365 Field Service, and Power Apps portals. In Supply Chain Management, the data is available for workflows such as purchase requisitions and purchase orders.

Vendor data flow

If you don't want to store vendor data in the **Account/Contact** table in Dataverse, you can use the new vendor design.



If you want to continue to store vendor data in the **Account/Contact** table, you can use the extended vendor design. To use the extended vendor design, you must configure the vendor workflows in the dual-write solution package. For more information, see Switch between vendor designs.



TIP

If you're using Power Apps portals for self-service vendors, the vendor information can flow directly to Finance and Operations apps.

Templates

Vendor data includes all information about the vendor, such as the vendor group, addresses, contact information, payment profile, and invoice profile. A collection of table maps work together during vendor data interaction, as shown in the following table.

FINANCE AND OPERATIONS APPS	OTHER DYNAMICS 365 APPS	DESCRIPTION
Vendor V2	Account	Businesses that use the Account table to store vendor information can continue to use it in the same way. They can also take advantage of the explicit vendor functionality that is coming because of Finance and Operations apps integration.
Vendor V2	Msdyn_vendors	Businesses that use a custom solution for vendors can take advantage of the out-of-box vendor concept that is being introduced in Dataverse because of Finance and Operations apps integration.
Vendor groups	msdyn_vendorgroups	This template synchronizes vendor group information.
Vendor payment method	msdyn_vendorpaymentmethods	This template synchronizes vendor payment method information.

FINANCE AND OPERATIONS APPS	OTHER DYNAMICS 365 APPS	DESCRIPTION
CDS Contacts V2	contacts	The contacts template synchronizes all primary, secondary, and tertiary contact information, for both customers and vendors.
Payment schedule lines	msdyn_paymentschedulelines	The payment schedule lines template synchronizes reference data for customers and vendors.
Payment schedule	msdyn_paymentschedules	The payment schedules template synchronizes payment schedule reference data, for both customers and vendors.
Payment day lines CDS V2	msdyn_paymentdaylines	The payment day lines template synchronizes payment day lines reference data for customers and vendors.
Payment days CDS	msdyn_paymentdays	The payment days template synchronizes payment days reference data, for both customers and vendors.
Terms of payment	msdyn_paymentterms	The terms of payment template synchronizes payment terms reference data, for both customers and vendors.
Name affixes	msdyn_nameaffixes	The name affixes template synchronizes name affixes reference data, for both customers and vendors.

Mapping tables

Mapping types

There are several different mapping types. The following table explains the symbols used in the template tables.

SYMBOL	DESCRIPTION
>	One-way
>>	One-way, and data is transformed in the process.
=	Bidirectional
><	Bidirectional, and data is transformed in the process.
<<	One-way, and data is transformed in the process.

Filters

The source filter and reverse source filter determine which rows are synchronized.

Default values

If a synchronized field does not exist in either the Finance and Operations table or the other Dynamics 365 table,

then a default value is assigned in the synchronized table. In some cases, the default value is an integer that is a lookup to an attribute value in the Common Data Model. For example, in the Contacts table of the Common Data Model, the default value of **address1_addresstypecode** is **3**. In the Common Data Model, for address1AddressTypeCode the value of **3** is **Primary address**.

Vendors V2 to msdyn_vendors

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
VENDORACCOUNTNUMBER	=	msdyn_vendoraccountnumber	
VENDORGROUPID	=	msdyn_vendorgroupid.msdyn_vendorgroup	
VENDORORGANIZATIONNAME	=	msdyn_name	
VENDORPARTYTYPE	><	msdyn_isperson	
PERSONFIRSTNAME	=	msdyn_firstname	
PERSONLASTNAME	=	msdyn_lastname	
CREDITLIMIT	=	msdyn_vendorcreditlimit	
ISFOREIGNENTITY	><	msdyn_isforeignentity	
ISONETIMEVENDOR	><	msdyn_isonetimevendor	
ADDRESSBUILDINGCOMPLIMENT	=	msdyn_addressbuildingcompliment	
PERSONCHILDRENNAMES	=	msdyn_childrennames	
ADDRESSCITY	=	msdyn_addresscity	
ADDRESSCOUNTRYREGIONID	=	msdyn_addresscountryregionid	
ADDRESSCOUNTRYREGIONISOCODE	=	msdyn_addresscountryregionisocode	
ADDRESSCOUNTYID	=	msdyn_addresscountyid	
CREDITRATING	=	msdyn_creditrating	
ADDRESSDESCRIPTION	=	msdyn_addressdescription	
ADDRESSDISTRICTNAME	=	msdyn_addressdistrictname	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
DUNSNUMBER	=	msdyn_dunsnumber	
ETHNICORIGINID	=	msdyn_ethnicorigin	
FORMATTEDPRIMARYADDRESS	=	msdyn_formattedprimaryaddress	
PERSONHOBBIES	=	msdyn_hobbies	
PERSONINITIALS	=	msdyn_initials	
LANGUAGEID	><	msdyn_language	
PERSONLASTNAMEPREFIX	=	msdyn_lastnameprefix	
PERSONMIDDLENAME	=	msdyn_middlename	
ORGANIZATIONNUMBER	=	msdyn_organizationnumber	
OURACCOUNTNUMBER	=	msdyn_ourvendoraccountnumber	
PAYMENTID	=	msdyn_paymentid	
PERSONPHONETICFIRSTNAME	=	msdyn_phoneticfirstname	
PERSONPHONETICMIDDLENAME	=	msdyn_phoneticmiddlename	
PERSONPHONETICLASTNAME	=	msdyn_phoneticlastname	
ORGANIZATIONPHONETICNAME	=	msdyn_organizationphoneticname	
ADDRESSPOSTBOX	=	msdyn_addresspostbox	
PRIMARYURL	=	msdyn_primarycontacturl	
PRIMARYEMAILADDRESS	=	msdyn_primaryemailaddress	
PRIMARYEMAILADDRESSDESCRIPTION	=	msdyn_primaryemailaddressdescription	
PRIMARYFACEBOOK	=	msdyn_primaryfacebook	
PRIMARYFACEBOOKDESCRIPTION	=	msdyn_primaryfacebookdescription	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
PRIMARYFAXNUMBER	=	msdyn_primaryfaxnumber	
PRIMARYFAXNUMBERDESCRIPTION	=	msdyn_primaryfaxnumberdescription	
PRIMARYFAXNUMBEREXTENSION	=	msdyn_primaryfaxnumberextension	
PRIMARYLINKEDIN	=	msdyn_primarylinkedin	
PRIMARYLINKEDINDESCRIPTION	=	msdyn_primarylinkedindescription	
PRIMARYPHONENUMBER	=	msdyn_pimaryphonenumber	
PRIMARYPHONENUMBERDESCRIPTION	=	msdyn_primaryphonenumberdescription	
PRIMARYPHONENUMBEREXTENSION	=	msdyn_primaryphonenumberextension	
PRIMARYTELEX	=	msdyn_primarytelex	
PRIMARYTELEXDESCRIPTION	=	msdyn_primarytelexdescription	
PRIMARYTWITTER	=	msdyn_primarytwitter	
PRIMARYTWITTERDESCRIPTION	=	msdyn_primarytwitterdescription	
PRIMARYURLDESCRIPTION	=	msdyn_primaryurldescription	
PERSONPROFESSIONALSUFFIX	=	msdyn_professionalsuffix	
PERSONPROFESSIONALTITLE	=	msdyn_professionatitle	
ADDRESSSTATEID	=	msdyn_addressstateid	
ADDRESSSTREET	=	msdyn_addressstreet	
ADDRESSSTREETNUMBER	=	msdyn_addressstreetnumber	
VENDORKNOWNASNAME	=	msdyn_vendorknownasname	
ADDRESSZIPCODE	=	msdyn_addresszipcode	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
DEFAULTPAYMENTDAYNAME	=	msdyn_defaultpaymentdayname.msdyn_name	
DEFAULTPAYMENTSCHEDULENAME	=	msdyn_paymentschedule.msdyn_name	
DEFAULTPAYMENTTERMSNAME	=	msdyn_paymentterms.msdyn_name	
HASONLYTAKENBIDS	><	msdyn_hasonlytakenbids	
ISMINORITYOWNED	><	msdyn_isminorityowned	
ISVENDORLOCALLYOWNED	><	msdyn_isvendorlocallyowned	
ISSERVICEVETERANOWNED	><	msdyn_isserviceveteranowned	
ISOWNERDISABLED	><	msdyn_ownerisdisabled	
ISWOMANOWNER	><	msdyn_womanowner	
PERSONANNIVERSARYDAY	=	msdyn_personanniversaryday	
PERSONANNIVERSARYYEAR	=	msdyn_anniversaryyear	
PERSONBIRTHDAY	=	msdyn_birthday	
PERSONBIRTHYEAR	=	msdyn_birthyear	
ORGANIZATIONEMPLOYEEAMOUNT	=	msdyn_numberofemployees	
VENDORHOLDRELEASEDATE	=	msdyn_vendoronholdreleasedate	
VENDORPARTYNUMBER	=	msdyn_vendorpartynumber	
ADDRESSLOCATIONID	=	msdyn_addresslocationid	
PERSONANNIVERSARYMONTH	=	msdyn_vendorpersonanniversarymonth	
PERSONBIRTHMONTH	=	msdyn_vendorpersonbirthmonth	
PERSONMARITALSTATUS	><	msdyn_maritalstatus	
ADDRESSLATITUDE	>>	msdyn_addresslatitude	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
ADDRESSLONGITUDE	>>	msdyn_addresslongitude	
ONHOLDSTATUS	><	msdyn_onholdstatus	
CURRENCYCODE	=	msdyn_currencycode.isocurrencycode	
ISVENDORLOCATEDINHUBZONE	><	msdyn_isvendorlocatedinhubzone	
DEFAULTVENDORPAYMENTMETHODNAME	=	msdyn_vendorpaymentmethod.msdyn_name	
INVOICEVENDORACCOUNTNUMBER	=	msdyn_invoicevendoraccountnumber.msdyn_vendoraccountnumber	
PERSONGENDER	><	msdyn_gender	
AREPRICESINCLUDINGSALESTAX	><	msdyn_priceincludessalestax	
SALESTAXGROUPCODE	=	msdyn_taxgroup.msdyn_name	
PRIMARYCONTACTPERSONID	=	msdyn_vendorprimarycontactperson.msdyn_contactpersonid	

Vendor groups to msdyn_vendorgroups

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
DEFAULTPAYMENTTERMNAME	=	msdyn_paymentterms.msdyn_name	
DESCRIPTION	=	msdyn_description	
VENDORGROUPID	=	msdyn_vendorgroup	
CLEARINGPERIODPAYMENTTERMNAME	=	msdyn_clearingperiodpaymentpermname.msdyn_name	

Vendor payment method to msdyn_vendorpaymentmethods

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
NAME	=	msdyn_name	
DESCRIPTION	=	msdyn_description	
SUMBYPERIOD	><	msdyn_sumbyperiod	
DISCOUNTGRACEPERIODDAYS	=	msdyn_discountgraceperioddays	
PAYMENTSTATUS	><	msdyn_paymentstatus	
ALLOWPAYMENTCOPIES	><	msdyn_allowpaymentcopies	
PAYMENTTYPE	><	msdyn_paymenttype	
LASTFILENUMBER	=	msdyn_lastfilenumber	
LASTFILENUMBERTODAY	=	msdyn_lastfilenumbertoday	
ACCOUNTTYPE	><	msdyn_accounttype	
BRIDGINGPOSTINGENABLED	><	msdyn_bridgingposting	
ENABLEPOSTDATEDCHECKCLEARINGPOSTING	><	msdyn_postdatedcheckclearingposting	
PROMISSORYNOTEDRAFTTYPE	><	msdyn_promissorynotedrafttype	
DIRECTDEBIT	><	msdyn_directdebit	

Switch between vendor designs

2/18/2021 • 2 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

Vendor data flow

If you choose to use the **Account** table to store vendors of the **Organization** type and the **Contact** table to store vendors of the **Person** type, configure the following workflows. Otherwise, this configuration isn't required.

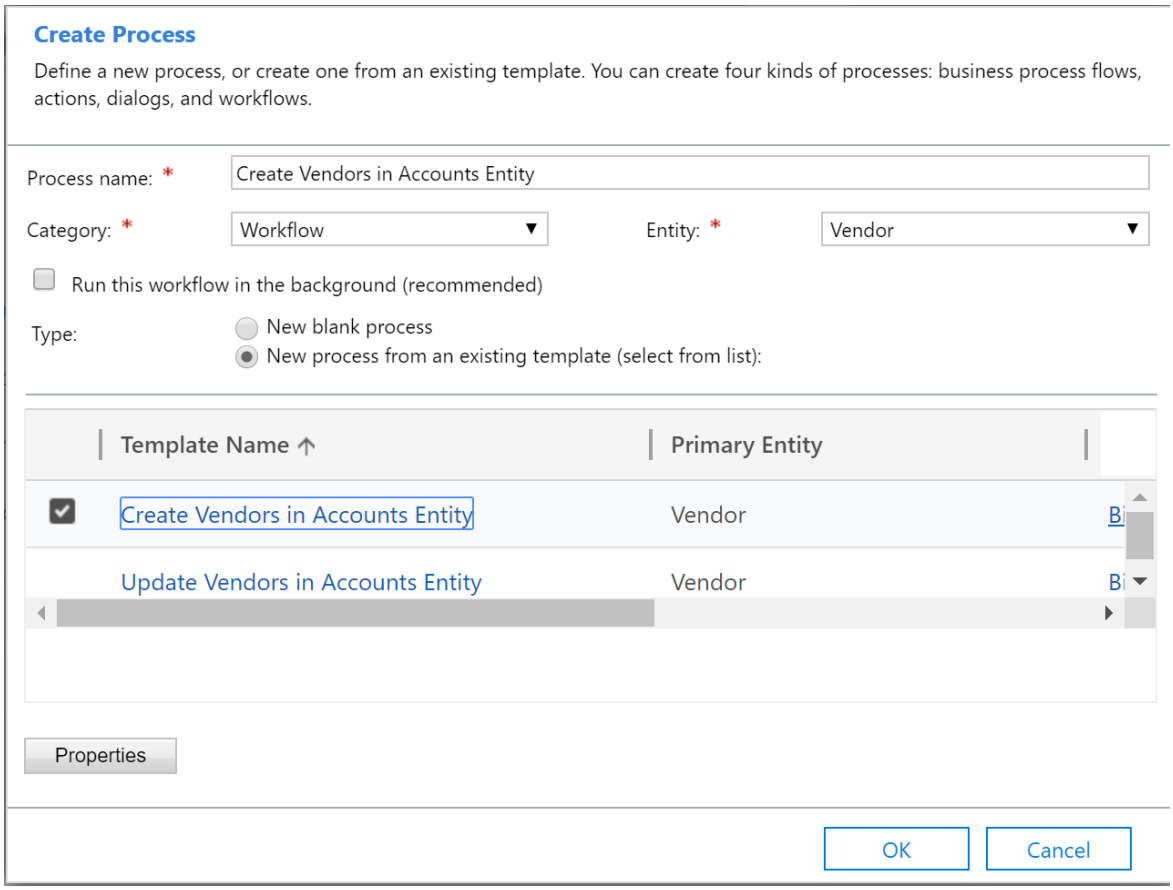
Use the extended vendor design for vendors of the Organization type

The **Dynamics365FinanceExtended** solution package contains the following workflow process templates. You will create a workflow for each template.

- Create Vendors in Accounts Table
- Create Vendors in Vendors Table
- Update Vendors in Accounts Table
- Update Vendors in Vendors Table

To create new workflow processes by using the workflow process templates, follow these steps.

1. Create a workflow process for the **Vendor** table, and select the **Create Vendors in Accounts Table** workflow process template. Then select **OK**. This workflow handles the vendor creation scenario for the **Account** table.

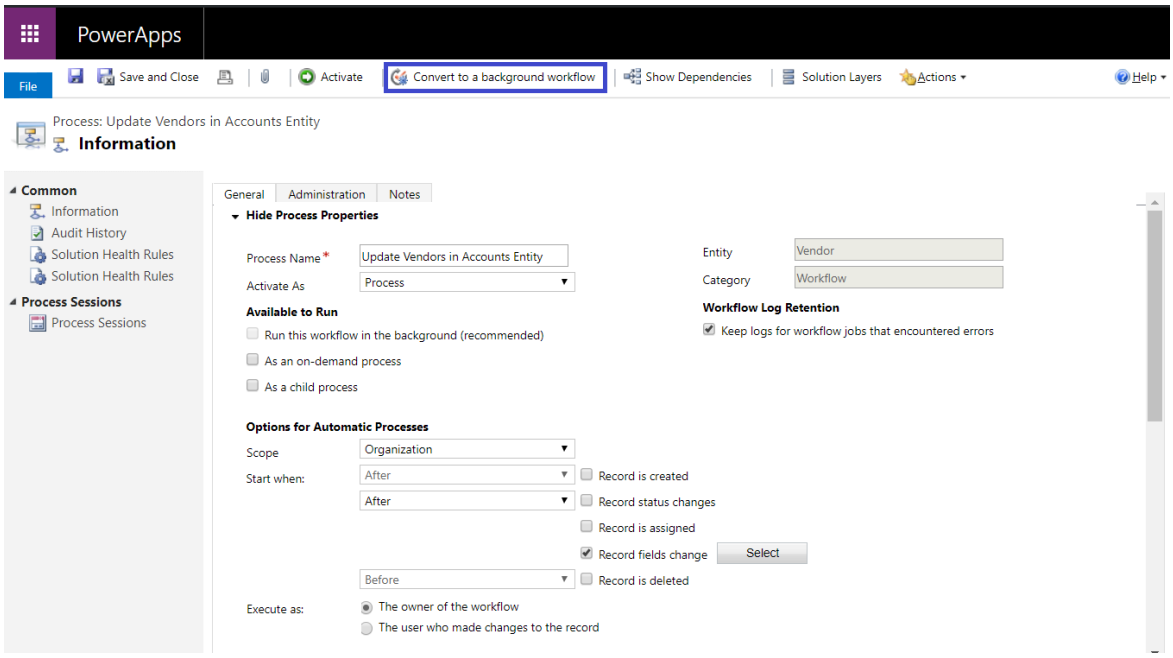


2. Create a workflow process for the **Vendor** table, and select the **Update Vendors in Accounts Table** workflow process template. Then select **OK**. This workflow handles the vendor update scenario for the **Account** table.

3. Create a workflow process for the **Account** table, and select the **Create Vendors in Vendors Table** workflow process template.

4. Create a workflow process for the **Account** table, and select the **Update Vendors in Vendors Table** workflow process template.

5. You can configure the workflows as either real-time workflows or background workflows, depending on your requirements. To configure a workflow as a background workflow, select **Convert to a background workflow**.



6. Activate the workflows that you created for the **Account** and **Vendor** tables to start to use the **Account** table to store information for vendors of the **Organization** type.

Use the extended vendor design for vendors of the Person type

The **Dynamics365FinanceExtended** solution package contains the following workflow process templates. You will create a workflow for each template.

- Create Vendors of type Person in Vendors Table
- Create Vendors of type Person in Contacts Table
- Update Vendors of type Person in Contacts Table
- Update Vendors of type Person in Vendors Table

To create new workflow processes by using the workflow process templates, follow these steps.

1. Create a workflow process for the **Vendor** table, and select the **Create Vendors of type Person in Contacts Table** workflow process template. Then select **OK**. This workflow handles the vendor creation scenario for the **Contact** table.
2. Create a workflow process for the **Vendor** table, and select the **Update Vendors of type Person in Contacts Table** workflow process template. Then select **OK**. This workflow handles the vendor update scenario for the **Contact** table.
3. Create a workflow process for the **Contact** table, and select the **Create Vendors of type Person in Vendors Table** template.
4. Create a workflow process for the **Contact** table, and select the **Update Vendors of type Person in Vendors Table** template.
5. You can configure the workflows as either real-time workflows or background workflows, depending on your requirements. To configure a workflow as a background workflow, select **Convert to a background workflow**.
6. Activate the workflows that you created on the **Contact** and **Vendor** tables to start to use the **Contact** table to store information for vendors of the **Person** type.

Customer loyalty cards and reward points

2/18/2021 • 2 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

Businesses classify customers and provide sophisticated services, based on customer shopping and spending patterns. For example, Dynamics 365 Commerce has the infrastructure and functions to facilitate and handle customer loyalty cards, reward points, loyalty-based pricing, and rewards-based shopping experiences. When data about customer loyalty cards and reward points in Commerce is synced to Dataverse, customer engagement apps can use that data. For example, Dynamics 365 Customer Service users can use the data to provide the same sophisticated services through the help desk.

Templates

FINANCE AND OPERATIONS APPS	MODEL-DRIVEN APPS IN DYNAMICS 365	DESCRIPTION
Loyalty card	msdyn_loyaltycards	This template syncs information about customer loyalty cards.
Loyalty reward points	msdyn_loyaltyrewardpoints	This template syncs information about customer reward points.

Mapping tables

Mapping types

There are several different mapping types. The following table explains the symbols used in the template tables.

SYMBOL	DESCRIPTION
>	One-way
>>	One-way, and data is transformed in the process.
=	Bidirectional
><	Bidirectional, and data is transformed in the process.
<<	One-way, and data is transformed in the process.

Filters

The source filter and reverse source filter determine which rows are synchronized.

Default values

If a synchronized field does not exist in either the Finance and Operations table or the other Dynamics 365 table, then a default value is assigned in the synchronized table. In some cases, the default value is an integer that is a lookup to an attribute value in the Common Data Model. For example, in the Contacts table of the Common Data Model, the default value of **address1_addresstypecode** is **3**. In the Common Data Model, for address1AddressTypeCode the value of **3** is **Primary address**.

Loyalty card to msdyn_loyaltycards

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
CARDNUMBER	=	msdyn_cardnumber	
CARDTENDERTYPE	><	msdyn_cardtendertype	
PARTYNUMBER	=	msdyn_partynumber	
REPLACEMENTCARDNUMBER	>	msdyn_replacementcardnumber	
OMOPERATINGUNITNUMBER	=	msdyn_operatingunitnumber	
LOYALTYENROLLMENTDATE	=	msdyn_enrollmentdate	

Loyalty reward points to msdyn_loyaltyrewardpoints

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
EXPIRATIONTIMEUNIT	><	msdyn_expirationtimeunit	
EXPIRATIONTIMEVALUE	=	msdyn_expirationtimevalue	
REDEEMABLE	><	msdyn_redeemable	
REDEEMRANKING	=	msdyn_redeemranking	
REWARDPOINTCURRENCY	=	msdyn_rewardpointcurrency.isocurrencycode	
REWARDPOINTID	=	msdyn_rewardpointid	
REWARDPOINTTYPE	><	msdyn_rewardpointtype	

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
MAXIMUMLOYALTYREWARDPOINTS	=	msdyn_maximumloyaltyrewardpoints	
VESTINGTIMEUNIT	><	msdyn_vestingtimeunit	
VESTINGTIMEVALUE	=	msdyn_vestingtimevalue	

Unified product experience

2/18/2021 • 22 minutes to read • Edit Online

NOTE

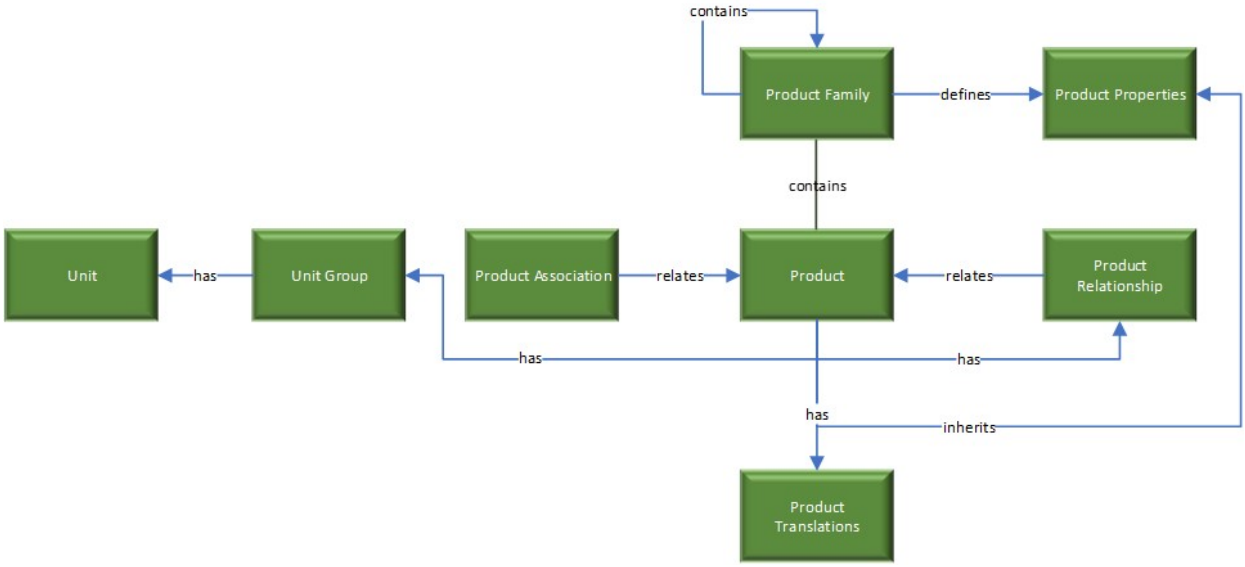
Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

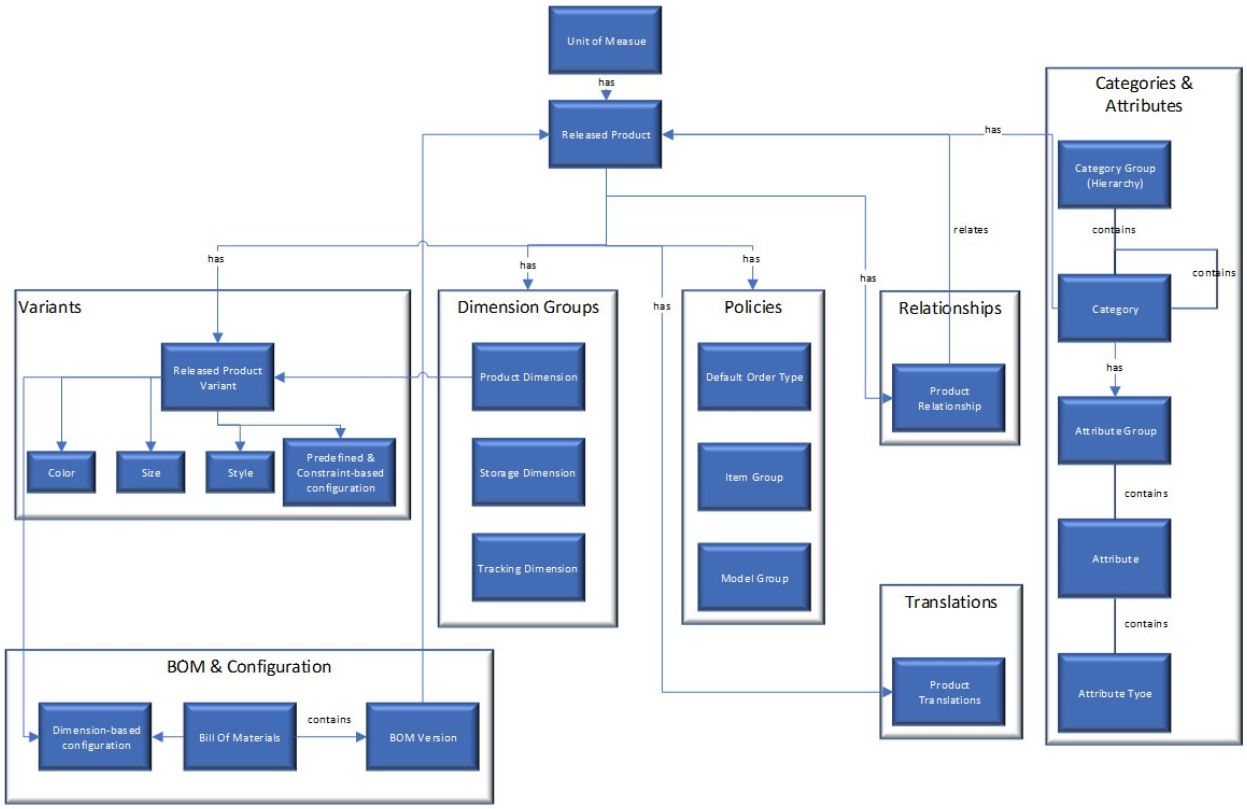
This topic will be updated soon to reflect the latest terminology.

When a business ecosystem is made up of Dynamics 365 applications, such as Finance, Supply Chain Management, and Sales, businesses often use these applications to source product data. This is because these apps provide a robust product infrastructure complemented with sophisticated pricing concepts and accurate on-hand inventory data. Businesses who use an external Product Lifecycle Management (PLM) system for sourcing the product data can channelize products from Finance and Operations apps to other Dynamics 365 apps. The unified product experience brings the integrated product data model in to Dataverse, so that all application users, including Power Platform users, can take advantage of the rich product data coming from Finance and Operations apps.

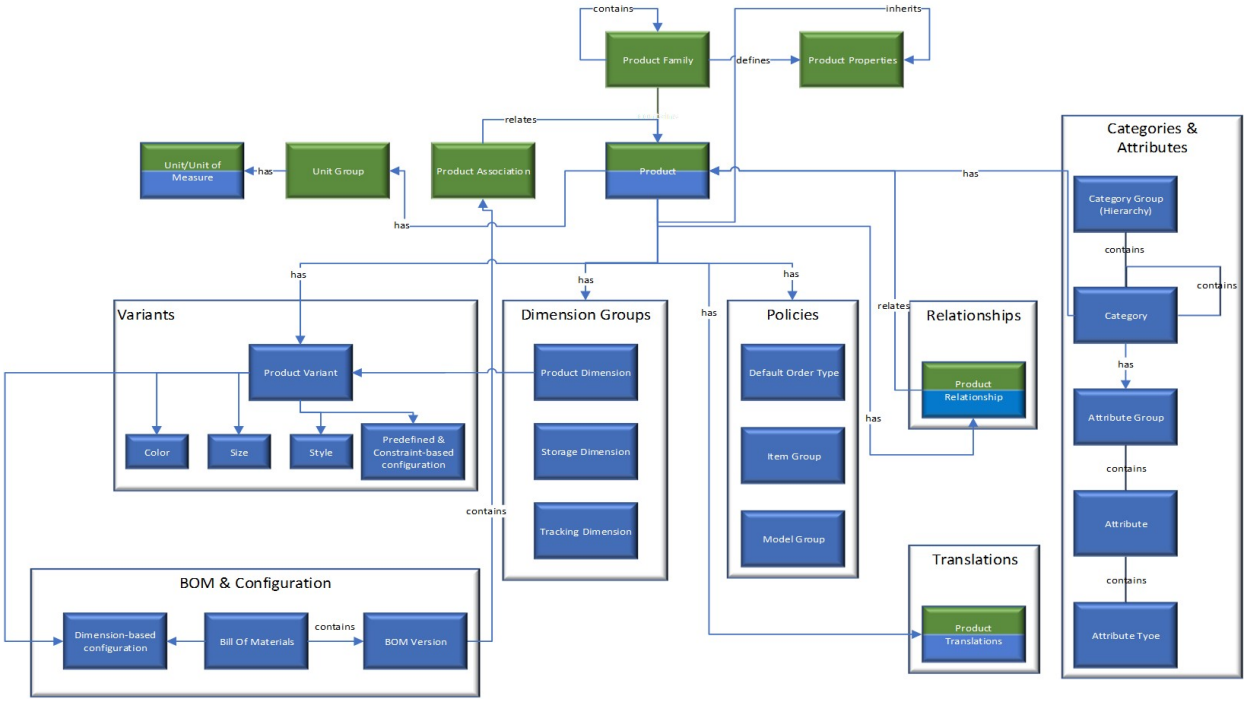
Here is the product data model from Sales.



Here is the product data model from Finance and Operations apps.



These two product data models have been integrated in Dataverse as shown below.



The dual-write table maps for products have been designed to flow data one-way only, in near-real time from Finance and Operations apps to Dataverse. However, the product infrastructure has been made open to make it bi-directional if required. Although you can customize it, it's at your own risk, as Microsoft does not recommend this approach.

Templates

Product information contains all the information related to the product and its definition, such as the product dimensions or the tracking and storage dimensions. As the following table shows, a collection of table maps is created to sync products and related information.

FINANCE AND OPERATIONS APPS	OTHER DYNAMICS 365 APPS	DESCRIPTION
Released products V2	msdyn_sharedproductdetails	The **msdyn_sharedproductdetails** table contains the columns from Finance and Operations apps that define the product, and that contain the product's financial and management information.
Dataverse released distinct products	Product	The **Product** table contains the columns that define the product. It includes individual products (products with subtype product) and the product variants. The following table shows the mappings.
Product number identified barcode	msdyn_productbarcodes	Product bar codes are used to uniquely identify products.
Default order settings	msdyn_productdefaultordersettings	
Product specific default order settings	msdyn_productdefaultordersettings	
Product dimension groups	msdyn_productdimensiongroups	The product dimension group defined which product dimensions define the product.
Storage dimension groups	msdyn_productstoragedimensiongroups	The product storage dimension group represents the method used to define the placement the product in the warehouse.
Tracking dimension groups	msdyn_producttrackingdimensiongroups	The product tracking dimension group represents the method used to track the product in inventory.
Colors	msdyn_productcolors	
Sizes	msdyn_productsizes	
Styles	msdyn_productsytles	
Configurations	msdyn_productconfigurations	
Product master colors	msdyn_sharedproductcolors	The **Shared product color** table indicates the colors that a specific product master can have. This concept is migrated to Dataverse to keep data consistent.
Product master sizes	msdyn_sharedproductsizes	The **Shared product size** table indicates the sizes that a specific product master can have. This concept is migrated to Dataverse to keep data consistent.

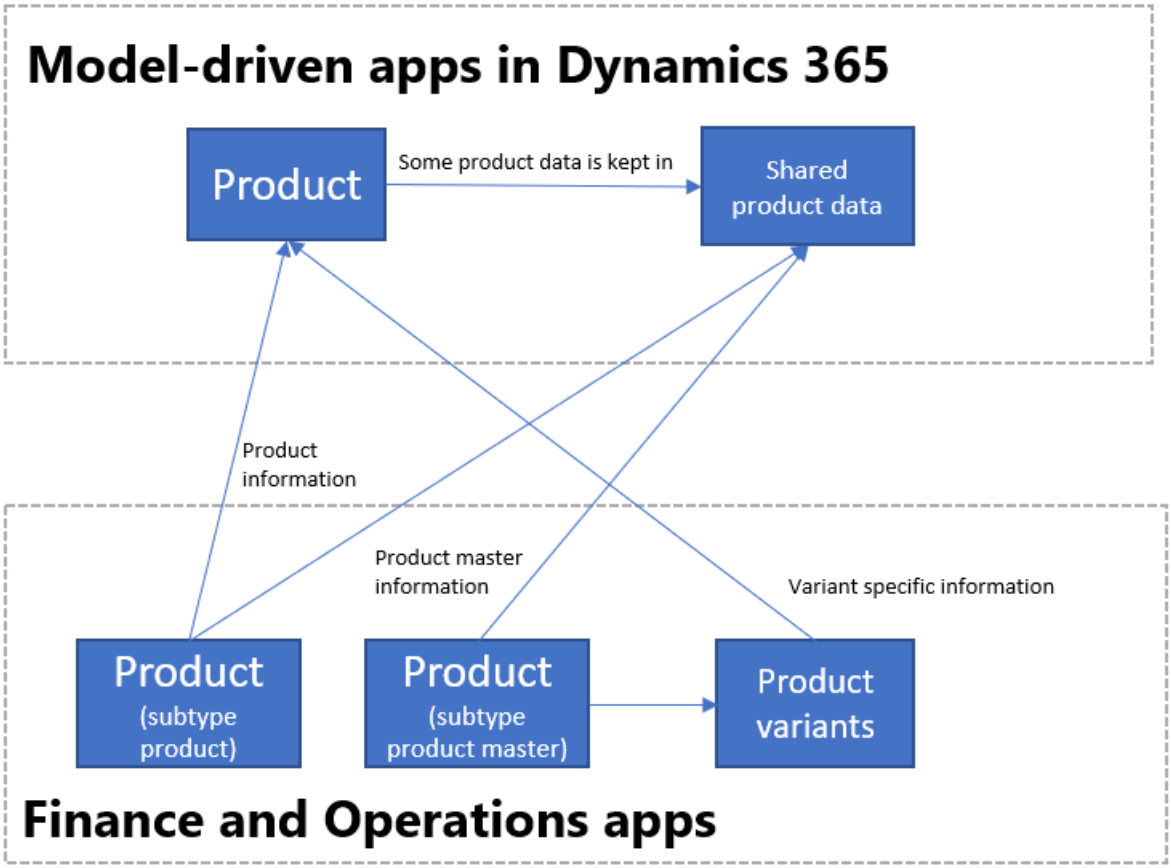
FINANCE AND OPERATIONS APPS	OTHER DYNAMICS 365 APPS	DESCRIPTION
Product master styles	msdyn_sharedproductstyles	The **Shared product style** table indicates the styles that a specific product master can have. This concept is migrated to Dataverse to keep data consistent.
Product master configurations	msdyn_sharedproductconfigurations	The **Shared product configuration** table indicates the configurations that a specific product master can have. This concept is migrated to Dataverse to keep data consistent.
All products	msdyn_globalproducts	The all products table contains all the products available in Finance and Operations apps, both the released products and the non-released products.
Unit	uoms	
Unit conversions	msdyn_ unitofmeasureconversions	
Product specific unit of measure conversion	msdyn_productspecificunitofmeasurec onversion	
Product categories	msdyn_productcategories	Each of the product categories and information about its structure and characteristics are contained in the product category table.
Product category hierarchies	msdyn_productcategoryhierarhies	You use product hierarchies to categorize or group products. The category hierarchies are available in Dataverse using the Product category hierarchy table.
Product category hierarchy roles	msdyn_productcategoryhierarchies	Product hierarchies can be used for different roles in D365 Finance and Operations. They specify which category is used in each role the product category role table is used.
Product category assignments	msdyn_productcategoryassignments	To assign a product to a category the product category assignments table can be used.

Integration of products

In this model, the product is represented by the combination of two tables in Dataverse: **Product** and **msdyn_sharedproductdetails**. Whereas the first table contains the definition of a product (the unique identifier for the product, the product name, and the description), the second table contains the columns stored at the product level. The combination of these two tables is used to define the product according to the concept of the stockkeeping unit (SKU). Each released product will have its information in the mentioned tables (Product and Shared Product Details). To keep track of all products (released and not released), the **Global products** table is used.

Because the product is represented as a SKU, the concepts of distinct products, product masters, and product variants can be captured in Dataverse in the following way:

- **Products with subtype product** are products that are defined by themselves. No dimensions have to be defined. An example is a specific book. For these products, one row is created in the **Product** table, and one row is created in the **msdyn_sharedproductdetails** table. No product family row is created.
- **Product masters** are used as generic products that hold the definition and rules that determine the behavior in business processes. Based on these definitions, distinct products that are known as product variants can be generated. For example, T-shirt is the product master, and it can have Color and Size as dimensions. Variants can be released that have different combinations of these dimensions, such a small blue T-shirt or a medium green T-shirt. In the integration, one row per variant is created in the product table. This row contains the variant-specific information, such as the different dimensions. The generic information for the product is stored in the **msdyn_sharedproductdetails** table. (This generic information is held in the product master.) The product master information is synced to Dataverse as soon as the released product master is created (but before variants are released).
- **Distinct products** refer to all the products subtype product and all the product variants.



With the dual-write functionality enabled, the products from Finance and Operations will be synchronized in other Dynamics 365 products in **Draft** state. They are added to the first price list with the same currency. In other words, they are added to the first price list in a Dynamics 365 app that matches the currency of your legal table where the product is released in a Finance and Operations app. If there is no price list for the given currency, a price list will automatically be created and the product will be assigned to it.

By default products from Finance and Operations apps are synchronized to other Dynamics 365 apps in **Draft** state. To synchronize the product with **Active** state so that you can directly use it in sales order quotations, for example, the following setting needs to be chosen: **System > Adminstration > System administration > System settings > Sales** tab and select **Create products in active state = yes**.

When products are synchronized, you must enter a value for the **Sales unit** field in the Finance and Operations app, because it is a mandatory field in Sales.

The synchronization of products happens from the Finance and Operations app to Dataverse. This means that the values of the product table columns can be changed in Dataverse, but when the synchronization is triggered (when a product column is modified in a Finance and Operations app), this will overwrite the values in Dataverse.

Mapping tables

Mapping types

There are several different mapping types. The following table explains the symbols used in the template tables.

SYMBOL	DESCRIPTION
>	One-way
>>	One-way, and data is transformed in the process.
=	Bidirectional
><	Bidirectional, and data is transformed in the process.
<<	One-way, and data is transformed in the process.

Filters

The source filter and reverse source filter determine which rows are synchronized.

Default values

If a synchronized field does not exist in either the Finance and Operations table or the other Dynamics 365 table, then a default value is assigned in the synchronized table. In some cases, the default value is an integer that is a lookup to an attribute value in the Common Data Model. For example, in the Contacts table of the Common Data Model, the default value of **address1_addresstypecode** is **3**. In the Common Data Model, for address1AddressTypeCode the value of **3** is **Primary address**.

CDS released distinct products to products

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
PRODUCTNUMBER	>>	msdyn_productnumber	
PRODUCTNAME	>>	name	
PRODUCTDESCRIPTION	>>	description	
ITEMNUMBER	>>	msdyn_itemnumber	
CURRENCYCODE	>>	transactioncurrencyid.isocurrencycode	
SALESUNITSYMBOL	>>	defaultuomid.msdyn_symbol	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
SALESPRICE	>>	price	
UNITCOST	>>	currentcost	
PRODUCTTYPE	>>	producttypecode	
SALESUNITDECIMALPRECISION	>>	quantitydecimal	0
ISCATCHWEIGHTPRODUCT	>>	msdyn_iscatchweight	
PRODUCTCOLORID	>>	msdyn_productcolor.msdyn_productcolorname	
PRODUCTCONFIGURATIONID	>>	msdyn_productconfiguration.msdyn_productconfiguration	
PRODUCTSIZEID	>>	msdyn_productsize.msdyn_productsize	
PRODUCTSTYLEID	>>	msdyn_productstyle.msdyn_productstyle	
ISTOCKEDPRODUCT	>>	msdyn_isstockedproduct	

Released products V2 to msdyn_sharedproductdetails

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
PRODUCTNUMBER	>	msdyn_globalproduct.msdyn_productnumber	
INTRASTATCHARGEPERCENTAGE	>	msdyn_intrastatchargepercentage	
ITEMNUMBER	>>	msdyn_itemnumber	
APPROXIMATESALESTAXPERCENTAGE	>	msdyn_approximatesalestaxpercentage	
BESTBEFOREPERIODDAYS	>	msdyn_bestbeforeperioddays	
CARRYINGCOSTABCCODE	>>	msdyn_carryingcostabccode	
CONSTANTSCRAPQUANTITY	>	msdyn_constantscrapquantity	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
COSTCHARGESQUANTITY	>	msdyn_costchargesquantity	
DEFAULTRECEIVINGQUANTITY	>	msdyn_defaultreceivingquantity	
FIXEDPURCHASEPRICECHARGES	>	msdyn_fixedpurchasepricecharges	
FIXEDSALESPRICECHARGES	>	msdyn_fixedsalespricecharges	
GROSSDEPTH	>	msdyn_grossdepth	
GROSSPRODUCTHEIGHT	>	msdyn_grossproductheight	
GROSSPRODUCTWIDTH	>	msdyn_grossproductwidth	
INVENTORYUNITSYMBOL	>	msdyn_inventoryunitsymbol.msdyn_symbol	
ISDISCOUNTPOSREGISTRATIONPROHIBITED	> >	msdyn_isdiscountposregistrationprohibited	
ISEXEMPTFROMAUTOMATICNOTIFICATIONANDCANCELLATION	> >	msdyn_exemptautomaticnotificationcancel	
ISINSTALLMENTELIGIBLE	> >	msdyn_isinstallmenteligible	
ISINTERCOMPANYPURCHASEUSAGEBLOCKED	> >	msdyn_isintercompanypurchaseusageblocked	
ISINTERCOMPANYSALESUSAGEBLOCKED	> >	msdyn_isintercompanysalesusageblocked	
ISMANUALDISCOUNTPOSREGISTRATIONPROHIBITED	> >	msdyn_ismanualdiscposregistrationprohibited	
ISPHANTOM	> >	msdyn_isphantom	
ISPOSREGISTRATIONBLOCKED	> >	msdyn_isposregistrationblocked	
ISPOSREGISTRATIONQUANTITYNEGATIVE	> >	msdyn_isposregistrationquantitynegative	
ISPURCHASEPRICEAUTOMATICALLYUPDATED	> >	msdyn_ispurchasepriceautomaticallyupdated	
ISPURCHASEPRICEINCLUDINGCHARGES	> >	msdyn_ispurchasepriceincludingcharges	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
ISSALESWITHHOLDINGTAXCALCULATED	>>	msdyn_issaleswithholdingtaxcalculated	
ISRESTRICTEDFORCOUPONS	>>	msdyn_isrestrictedforcoupons	
ISSALESPRICEADJUSTMENTALLOWED	>>	msdyn_issalespriceadjustmentallowed	
ISSALESPRICEINCLUDINGCHARGES	>>	msdyn_issalespriceincludingcharges	
ISSCALEPRODUCT	>>	msdyn_isscaleproduct	
ISSHIPALONEENABLED	>>	msdyn_isshipaloneenabled	
ISUNITCOSTPRODUCTVARIANTSPECIFIC	>>	msdyn_isunitcostproductvariantspecific	
ISVARIANTSHELFLABELSPRINTINGENABLED	>>	msdyn_isvariantshelflabelsprintingenabled	
ISZEROPRICEPOSREGISTRATIONALLOWED	>>	msdyn_iszeropriceposregistrationallowed	
KEYINPRICEREQUIREMENTSATPOSREGISTER	>>	msdyn_keyinpricerequirementsatposregister	
KEYINQUANTITYREQUIREMENTSATPOSREGISTER	>>	msdyn_keyinquantityrequirementsatposregister	
MARGINABCCODE	>>	msdyn_marginabccode	
MAXIMUMPICKQUANTITY	>	msdyn_maximumpickquantity	
MUSTKEYINCOMMENTATPOSREGISTER	>>	msdyn_mustkeyincommentatposregister	
NECESSARYPRODUCTIONWORKINGTIMESCHEDULINGPROPERTYID	>	msdyn_necessaryproductionworkingtimeschedulingp	
NETPRODUCTWEIGHT	>	msdyn_netproductweight	
PACKINGDUTYQUANTITY	>	msdyn_packingdutyquantity	
POSREGISTRATIONACTIVATIONDATE	>	msdyn_posregistrationactivationdate	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
POSREGISTRATIONBLOCKEDDATE	>	msdyn_posregistrationblockeddate	
POSREGISTRATIONPLANNEDBLOCKEDDATE	>	msdyn_posregistrationplannedblockeddate	
POTENCYBASEATTIBUTETARGETVALUE	>	msdyn_potencybaseattibutetargetvalue	
POTENCYBASEATTRIBUTEVALUEENTRYEVENT	>>	msdyn_potencybaseattributevalueentryevent	
PRODUCTTYPE	>>	msdyn_producttype	
PRODUCTIONCONSUMPTIONDENSITYCONVERSIONFACTOR	>	msdyn_productionconsumptiondensityconversion	
PRODUCTIONCONSUMPTIONDEPTHCONVERSIONFACTOR	>	msdyn_productionconsumptiondepthconversion	
PRODUCTIONCONSUMPTIONHEIGHTCONVERSIONFACTOR	>	msdyn_productionconsumptionheightconversion	
PRODUCTIONCONSUMPTIONWIDTHCONVERSIONFACTOR	>	msdyn_productionconsumptionwidthconversion	
PRODUCTVOLUME	>	msdyn_productvolume	
PURCHASECHARGESQUANTITY	>	msdyn_purchasechargesquantity	
PURCHASEOVERDELIVERYPERCENTAGE	>	msdyn_purchaseoverdeliverypercentage	
PURCHASEPRICE	>	msdyn_purchaseprice	
PURCHASEPRICEDATE	>	msdyn_purchasepricedate	
PURCHASEPRICINGPRECISION	>	msdyn_purchasepricingprecision	
PURCHASEUNDERDELIVERYPERCENTAGE	>	msdyn_purchaseunderdeliverypercentage	
RAWMATERIALPICKINGPRINCIPLE	>>	msdyn_rawmaterialpickingprinciple	
SALESCHARGESQUANTITY	>	msdyn_saleschargesquantity	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
SALESOVERDELIVERYPERCENTAGE	>	msdyn_salesoverdeliverypercentage	
SALESPRICE	>	msdyn_salesprice	
SALESPRICECALCULATIONCHARGESPERCENTAGE	>	msdyn_salespricecalculationchargespercentage	
SALESPRICECALCULATIONCONTRIBUTIONRATIO	>	msdyn_salespricecalculationcontributionratio	
SALESPRICECALCULATIONMODEL	>>	msdyn_salespricecalculationmodel	
SALESPRICEDATE	>	msdyn_salespricedate	
SALESPRICINGPRECISION	>	msdyn_salespricingprecision	
SALESUNDERDELIVERYPERCENTAGE	>	msdyn_salesunderdeliverypercentage	
SALESUNITSYMBOL	>	msdyn_salesunitsymbol.msdyn_symbol	
SCALEINDICATOR	>>	msdyn_scaleindicator	
SELLSTARTDATE	>	msdyn_sellstartdate	
SHELFADVICEPERIODDAYS	>	msdyn_shelfadviceperioddays	
SHELFLIFEPERIODDAYS	>	msdyn_shelflifeperioddays	
SHIPSTARTDATE	>	msdyn_shipstartdate	
TAREPRODUCTWEIGHT	>	msdyn_tareproductweight	
TRANSFERORDEROVERDELIVERYPERCENTAGE	>	msdyn_transferorderoverdeliverypercentage	
TRANSFERORDERUNDERDELIVERYPERCENTAGE	>	msdyn_transferorderunderdeliverypercentage	
UNITCOST	>	msdyn_unitcost	
UNITCOSTDATE	>	msdyn_unitcostdate	
UNITCOSTQUANTITY	>	msdyn_unitcostquantity	
VARIABLESCRAPPERCENTAGE	>	msdyn_variablescrappercentage	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
WAREHOUSEMOBILEDEVICEDESCRIPTIONLINE1	>	msdyn_warehousemobiledevicedescriptionline1	
WAREHOUSEMOBILEDEVICEDESCRIPTIONLINE2	>	msdyn_warehousemobiledevicedescriptionline2	
WILLINVENTORYISSUEAUTOMATICALLYREPORTASFINISHED	> >	msdyn_willinventoryissueautoreportasfinished	
WILLINVENTORYRECEIPTIGNOREFLUSHINGPRINCIPLE	> >	msdyn_willinventoryreceiptignoreflushing	
WILLPICKINGWORKBENCHAPPLYBOXINGLOGIC	> >	msdyn_willpickingworkbenchapplyboxinglogic	
WILLTOTALPURCHASEDISCOUNTCALCULATIONINCLUDEPRODUCT	> >	msdyn_willtotalpurchdiscountcalcincludeproduct	
WILLTOTALSALESDISCOUNTCALCULATIONINCLUDEPRODUCT	> >	msdyn_willtotalsalesdiscountcalcincludeproduct	
WILLWORKCENTERPICKINGALLOWNEGATIVEINVENTORY	> >	msdyn_willworkcenterpickingallownegativeinvent	
YIELDPERCENTAGE	>	msdyn_yieldpercentage	
ISUNITCOSTAUTOMATICALLYUPDATED	> >	msdyn_isunitcostautomaticallyupdated	
PURCHASEUNITSYMBOL	>	msdyn_purchaseunitsymbol.msdyn_symbol	
PURCHASEPRICEQUANTITY	>	msdyn_purchasepricequantity	
ISUNITCOSTINCLUDINGCHARGES	> >	msdyn_isunitcostincludingcharges	
FIXEDCOSTCHARGES	> >	msdyn_fixedcostcharges	
MINIMUMCATCHWEIGHTQUANTITY	> >	msdyn_minimumcatchweightquantity	
MAXIMUMCATCHWEIGHTQUANTITY	> >	msdyn_maximumcatchweightquantity	
ALTERNATIVEITEMNUMBER	> >	msdyn_alternativeitemnumber.msdyn_itemnumber	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
BOMUNITSYMBOL	>>	msdyn_bomunitsymbol.msdyn_symbol	
CATCHWEIGHTUNITSYMBOL	>>	msdyn_catchweightunitsymbol.msdyn_symbol	
COMPARISONPRICEBASEUNITSYMBOL	>>	msdyn_comparisonpricebaseunitsymbol.msdyn_symbol	
PRIMARYVENDORACCOUNTNUMBER	>>	msdyn_vendorid.msdyn_vendoraccountnumber	
ISCATCHWEIGHTPRODUCT	>>	msdyn_iscatchweight	
PRODUCTDIMENSIONGROUPNAME	>>	msdyn_productdimensiongroupid.msdyn_groupname	

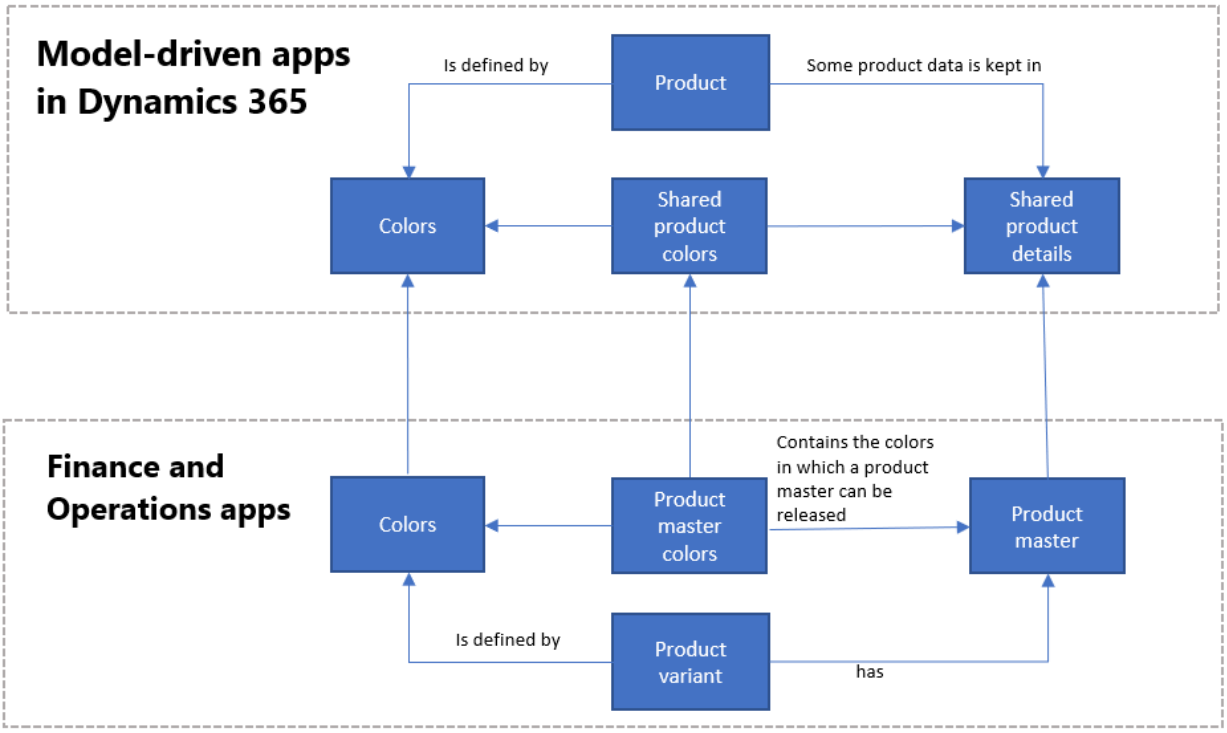
All products to msdyn_globalproducts

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
PRODUCTNAME	>>	msdyn_productname	
PRODUCTNUMBER	>>	msdyn_productnumber	

Product dimensions

Product dimensions are characteristics that identify a product variant. The four product dimensions (Color, Size, Style, and Configuration) are also mapped to Dataverse to define the product variants. The following illustration shows the data model for the product dimension Color. The same model is applied to Sizes, Styles and Configurations.



Colors to msdyn_productcolors

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
COLORID	>>	msdyn_productcolorname	

Sizes to msdyn_productsizes

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
SIZEID	>>	msdyn_productsize	

Styles to msdyn_productstyles

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
STYLEID	>>	msdyn_productstyle	

Configurations to msdyn_productconfigurations

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
CONFIGURATIONID	>>	msdyn_productconfiguration	

When a product has different product dimensions (for example, a product master has Size and Color as product dimensions), each distinct product (that is, each product variant) is defined as a combination of those product dimensions. For example, product number B0001 is an extra-small black T-shirt, and product number B0002 is a small black T-shirt. In this case, the existing combinations of product dimensions are defined. For example, the T-shirt from the preceding example can be extra-small and black, small and black, medium and black, or large and black, but it can't be extra-large and black. In other words, the product dimensions that a product master can take are specified, and variants can be released based on these values.

To keep track of the product dimensions that a product master can take, the following tables are created and mapped in Dataverse for each product dimension. For more information, see Product information overview.

Product master colors to msdyn_sharedproductcolors

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
PRODUCTCOLORID	>>	msdyn_productcolor.msdyn_productcolorname	
PRODUCTMASTERNUMBER	>>	msdyn_globalproduct.msdyn_productnumber	
REPLENISHMENTWEIGHT	>>	msdyn_replenishmentweight	
DISPLAYSEQUENCENUMBER	>>	msdyn_displaysequencenumber	

Product master sizes to msdyn_sharedproductsizes

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
PRODUCTMASTERNUMBER	>>	msdyn_globalproduct.msdyn_productnumber	
PRODUCTSIZEID	>>	msdyn_productsize.msdyn_productsize	
REPLENISHMENTWEIGHT	>>	msdyn_replenishmentweight	
DISPLAYSEQUENCENUMBER	>>	msdyn_displaysequencenumber	

Product master styles to msdyn_sharedproductstyles

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
PRODUCTMASTERNUMBER	>>	msdyn_globalproduct.msdyn_productnumber	
PRODUCTSTYLEID	>>	msdyn_productstyle.msdyn_productstyle	
REPLENISHMENTWEIGHT	>>	msdyn_replenishmentweight	
DISPLAYSEQUENCENUMBER	>>	msdyn_displaysequencenumber	

Product master configurations to msdyn_sharedproductconfigurations

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
CONTAINERUNITSYMBOL	>>	msdyn_containerunit.msdyn_symbol	
PRODUCTCONFIGURATIONID	>>	msdyn_productconfiguration.msdyn_productconfiguration	
PRODUCTMASTERNUMBER	>>	msdyn_globalproduct.msdyn_productnumber	
REPLENISHMENTWEIGHT	>>	msdyn_replenishmentweight	
DISPLAYSEQUENCENUMBER	>>	msdyn_displaysequencenumber	

Product Number Identified Barcode to msdyn_productbarcodes

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
PRODUCTNUMBER	>	msdyn_productnumberid.msdyn_productnumber	
BARCODE	>	msdyn_name	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
BARCODE	>	msdyn_barcode	
PRODUCTQUANTITY	>	msdyn_productquantity	
PRODUCTDESCRIPTION	>	msdyn_productdescription	
BARCODESETUPID	>	msdyn_barcodesetupid	
PRODUCTQUANTITYUNITSYMBOL	>	msdyn_unitofmeasureid.msdyn_symbol	
ISDEFAULTSCANNEDBARCODE	>>	msdyn_isdefaultscannedbarcode	
ISDEFAULTPRINTEDBARCODE	>>	msdyn_isdefaultprintedbarcode	
ISDEFAULTDISPLAYEDBARCODE	>>	msdyn_isdefaultdisplayedbarcode	

Default order settings and product-specific default order settings

Default order settings define the site and warehouse where items will be sourced from or stored, the minimum, maximum, multiple and standard quantities that will be used for trading or inventory management, the lead times, the stop flag, and the order promising method. This information is available in Dataverse using the default order settings and product-specific default order settings entity. You can read more information about the functionality in the Default order settings topic.

Default order settings to msdyn_productdefaultordersettings

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
INVENTWAREHOUSEID	=	msdyn_inventorywarehouse.msdyn_warehouseidentifier	
INVENTORYSITEID	=	msdyn_inventorysite.msdyn_siteid	
INVENTORYATPDELAYEDDEMANDOFFSETDAYS	=	msdyn_inventoryatpdelayeddemandoffsetdays	
INVENTORYATPDELAYEDSUPPLYOFFSETDAYS	=	msdyn_inventoryatpdelayedsupplyoffsetdays	
ITEMNUMBER	=	msdyn_itemnumber.msdyn_itemnumber	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
INVENTORYATPBACKWARDDEMANDTIMEFENCEDAYS	=	msdyn_inventoryatpbackwarddemandtimefencedays	
INVENTORYATPBACKWARDSUPPLYTIMEFENCEDAYS	=	msdyn_inventoryatpbackwardsupplytimefencedays	
INVENTORYATPTIMEFENCEDAYS	=	msdyn_inventoryatptimefencedays	
MAXIMUMINVENTORYORDERQUANTITY	=	msdyn_maximuminventoryorderquantity	
MAXIMUMPROCUREMENTORDERQUANTITY	=	msdyn_maximumprocurementorderquantity	
MAXIMUMSALESORDERQUANTITY	=	msdyn_maximumsalesorderquantity	
MINIMUMINVENTORYORDERQUANTITY	=	msdyn_minimuminventoryorderquantity	
MINIMUMPROCUREMENTORDERQUANTITY	=	msdyn_minimumprocurementorderquantity	
MINIMUMSALESORDERQUANTITY	=	msdyn_minimumsalesorderquantity	
STANDARDINVENTORYORDERQUANTITY	=	msdyn_standardinventoryorderquantity	
STANDARDPROCUREMENTORDERQUANTITY	=	msdyn_standardprocurementorderquantity	
STANDARDSALESORDERQUANTITY	=	msdyn_standardsalesorderquantity	
INVENTORYLEADTIMEDAYS	=	msdyn_inventoryleadtimedays	
INVENTORYQUANTITYMULTIPLES	=	msdyn_inventoryquantitymultiples	
PROCUREMENTQUANTITYMULTIPLES	=	msdyn_procurementquantitymultiples	
SALESQUANTITYMULTIPLES	=	msdyn_salesquantitymultiples	
PROCUREMENTSITEID	=	msdyn_procurementsite.msdyn_siteid	
PROCUREMENTLEADTIMEDAYS	=	msdyn_procurementleadtimedays	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
SALESSITEID	=	msdyn_salessite.msdyn_siteid	
SALESATPDELAYEDDEMANDOFFSETDAYS	=	msdyn_salesatpdelayeddemandoffsetdays	
SALESATPDELAYEDSUPPLYOFFSETDAYS	=	msdyn_salesatpdelayedsupplyoffsetdays	
SALESATPBACKWARDDEMANDTIMEFENCEDAYS	=	msdyn_salesatpbackwarddemandtimefencedays	
SALESATPBACKWARDSUPPLYTIMEFENCEDAYS	=	msdyn_salesatpbackwardsupplytimefencedays	
SALESATPTIMEFENCEDAYS	=	msdyn_salesatptimefencedays	
SALESLEADTIMEDAYS	=	msdyn_salesleadtimedays	
PROCUREMENTWAREHOUSEID	=	msdyn_procurementwarehouse.msdyn_warehouseidentifier	
SALESWAREHOUSEID	=	msdyn_saleswarehouse.msdyn_warehouseidentifier	
AREINVENTORYORDERPROMISINGDEFAULTSOVERRIDDEN	> <	msdyn_areinventoryorderdefaultsoverridden	
INVENTORYORDERPROMISINGMETHOD	> <	msdyn_inventoryorderpromisingmethod	
ISINVENTORYATPINCLUDINGPLANNEDORDERS	> <	msdyn_isinventoryatpincludingplannedorders	
ISINVENTORYUSINGWORKINGDAYS	> <	msdyn_isinventoryusingworkingdays	
ISINVENTORYSITEMANDATORY	> <	msdyn_isinventorysitemandatory	
ISINVENTORYPROCESSINGSTOPPED	> <	msdyn_isinventoryprocessingstopped	
ISPROCUREMENTUSINGWORKINGDAYS	> <	msdyn_isprocurementusingworkingdays	
ISPROCUREMENTSITEMANDATORY	> <	msdyn_isprocurementsitemandatory	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
ISPROCUREMENTPROCESSINGSTOPPED	> <	msdyn_isprocurementprocessingstopped	
ARESALESORDERPROMISINGDEFAULTSOVERRIDDEN	> <	msdyn_aresalesorderdefaultsoverridden	
SALESORDERPROMISINGMETHOD	> <	msdyn_salesorderpromisingmethod	
ISSALESATPINCLUDINGPLANNEDORDERS	> <	msdyn_issalesatpincludingplannedorders	
ISSALESSITEMANDATORY	> <	msdyn_issalessitemandatory	
ISSALESLEADTIMEOVERRIDDEN	> <	msdyn_issalesleadtimeoverridden	
ISSALESPROCESSINGSTOPPED	> <	msdyn_issalesprocessingstopped	
ISINVENTORYWAREHOUSEMANDATORY	> <	msdyn_isinventorywarehousemandatory	
ISPROCUREMENTWAREHOUSEMANDATORY	> <	msdyn_isprocurementwarehousemandatory	
ISSALESWAREHOUSEMANDATORY	> <	msdyn_issaleswarehousemandatory	

Product default order settings V2 to msdyn_productspecificdefaultordersettings

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
INVENTORYWAREHOUSEID	=	msdyn_inventorywarehouse.msdyn_warehouseidentifier	
INVENTORYSITEID	=	msdyn_inventorysite.msdyn_siteid	
INVENTORYATPDELAYEDDEMANDOFFSETDAYS	=	msdyn_inventoryatpdelayeddemandoffsetdays	
INVENTORYATPDELAYEDSUPPLYOFFSETDAYS	=	msdyn_inventoryatpdelayedsupplyoffsetdays	
ITEMNUMBER	=	msdyn_itemnumber.msdyn_itemnumber	

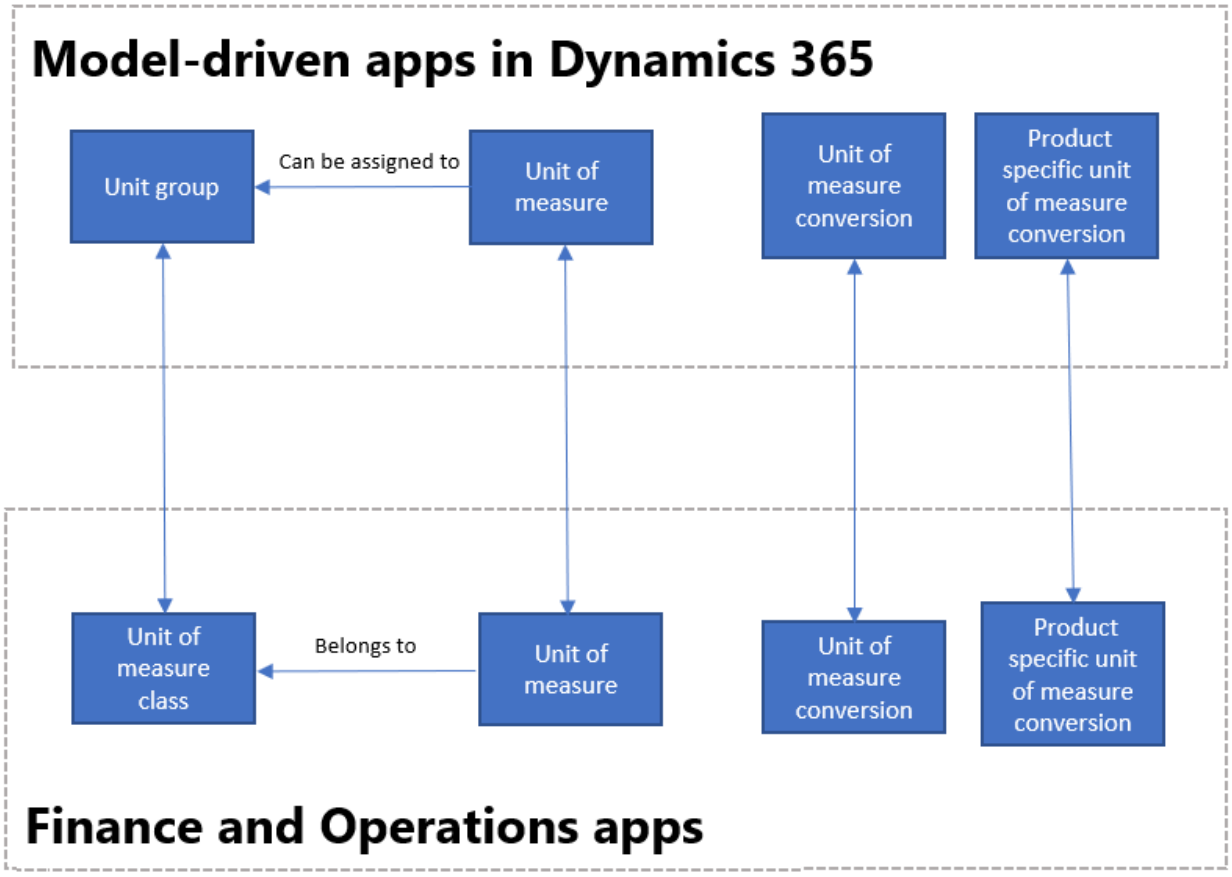
FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
INVENTORYATPBACKWARDDEMANDTIMEFENCEDAYS	=	msdyn_inventoryatpbackwarddemandtimefencedays	
INVENTORYATPBACKWARDSUPPLYTIMEFENCEDAYS	=	msdyn_inventoryatpbackwardsupplytimefencedays	
INVENTORYATPTIMEFENCEDAYS	=	msdyn_inventoryatptimefencedays	
MAXIMUMINVENTORYORDERQUANTITY	=	msdyn_maximuminventoryorderquantity	
MAXIMUMPROCUREMENTORDERQUANTITY	=	msdyn_maximumprocurementorderquantity	
MAXIMUMSALESORDERQUANTITY	=	msdyn_maximumsalesorderquantity	
MINIMUMINVENTORYORDERQUANTITY	=	msdyn_minimuminventoryorderquantity	
MINIMUMPROCUREMENTORDERQUANTITY	=	msdyn_minimumprocurementorderquantity	
MINIMUMSALESORDERQUANTITY	=	msdyn_minimumsalesorderquantity	
STANDARDINVENTORYORDERQUANTITY	=	msdyn_standardinventoryorderquantity	
STANDARDPROCUREMENTORDERQUANTITY	=	msdyn_standardprocurementorderquantity	
STANDARDSALESORDERQUANTITY	=	msdyn_standardsalesorderquantity	
INVENTORYLEADTIMEDAYS	=	msdyn_inventoryleadtimedays	
INVENTORYQUANTITYMULTIPLES	=	msdyn_inventoryquantitymultiples	
PROCUREMENTQUANTITYMULTIPLES	=	msdyn_procurementquantitymultiples	
SALESQUANTITYMULTIPLES	=	msdyn_salesquantitymultiples	
PROCUREMENTSITEID	=	msdyn_procurementsite.msdyn_siteid	
PROCUREMENTLEADTIMEDAYS	=	msdyn_procurementleadtimedays	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
SALESSITEID	=	msdyn_salessite.msdyn_siteid	
SALESATPDELAYEDDEMANDOFFSETDAYS	=	msdyn_salesatpdelayeddemandoffsetdays	
SALESATPDELAYEDSUPPLYOFFSETDAYS	=	msdyn_salesatpdelayedsupplyoffsetdays	
SALESATPBACKWARDDEMANDTIMEFENCEDAYS	=	msdyn_salesatpbackwarddemandtimefencedays	
SALESATPBACKWARDSUPPLYTIMEFENCEDAYS	=	msdyn_salesatpbackwardsupplytimefencedays	
SALESATPTIMEFENCEDAYS	=	msdyn_salesatptimefencedays	
SALESLEADTIMEDAYS	=	msdyn_salesleadtimedays	
PROCUREMENTWAREHOUSEID	=	msdyn_procurementwarehouse.msdyn_warehouseidentifier	
SALESWAREHOUSEID	=	msdyn_saleswarehouse.msdyn_warehouseidentifier	
AREINVENTORYDEFAULTORDERSETTINGSOVERRIDDEN	> <	msdyn_areinventoryorderdefaultsoverridden	
INVENTORYORDERPROMISINGMETHOD	> <	msdyn_inventoryorderpromisingmethod	
ISINVENTORYATPINCLUDINGPLANNEDORDERS	> <	msdyn_isinventoryatpincludingplannedorders	
ISINVENTORYUSINGWORKINGDAYS	> <	msdyn_isinventoryusingworkingdays	
ISINVENTORYSITEMANDATORY	> <	msdyn_isinventorysitemandatory	
ISINVENTORYPROCESSINGSTOPPED	> <	msdyn_isinventoryprocessingstopped	
ISPROCUREMENTUSINGWORKINGDAYS	> <	msdyn_isprocurementusingworkingdays	
ISPROCUREMENTSITEMANDATORY	> <	msdyn_isprocurementsitemandatory	
ISPROCUREMENTPROCESSINGSTOPPED	> <	msdyn_isprocurementprocessingstopped	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
ARESALESDEFAULTORDERSETTINGSOVERRIDDEN	> <	msdyn_aresalesorderdefaultsoverridden	
SALESORDERPROMISINGMETHOD	> <	msdyn_salesorderpromisingmethod	
ISSALESATPINCLUDINGPLANNEDORDERS	> <	msdyn_issalesatpincludingplannedorders	
ISSALESSITEMANDATORY	> <	msdyn_issalessitemandatory	
ISSALESLEADTIMEOVERRIDDEN	> <	msdyn_issalesleadtimeoverridden	
ISSALESPROCESSINGSTOPPED	> <	msdyn_issalesprocessingstopped	
ISINVENTORYWAREHOUSEMANDATORY	> <	msdyn_isinventorywarehousemandatory	
ISPROCUREMENTWAREHOUSEMANDATORY	> <	msdyn_isprocurementwarehousemandatory	
ISSALESWAREHOUSEMANDATORY	> <	msdyn_issaleswarehousemandatory	
OPERATIONALSITEID	=	msdyn_operationalsite.msdyn_siteid	
PRODUCTCOLORID	=	msdyn_productcolor.msdyn_productcolorname	
PRODUCTCONFIGURATIONID	=	msdyn_productconfiguration.msdyn_productconfiguration	
PRODUCTSIZEID	=	msdyn_productsize.msdyn_productsize	
PRODUCTSTYLEID	=	msdyn_productstyle.msdyn_productstyle	

Unit of measure and unit of measure conversions

The units of measure and its respective conversion are available in the Dataverse following the data model shown in the diagram.



The unit of measure concept is integrated between Finance and Operations apps and other Dynamics 365 apps. For each unit class in a Finance and Operations app, a unit group is created in a Dynamics 365 app, which contains the units belonging to the unit class. A default base unit is also created for every unit group.

Units to uoms

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
UNITSYMBOL	>>	msdyn_symbol	
UNITCLASS	>>	msdyn_externalunitclassname	
DECIMALPRECISION	>>	msdyn_decimalprecision	
ISBASEUNIT	>>	msdyn_isbaseunit	
ISSYSTEMUNIT	>>	msdyn_issystemunit	
SYSTEMOFUNITS	>>	msdyn_systemofunits	
UNITSYMBOL	>>	name	
UNITDESCRIPTION	>>	msdyn_description	

Unit conversions to msdyn_unitofmeasureconversions

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
DENOMINATOR	=	msdyn_denominator	
NUMERATOR	=	msdyn_numerator	
FACTOR	=	msdyn_factor	
INNEROFFSET	=	msdyn_inneroffset	
OUTEROFFSET	=	msdyn_outeroffset	
ROUNDING	><	msdyn_rounding	
TOUNITSYMBOL	=	msdyn_tounit.msdyn_symbol	
FROMUNITSYMBOL	=	msdyn_fromunit.msdyn_symbol	

Product specific unit conversions to msdyn_productspecificunitofmeasureconversions

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
DENOMINATOR	=	msdyn_denominator	
NUMERATOR	=	msdyn_numerator	
FACTOR	=	msdyn_factor	
FROMUNITSYMBOL	=	msdyn_fromunit.msdyn_symbol	
TOUNITSYMBOL	=	msdyn_tounit.msdyn_symbol	
PRODUCTNUMBER	=	msdyn_globalproduct.msdyn_productnumber	
INNEROFFSET	=	msdyn_inneroffset	
OUTEROFFSET	=	msdyn_outeroffset	
ROUNDING	><	msdyn_rounding	

Initial synchronization of units data matching between Finance and

Operations and Dataverse

Initial synchronization of units

When dual write is enabled, units from Finance and Operations apps are synchronized to other Dynamics 365 apps. The unit groups synchronized from Finance and Operations apps in Dataverse have a flag set that indicates they are "Externally maintained".

Matching units and unit classes/groups data from Finance and Operations and other Dynamics 365 apps

First, it's important to note that the integration key for unit is msdyn_symbol. Therefore, this value must be unique in Dataverse or other Dynamics 365 apps. Because in other Dynamics 365 apps it is the pair "Unit group ID" and "Name" that define the uniqueness of a unit, you need to consider different scenarios for matching unit data between Finance and Operations apps and Dataverse.

For units matching/overlapping in Finance and Operations apps and other Dynamics 365 apps:

- **The unit belongs to a unit group in other Dynamics 365 apps that corresponds to the associated unit class in Finance and Operations apps**. In this case, the column msdyn_symbol in other Dynamics 365 apps must be filled in with the unit symbol from Finance and Operations apps. Therefore, when the data will be matched, and the unit group will be set as "Externally maintained" in other Dynamics 365 apps.
- **The unit belongs to a unit group in other Dynamics 365 apps that does not correspond to the associated unit class in Finance and Operations apps (no existing unit class in Finance and Operations apps for the unit class in other Dynamics 365 apps).** In this case, the msdyn_symbol must be filled in with a random string. Note that this value must be unique in other Dynamics 365 apps.

For units and unit classes in Finance and Operations not existing in other Dynamics 365 apps:

As part of dual-write the unit groups from Finance and Operations apps and its corresponding units are created and synchronized in other Dynamics 365 apps and Dataverse and the unit group will be set as "Externally maintained". No extra bootstrapping effort is required.

For units in other Dynamics 365 apps that do not exist in Finance and Operations apps:

The column msdyn_symbol must be filled in for all units. The units can always be created in Finance and Operations apps in the corresponding unit class (if it exists). If the unit class doesn't exist, first the unit class must be created (note that you cannot create a unit class in Finance and Operations apps except through extension if you are extending the enum) matching the other Dynamics 365 apps unit group. Then you can create the unit. Note that the unit symbol in Finance and Operations apps must be the msdyn_symbol previously specified in other Dynamics 365 apps for the unit.

Product policies: dimension, tracking and storage groups

The product policies are sets of policies used for defining products and its characteristics in inventory. The product dimension group, product tracking dimension group and storage dimension group can be found as product policies.

Product dimension groups to msdyn_productdimensiongroups

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
WILLSALESPRICESEARCHUSEPRODUCTSTYLE	><	msdyn_willsalespricesearchuseproductstyle	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
WILLPURCHASEPRICESEARCHUSEPRODUCTSIZE	> <	msdyn_willpurchasepricesearchuseproductsize	
WILLSALESPRICESEARCHUSEPRODUCTCONFIGURATION	> <	msdyn_willsalespricesearchuseprodconfig	
WILLSALESPRICESEARCHUSEPRODUCTCOLOR	> <	msdyn_willsalespricesearchuseproductcolor	
WILLPURCHASEPRICESEARCHUSEPRODUCTSTYLE	> <	msdyn_willpurchasepricesearchuseproductstyle	
WILLPURCHASEPRICESEARCHUSEPRODUCTCONFIGURATION	> <	msdyn_willpurchpricesearchuseprodconfig	
WILLPURCHASEPRICESEARCHUSEPRODUCTCOLOR	> <	msdyn_willpurchpricesearchuseproductcolor	
ISPRODUCTSTYLEACTIVE	> <	msdyn_isproductstyleactive	
ISPRODUCTSIZEACTIVE	> <	msdyn_isproductsizeactive	
ISPRODUCTCONFIGURATIONACTIVE	> <	msdyn_isproductconfigurationactive	
ISPRODUCTCOLORACTIVE	> <	msdyn_isproductcoloractive	
GROUPNAME	=	msdyn_groupname	
GROUPDESCRIPTION	=	msdyn_groupdescription	
PRODUCTVARIANTNOMENCLATURENAME	=	msdyn_productvariantnomenclaturename	
WILLSALESPRICESEARCHUSEPRODUCTSIZE	> <	msdyn_willsalespricesearchuseproductsize	

Tracking dimension groups to msdyn_producttrackingdimensiongroups

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
SERIALNUMBERCAPTURINGOPERATION	> <	msdyn_serialnumbercapturingoperation	
GROUPNAME	=	msdyn_groupname	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
GROUPDESCRIPTION	=	msdyn_groupdescription	
ISSERIALNUMBERENABLEDFORPRODUCTIONCONSUMPTIONPROCESS	> <	msdyn_issnenabledforpcprocess	
ISSERIALNUMBERCONTROLENABLED	> <	msdyn_isserialnumbercontrolenabled	
ISSERIALNUMBERENABLEDFORSALESPROCESS	> <	msdyn_isserialnumberenabledforsalesprocess	
ISSERIALNUMBERACTIVE	> <	msdyn_isserialnumberactive	
ISSALESPRICEBYSERIALNUMBER	> <	msdyn_issalespricebyserialnumber	
ISSALESPRICEBYBATCHNUMBER	> <	msdyn_issalespricebybatchnumber	
ISPURCHASEPRICEBYSERIALNUMBER	> <	msdyn_ispurchasepricebyserialnumber	
ISPURCHASEPRICEBYBATCHNUMBER	> <	msdyn_ispurchasepricebybatchnumber	
ISPRIMARYSTOCKINGENABLEDFORSERIALNUMBER	> <	msdyn_isprimarystockingenabledforsn	
ISPRIMARYSTOCKINGENABLEDFORBATCHNUMBER	> <	msdyn_isprimarystockingenabledforbn	
ISPHYSICALINVENTORYENABLEDFORSERIALNUMBER	> <	msdyn_isphysicalinventoryenabledforsn	
ISPHYSICALINVENTORYENABLEDFORBATCHNUMBER	> <	msdyn_isphysicalinventoryenabledforbn	
ISFINANCIALINVENTORYENABLEDFORSERIALNUMBER	> <	msdyn_isfinancialinventoryenabledforsn	
ISFINANCIALINVENTORYENABLEDFORBATCHNUMBER	> <	msdyn_isfinancialinventoryenabledforbn	
ISCOVERAGEPLANENABLEDFORSERIALNUMBER	> <	msdyn_iscoverageplanenabledforserialnumber	
ISCOVERAGEPLANENABLEDFORBATCHNUMBER	> <	msdyn_iscoverageplanenabledforbatchnumber	
ISBLANKRECEIPTALLOWEDFORSERIALNUMBER	> <	msdyn_isblankreceiptallowedforserialnumber	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
ISBLANKRECEIPTALLOWEDFORBATCHNUMBER	> <	msdyn_isblankreceiptallowedforbatchnumber	
ISBLANKISSUEALLOWEDFORSERIALNUMBER	> <	msdyn_isblankissueallowedforserialnumber	
ISBLANKISSUEALLOWEDFORBATCHNUMBER	> <	msdyn_isblankissueallowedforbatchnumber	
ISBATCHNUMBERACTIVE	> <	msdyn_isbatchnumberactive	
ISINVENTORYOWNERACTIVE	> <	msdyn_isinventoryowneractive	

Storage dimension groups to msdyn_productstoragedimensiongroups

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
WILLSALESPRICESEARCHUSEWAREHOUSE	> <	msdyn_willsalespricesearchusewarehouse	
WILLSALESPRICESEARCHUSESITE	> <	msdyn_willsalespricesearchusesite	
WILLSALESPRICESEARCHUSEINVENTORYSTATUS	> <	msdyn_willsalespricesearchuseinventorystatus	
WILLPURCHASEPRICESEARCHUSEWAREHOUSE	> <	msdyn_willpurchasepricesearchusewarehouse	
WILLPURCHASEPRICESEARCHUSESITE	> <	msdyn_willpurchasepricesearchusesite	
WILLPURCHASEPRICESEARCHUSEINVENTORYSTATUS	> <	msdyn_willpurchpricesearchuseinventstatus	
WILLCOVERAGEPLANNINGUSEWAREHOUSE	> <	msdyn_willcoverageplanusewarehouse	
WILLCOVERAGEPLANNINGUSELOCATION	> <	msdyn_iscoverageplanenabledforlocation	
WILLCOVERAGEPLANNINGUSEINVENTORYSTATUS	> <	msdyn_willcoverageplanuseinventorystatus	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
AREADVANCEDWAREHOUSEMANAGEMENTPROCESSESENABLED	> <	msdyn_areadvancedwmprocessesenabled	
ISWAREHOUSEPRIMARYSTORAGEDIMENSION	> <	msdyn_iswarehouseprimarystoragedimension	
ISWAREHOUSEMANDATORY	> <	msdyn_iswarehousemandatory	
ISPHYSICALINVENTORYENABLEDFORWAREHOUSE	> <	msdyn_isphysicalinventoryenabledforwarehouse	
ISPHYSICALINVENTORYENABLEDFORLOCATION	> <	msdyn_isphysicalinventoryenabledforlocation	
ISLOCATIONACTIVE	> <	msdyn_islocationactive	
ISFINANCIALINVENTORYENABLEDFORWAREHOUSE	> <	msdyn_isfinancialinventoryenabledforwarehouse	
GROUPNAME	=	msdyn_groupname	
GROUPDESCRIPTION	=	msdyn_groupdescription	
ISBLANKRECEIPTALLOWEDFORLOCATION	> <	msdyn_isblankreceiptallowedforlocation	
ISBLANKISSUEALLOWEDFORLOCATION	> <	msdyn_isblankissueallowedforlocation	

Product hierarchies

Product category hierarchies to msdyn_productcategoryhierarchies

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
HIERARCHYNAME	=	msdyn_name	
HIERARCHYDESCRIPTION	=	msdyn_description	

Product categories to msdyn_productcategories

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
PRODUCTCATEGORYHIERARCHYNAME	=	msdyn_hierarchy.msdyn_name	
ISCATEGORYINHERITINGPARENTPRODUCTATTRIBUTES	><	msdyn_isinheritingparentproductattributes	
PROJECTCATEGORYNAME	=	msdyn_projectcategoryname	
ISTANGIBLEPRODUCT	><	msdyn_istangibleproduct	
ISCATEGORYINHERITINGPARENTCATEGORYATTRIBUTES	><	msdyn_isinheritingparentcategoryattributes	
CATEGORYCODE	=	msdyn_code	
CATEGORYDESCRIPTION	=	msdyn_description	
CATEGORYKEYWORDS	=	msdyn_keywords	
CATEGORYNAME	=	msdyn_name	
FRIENDLYCATEGORYNAME	=	msdyn_friendlycategoryname	
PARENTPRODUCTCATEGORYNAME	=	msdyn_parentproductcategory.msdyn_name	
PRODUCTCATEGORYHIERARCHYNAME	>>	msdyn_parentproductcategory.msdyn_hierarchy.msdyn_name	

Product category assignments to msdyn_productcategoryassignments

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
PRODUCTNUMBER	=	msdyn_globalproduct.msdyn_productnumber	
PRODUCTCATEGORYNAME	=	msdyn_productcategory.msdyn_name	
PRODUCTCATEGORYHIERARCHYNAME	=	msdyn_productcategory.msdyn_hierarchy.msdyn_name	
PRODUCTNUMBER	>>	msdyn_name	

Product category hierarchy roles to msdyn_productcategoryhierarchyroles

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
PRODUCTCATEGORYHIERARCHYNAME	=	msdyn_hierarchy.msdyn_name	
HIERARCHYROLE	><	msdyn_hierarchyrole	

Integration key for products

To uniquely identify products between Dynamics 365 for Finance and Operations and products in Dataverse the integration keys are used. For products, the **(productnumber)** is the unique key that identifies a product in Dataverse. It's composed by the concatenation of: **(company, msdyn_productnumber)**. The **company** indicates the legal entity in Finance and Operations and **msdyn_productnumber** indicates the product number for the specific product in Finance and Operations.

For users of other Dynamics 365 apps, the product is identified in the UI with the **msdyn_productnumber** (note that the label of the column is **Product number**). In the product form both the company and the msydn_productnumber are shown. However, the (productnumber) column, the unique key for a product, is not shown.

If you build apps on Dataverse, you should pay attention to using the **productnumber** (the unique product ID) as the integration key. Do not use **msdyn_productnumber**, because it's not unique.

Initial synchronization of products and migration of data from Dataverse to Finance and Operations

Initial synchronization of products

When dual-write is enabled, products from Finance and Operations apps are synchronized to Dataverse and customer engagement apps. Products created in Dataverse and other Dynamics 365 apps before dual-write was released will not be updated or matched with product data from Finance and Operations apps.

Matching product data from Finance and Operations and other Dynamics 365 apps

If the same products are kept (overlapping/matching) in Finance and Operations and in Dataverse and other Dynamics 365 apps, when enabling dual-write the synchronization of products from Finance and Operations will take place, and duplicate rows will appear in Dataverse for the same product. To avoid the previous situation, if other Dynamics 365 apps have products that are overlapping/matching with Finance and Operations, then the administrator enabling dual write must bootstrap the columns **Company** (example: "USMF") and **msdyn_productnumber** (example: "1234:Black:S") before the synchronization of products takes place. In other words, these two columns in the product in Dataverse must be filled in with the respective company in Finance and Operations to which the product needs to be matched with and with its product number.

Then, when the synchronization is enabled and takes place, the products from Finance and Operations will be synchronized with the matched products in Dataverse and other Dynamics 365 apps. This is applicable for both distinct products and product variants.

Migration of product data from other Dynamics 365 apps to Finance and Operations

If other Dynamics 365 apps have products that aren't present in Finance and Operations, the administrator can first use the **EcoResReleasedProductCreationV2Entity** for importing those products in Finance and

Operations. And secondly, match the product data from Finance and Operations and other Dynamics 365 apps as described above.

Integrated sites and warehouses

2/18/2021 • 3 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

This topic describes the integration of site and warehouse data between Finance and Operations and Dataverse. Operational sites and warehouses are common concepts in a Supply Chain Management application. They are used to model the supply chain of your company.

Templates

With the integration with Dataverse, these concepts and all their related information are available in Dataverse using the sites and warehouses data tables in the following table.

FINANCE AND OPERATIONS APPS	OTHER DYNAMICS 365 APPS	DESCRIPTION
Sites	msdyn_operationalsites	
Warehouses	msdyn_warehouses	

Mapping tables

Mapping types

There are several different mapping types. The following table explains the symbols used in the template tables.

SYMBOL	DESCRIPTION
>	One-way
>>	One-way, and data is transformed in the process.
=	Bidirectional
><	Bidirectional, and data is transformed in the process.
<<	One-way, and data is transformed in the process.

Filters

The source filter and reverse source filter determine which rows are synchronized.

Default values

If a synchronized field does not exist in either the Finance and Operations table or the other Dynamics 365 table, then a default value is assigned in the synchronized table. In some cases, the default value is an integer that is a lookup to an attribute value in the Common Data Model. For example, in the Contacts table of the Common Data Model, the default value of **address1_addresstypecode** is **3**. In the Common Data Model, for address1AddressTypeCode the value of **3** is **Primary address**.

Sites to msdyn_operationalsites

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
DEFAULTFINANCIALDIMENSIONVALUE	> <	msdyn_defaultfinancialdimensionvalue	
DEFAULTINVENTORYSTATUSID	> <	msdyn_defaultinventorystatusid	
ISRECEIVINGWAREHOUSEOVERRIDEALLOWED	> <	msdyn_isreceivingwarehouseoverrideallowed	
SITEID	> <	msdyn_siteid	
SITENAME	> <	msdyn_sitename	
TAXBRANCHCODE	> <	msdyn_taxbranchcode	
FISCALESTABLISHMENTID	> <	msdyn_fiscalestablishmentid	
ISPRIMARYADDRESSASSIGNED	> <	msdyn_isprimaryaddressassigned	
PRIMARYADDRESSCITY	> <	msdyn_primaryaddresscity	
PRIMARYADDRESSCOUNTRYREGIONID	> <	msdyn_primaryaddresscountryregionid	
PRIMARYADDRESSCOUNTYID	> <	msdyn_primaryaddresscountyid	
PRIMARYADDRESSDISTRICTNAME	> <	msdyn_primaryaddressdistrictname	
PRIMARYADDRESSLATITUDE	> <	msdyn_primaryaddresslatitude	
PRIMARYADDRESSLOCATIONROLES	> <	msdyn_primaryaddresslocationrole	
PRIMARYADDRESSLOCATIONSALESTAXGROUPCODE	> <	msdyn_primaryaddresslocationsalestaxgroupcode	
PRIMARYADDRESSLONGITUDE	> <	msdyn_primaryaddresslongitude	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
PRIMARYADDRESSSTATEID	> <	msdyn_primaryaddressstateid	
PRIMARYADDRESSSTREET	> <	msdyn_primaryaddressstreet	
PRIMARYADDRESSZIPCODE	> <	msdyn_primaryaddresszipcode	
PRIMARYADDRESSBUILDINGCOMPLIMENT	> <	msdyn_primaryaddressbuildingcompliment	
PRIMARYADDRESSCITYINKANA	> <	msdyn_primaryaddresscityinkana	
PRIMARYADDRESSSTREETINKANA	> <	msdyn_primaryaddressstreetinkana	
PRIMARYADDRESSDESCRIPTION	> <	msdyn_primaryaddressdescription	
FORMATTEDPRIMARYADDRESS	> <	msdyn_formattedprimaryaddress	
WILLMASTERPLANNEDINTRASITEMOVEMENTSUSETRANSFERJOURNALS	> <	msdyn_masterplannedusestransferjournal	
PRIMARYADDRESSPOSTBOX	> <	msdyn_primaryaddresspostbox	
PRIMARYADDRESSSTREETNUMBER	> <	msdyn_primaryaddressstreetnumber	

Warehouses to msdyn_warehouses

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
DEFAULTCONTAINERTYPEID	> <	msdyn_defaultcontainertypeid	
AREITEMSCOVERAGEPLANNEDMANUALLY	> <	msdyn_areitemscoverageplannedmanually	
ARELABORSTANDARDSALLOWED	> <	msdyn_arelaborstandardsallowed	
PRIMARYADDRESSBUILDINGCOMPLIMENT	> <	msdyn_primaryaddressbuildingcompliment	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
PRIMARYADDRESSPOSTBOX	> <	msdyn_primaryaddresspostbox	
PRIMARYADDRESSSTREETNUMBER	> <	msdyn_primaryaddressstreetnumber	
AREWAREHOUSELOCATIONCHECKDIGITSUNIQUE	> <	msdyn_arewarehouselocationcheckdigitsunique	
INVENTORYCOUNTINGREASONCODEPOLICYNAME	> <	msdyn_inventorycountingreasoncodepolicyname	
AUTOUPDATESHIPMENTRULE	> <	msdyn_autoupdateshipmentrule	
WAREHOUSERELEASERESERVATIONREQUIREMENTRULE	> <	msdyn_warehousereleasereservationrequirement	
EXTERNALLYLOCATEDWAREHOUSEVENDORACCOUNTNUMBER	> <	msdyn_externallylocatedwarehousevendoraccountnu	
INVENTORYSTATUSCHANGERESERVATIONREMOVALLEVEL	> <	msdyn_inventorystatuschangereservationremoval	
WAREHOUSEWORKPROCESSINGPOLICYNAME	> <	msdyn_warehouseworkprocessingpolicyname	
ISFALLBACKWAREHOUSE	> <	msdyn_isfallbackwarehouse	
ISFINANCIALNEGATIVERETAILSTOREINVENTORYALLOWED	> <	msdyn_financialnegativestoreinventoryallowed	
ISPALLETMOVEMENTDURINGCYCLECOUNTINGALLOWED	> <	msdyn_palletmovementduringcyclecountingallowed	
ISPHYSICALNEGATIVERETAILSTOREINVENTORYALLOWED	> <	msdyn_physicalnegativestoreinventoryallowed	
ISREFILLEDFROMMAINWAREHOUSE	> <	msdyn_isrefilledfrommainwarehouse	
ISRETAILSTOREWAREHOUSE	> <	msdyn_isretailstorewarehouse	
MAINREFILLINGWAREHOUSEID	> <	msdyn_mainrefillingwarehouse.msdyn_warehouseidentifier	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
MASTERPLANNINGWORKCALENDARDID	> <	msdyn_masterplanningworkcalendarid	
MAXIMUMBATCHPICKINGLISTQUANTITY	> <	msdyn_maximumbatchpickinglistquantity	
MAXIMUMPICKINGLISTLINEQUANTITY	> <	msdyn_maximumpickinglistlinequantity	
OPERATIONALSITEID	> <	msdyn_operationalsite.msdyn_siteid	
QUARANTINEWAREHOUSEID	> <	msdyn_quarantinewarehouse.msdyn_warehouseidentifier	
RETAILSTOREQUANTITYALLOCATIONREPLENISMENTRULEWEIGHT	> <	msdyn_storeqtyallocationreplenishmentweight	
SHOULDWAREHOUSELOCATIONIDINCLUDEAISLEID	> <	msdyn_shouldwarehouselocationincludeaisleid	
TRANSITWAREHOUSEID	> <	msdyn_transitwarehouse.msdyn_warehouseidentifier	
WAREHOUSEID	> <	msdyn_warehouseidentifier	
WAREHOUSEID	> >	msdyn_name	
WAREHOUSELOCATIONIDBINIDFORMAT	> <	msdyn_warehouselocationidbinidformat	
WAREHOUSELOCATIONIDRACKIDFORMAT	> <	msdyn_warehouselocationidrackidformat	
WAREHOUSELOCATIONIDSHELFIDFORMAT	> <	msdyn_warehouselocationidshelfidformat	
WAREHOUSENAME	> <	msdyn_description	
WAREHOUSESPECIFICDEFAULTINVENTORYSTATUSID	> <	msdyn_warehousespecificdefaultinventorystatusid	
WAREHOUSETYPE	> <	msdyn_warehousetype	
ISPRIMARYADDRESSASSIGNED	> <	msdyn_isprimaryaddressassigned	
PRIMARYADDRESSCITY	> <	msdyn_primaryaddresscity	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
PRIMARYADDRESSCOUNTRYREGIONID	> <	msdyn_primaryaddresscountryregionid	
PRIMARYADDRESSCOUNTYID	> <	msdyn_primaryaddresscountyid	
PRIMARYADDRESSDISTRICTNAME	> <	msdyn_primaryaddressdistrictname	
PRIMARYADDRESSLATITUDE	> <	msdyn_primaryaddresslatitude	
PRIMARYADDRESSLONGITUDE	> <	msdyn_primaryaddresslongitude	
PRIMARYADDRESSLOCATIONROLES	> <	msdyn_primaryaddresslocationroles	
PRIMARYADDRESSLOCATIONSALESTAXGROUPCODE	> <	msdyn_primaryaddresslocationsalestaxgroupcode	
PRIMARYADDRESSSTATEID	> <	msdyn_primaryaddressstateid	
PRIMARYADDRESSSTREET	> <	msdyn_primaryaddressstreet	
PRIMARYADDRESSZIPCODE	> <	msdyn_primaryaddresszipcode	
EXTERNALLYLOCATEDWAREHOUSECUSTOMERACCOUNTNUMBER	> <	msdyn_externallylocatedwarehousecustomeraccount	
PRIMARYADDRESSCITYINKANA	> <	msdyn_primaryaddresscityinkana	
PRIMARYADDRESSSTREETINKANA	> <	msdyn_primaryaddressstreetinkana	
PRIMARYADDRESSDESCRIPTION	> <	msdyn_primaryaddressdescription	
AREADVANCEDWAREHOUSEMANAGEMENTPROCESSESENABLED	> <	msdyn_useadvancedwarehousemanagementprocesses	
AREPICKINGLISTSDELIVERYMODESPECIFIC	> <	msdyn_arepickinglistsdeliverymodespecific	
AREPICKINGLISTSSHIPMENTSPECIFICONLY	> <	msdyn_arepickinglistshipmentspecificonly	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
FORMATTEDPRIMARYADDRESS	> <	msdyn_formattedprimaryaddress	
ISBILLOFLADINGPRINTINGBEFORESHIPMENTCONFIRMATIONENABLED	> <	msdyn_printbillofladingbeforeshipconfirmation	
RAWMATERIALPICKINGINVENTORYISSUESTATUS	> <	msdyn_rawmaterialpickinginventoryissuestatus	
WILLAUTOMATICLOADRELEASERESERVEINVENTORY	> <	msdyn_willautomaticloadreleaseinventory	
WILLINVENTORYSTATUSCHANGEREMOVEBLOCKING	> <	msdyn_willinventorystatuschangeremoveblocking	
WILLMANUALLOADRELEASERESERVEINVENTORY	> <	msdyn_willmanualloadreleasereserveinventory	
WILLORDERRELEASINGCONSOLIDATESHIPMENTS	> <	msdyn_willorderreleasingconsolidateshipments	
WILLPRODUCTIONBOMSRESERVEWAREHOUSELEVELONLY	> <	msdyn_productionbomsreservewarehouselevel	
WILLSHIPPINGCANCELLATIONDECREMENTLOADQUANITY	> <	msdyn_shippingcanceldecrementloadquantity	
WILLWAREHOUSELOCATIONIDINCLUDEBINIDBYDEFAULT	> <	msdyn_warehouselocationidincludeblindid	
WILLWAREHOUSELOCATIONIDINCLUDERACKIDBYDEFAULT	> <	msdyn_warehouselocationincluderackidbydefault	
WILLWAREHOUSELOCATIONIDINCLUDESHELFIDBYDEFAULT	> <	msdyn_warehouselocationidincludeshelfid	

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Company concept in Dataverse

2/18/2021 • 5 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

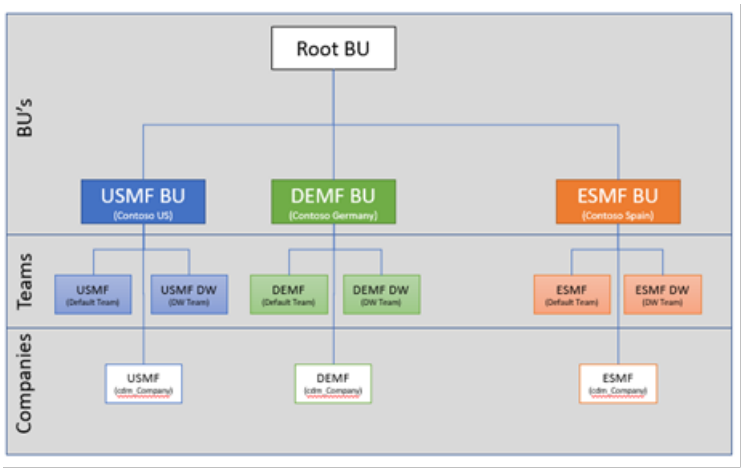
In Finance and Operations, the concept of a *company* is both a legal construct and a business construct. It's also a security and visibility boundary for data. Users always work in the context of a single company, and most of the data is striped by company.

Dataverse doesn't have an equivalent concept. The closest concept is *business unit*, which is primarily a security and visibility boundary for user data. This concept doesn't have the same legal or business implications as the company concept.

Because business unit and company aren't equivalent concepts, it isn't possible to force a one-to-one (1:1) mapping between them for the purpose of Dataverse integration. However, because users must, by default, be able to see the same rows in the application and Dataverse, Microsoft has introduced a new table in Dataverse that is named cdm_Company. This table is equivalent to the Company table in the application. To help guarantee that visibility of rows is equivalent between the application and Dataverse out of the box, we recommend the following setup for data in Dataverse:

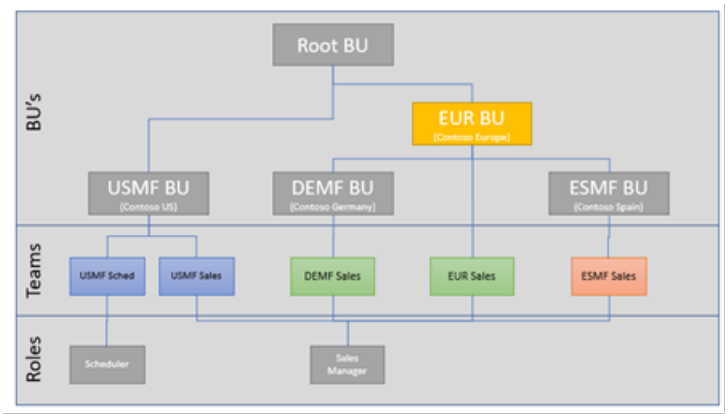
- For each Finance and Operations Company row that is enabled for dual-write, an associated cdm_Company row is created.
- When a cdm_Company row is created and enabled for dual-write, a default business unit is created that has the same name. Although a default team is automatically created for that business unit, the business unit isn't used.
- A separate owner team is created that has the same name. It's also associated with the business unit.
- By default, the owner of any row that is created and dual-written to Dataverse is set to the "DW Owner" team that is linked to the associated business unit.

The following illustration shows an example of this data setup in Dataverse.



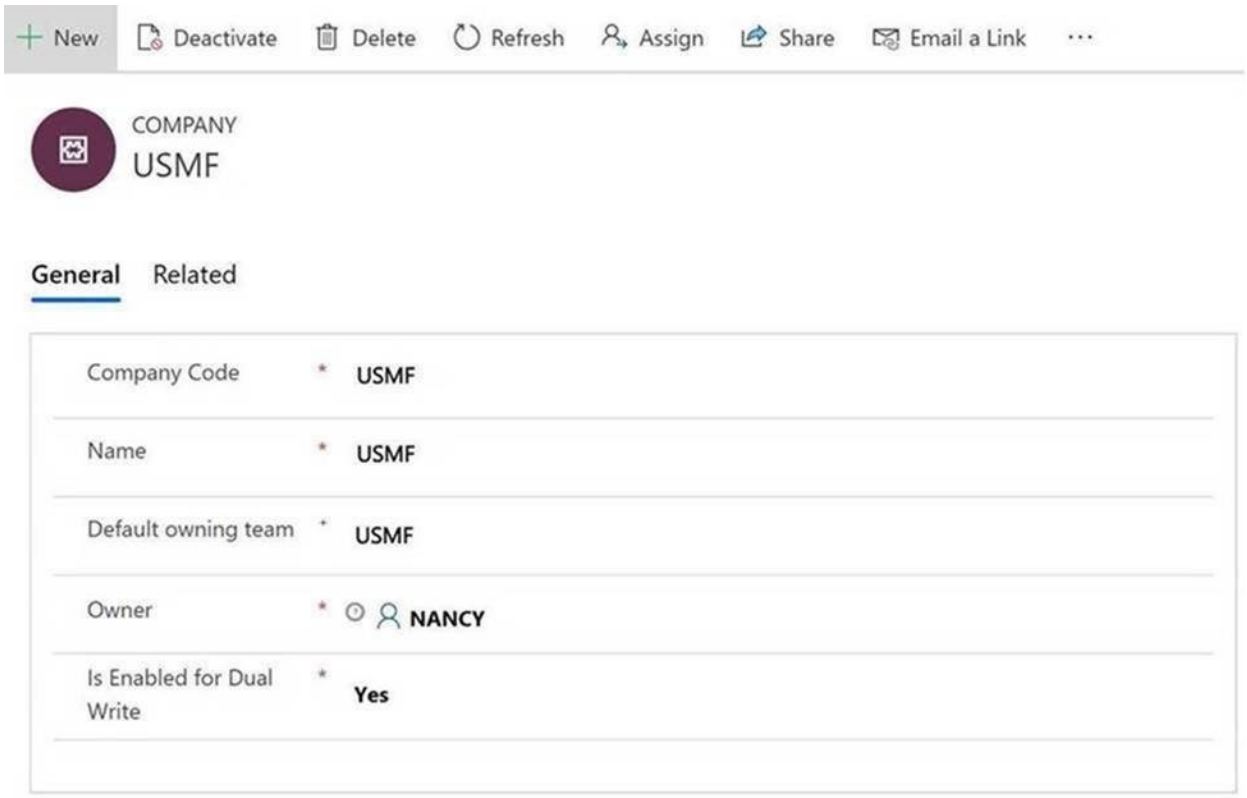
Because of this configuration, any row that is related to the USMF company will be owned by a team that is linked to the USMF business unit in Dataverse. Therefore, any user who has access to that business unit through a security role that is set to business unit–level visibility can now see those rows. The following example shows how teams can be used to provide the correct access to those rows.

- The "Sales Manager" role is assigned to members of the "USMF Sales" team.
- Users who have the "Sales Manager" role can access any account rows that are members of the same business unit that they are members of.
- The "USMF Sales" team is linked to the USMF business unit that was mentioned earlier.
- Therefore, members of the "USMF Sales" team can see any account that is owned by the "USMF DW" user, which would have come from the USMF Company table in Finance and Operations.



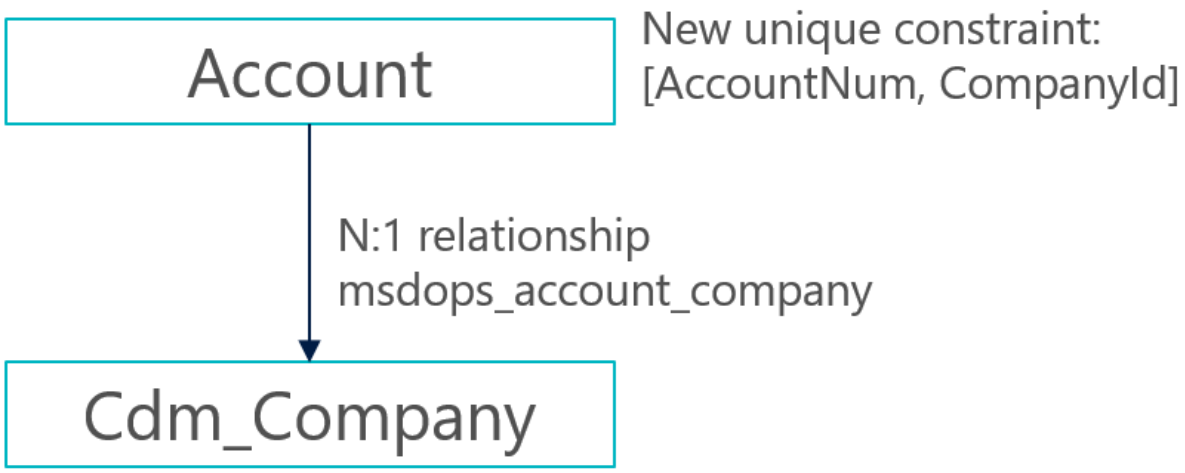
As the preceding illustration shows, this 1:1 mapping between business unit, company, and team is just a starting point. In this example, a new "Europe" business unit is manually created in Dataverse as the parent for both DEMF and ESMF. This new root business unit is unrelated to dual-write. However, it can be used to give members of the "EUR Sales" team access to account data in both DEMF and ESMF by setting the data visibility to **Parent/Child BU** in the associated security role.

A final topic to discuss is how dual-write determines which owner team it should assign rows to. This behavior is controlled by the **Default owning team** column on the cdm_Company row. When a cdm_Company row is enabled for dual-write, a plug-in automatically creates the associated business unit and owner team (if it doesn't already exist), and sets the **Default owning team** column. The admin can change this column to a different value. However, the admin can't clear the column as long as the table is enabled for dual-write.



Company striping and bootstrapping

Dataverse integration brings company parity by using a company identifier to stripe data. As the following illustration shows, all company-specific tables are extended so that they have a many-to-one (N:1) relationship with the cdm_Company table.

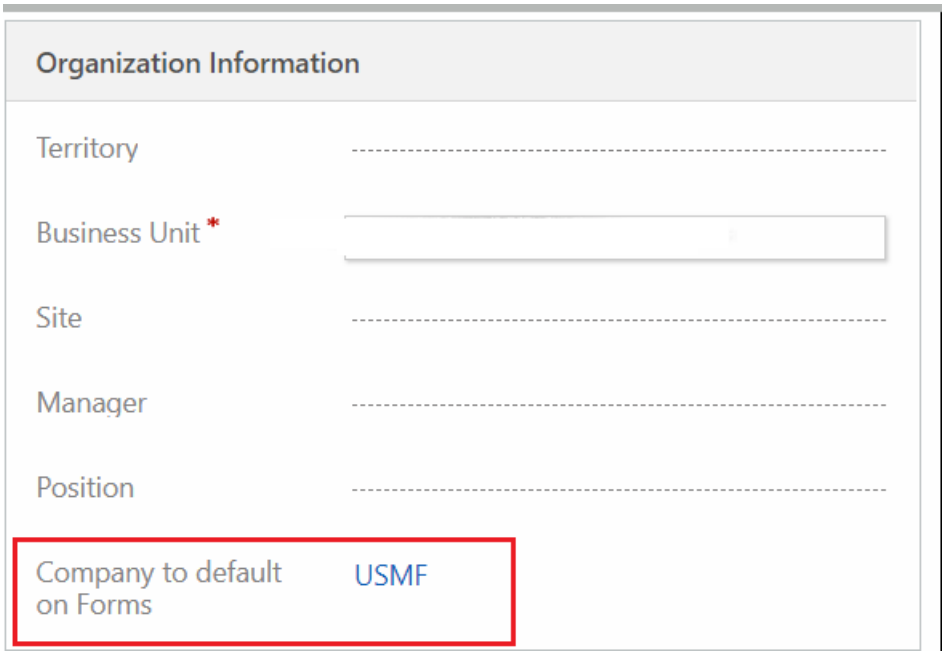


- For rows, after a company is added and saved, the value becomes read-only. Therefore, users should make sure that they select the correct company.
- Only rows that have company data are eligible for dual-write between the application and Dataverse.
- For existing Dataverse data, an admin-led bootstrapping experience will soon be available.

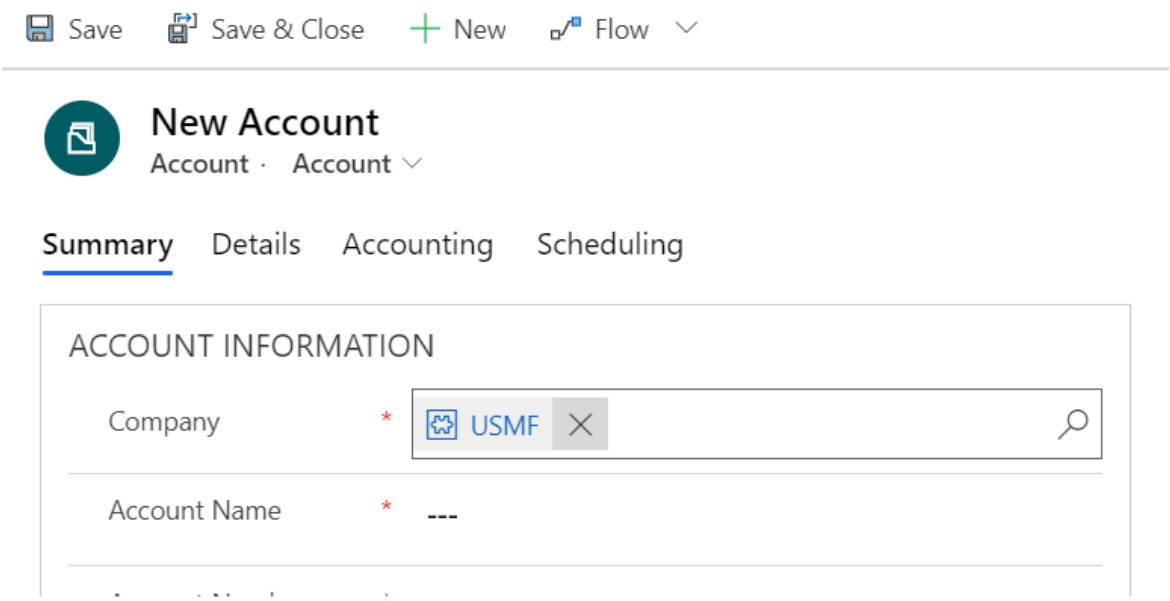
Autopopulate company name in customer engagement apps

There are several ways to auto-populate the company name in customer engagement apps.

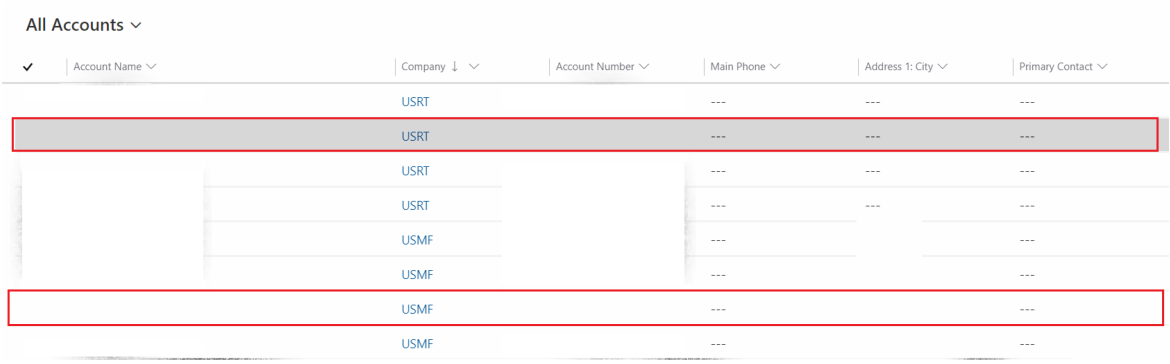
- If you are a system administrator, you can set the default company by navigating to **Advanced Settings** > **System** > **Security** > **Users**. Open the **User** form, and in the **Organization Information** section, set the **Company to default on Forms** value.



- If you have **Write** access to the **SystemUser** table for the **Business Unit** level, then you can change the default company on any form by selecting a company from the **Company** drop-down menu.



- If you have **Write** access to data in more than one company, then you can change the default company by choosing a row that belongs to different company.

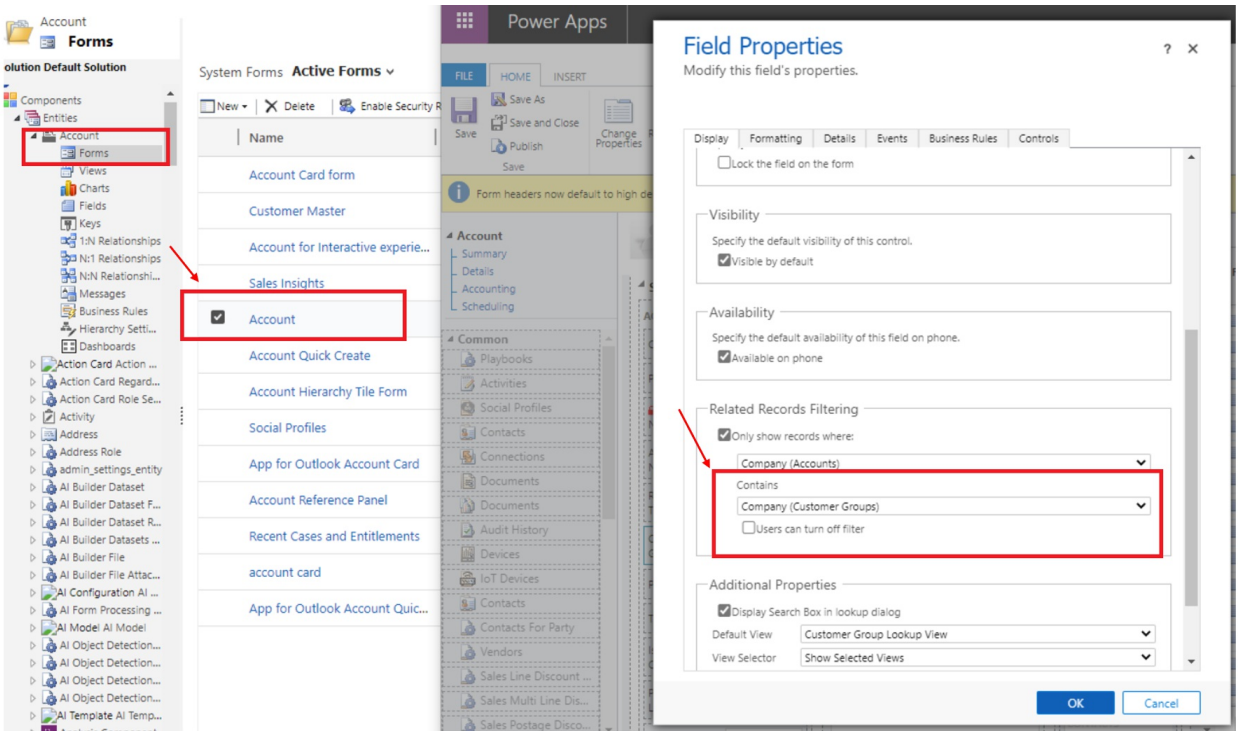


- If you are a system configurator or administrator, and you want to auto-populate company data on a custom form, then you can use form events. Add a JavaScript reference to **msdyn_/DefaultCompany.js** and use the following events. You can use any out-of-the-box form, for example, the **Account** form.
    - **OnLoad** event for the form: Set the **defaultCompany** column.

- **OnChange** event for the **Company** column: Set the **updateDefaultCompany** column.

Apply filtering based on the company context

To apply filtering based on the company context on your custom forms or on custom lookup columns added to the standard forms, open the form and use the **Related Records Filtering** section to apply the company filter. You must set this for each lookup column that requires filtering based on the underlying company on a given row. The setting is shown for **Account** in the following illustration.



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Initialize company data

2/18/2021 • 4 minutes to read • Edit Online

IMPORTANT

Some or all of the functionality noted in this topic is available as part of a preview release. The content and the functionality are subject to change. For more information about preview releases, see Service update availability.

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

If have an existing Microsoft Dataverse instance or Finance and Operations app instance that has business data, you might want to enable a dual-write connection against it. In this case, you must initialize the Dataverse data or Finance and Operations app data with company information before you enable dual-write. This initialization process is sometimes referred to as *bootstrapping*.

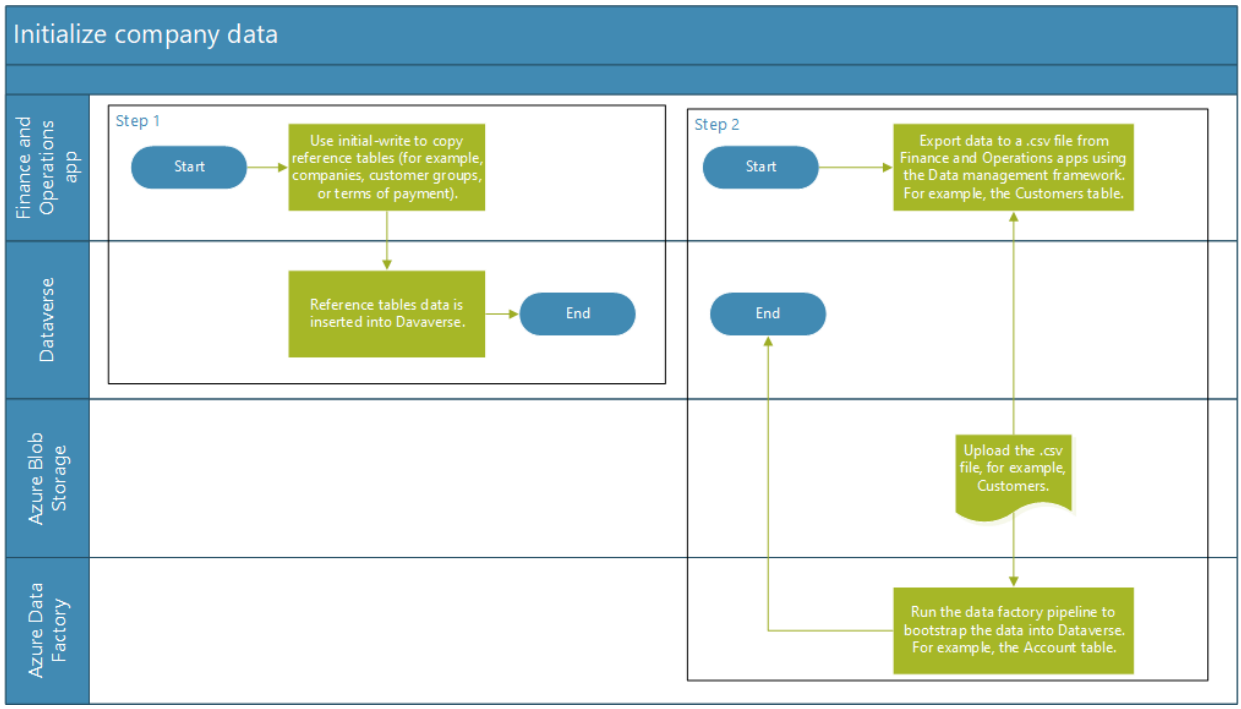
This topic includes sample scenarios that explain how to use Azure Data Factory to initialize data in Dataverse tables for dual-write. It doesn't cover all tables, error handling scenarios, or lookups. Use this topic and template as a reference to set up your own Azure Data Factory pipeline to import data into Dataverse or update data in Dataverse.

High-level scenario

Consider the **Customers** table in a Finance and Operations app, and the **Account** table in Dataverse.

- Use initial write to copy reference and dependent tables, such as **Company**, **Customer groups**, and **Terms of payment**, from the Finance and Operations app to Dataverse.
- Use the Data management framework to export data from the Finance and Operations app in comma-separated values (CSV) format. For example, set up an export project in Data management to export customers from each company by using the **DataAreaId** field in the Finance and Operations app. This process is a one-time manual process.
- Use Azure Blob Storage to store the CSV files for lookup and transformation. Upload the CSV file for your Finance and Operations customers into Azure Blob Storage.
- Use Azure Data Factory to initialize data in Dataverse.

The following illustration shows the workflow.



This scenario is based on the following assumptions:

- The source data is in the Finance and Operations app.
- If an account exists in Dataverse, but it doesn't exist in the Finance and Operations app, it won't be initialized as part of this flow.
- All account records in the customer engagement apps have a natural key (account number) that matches the Finance and Operations natural key (**CustomerAccount**).
- Rows have a one-to-one (1:1) mapping across the apps.

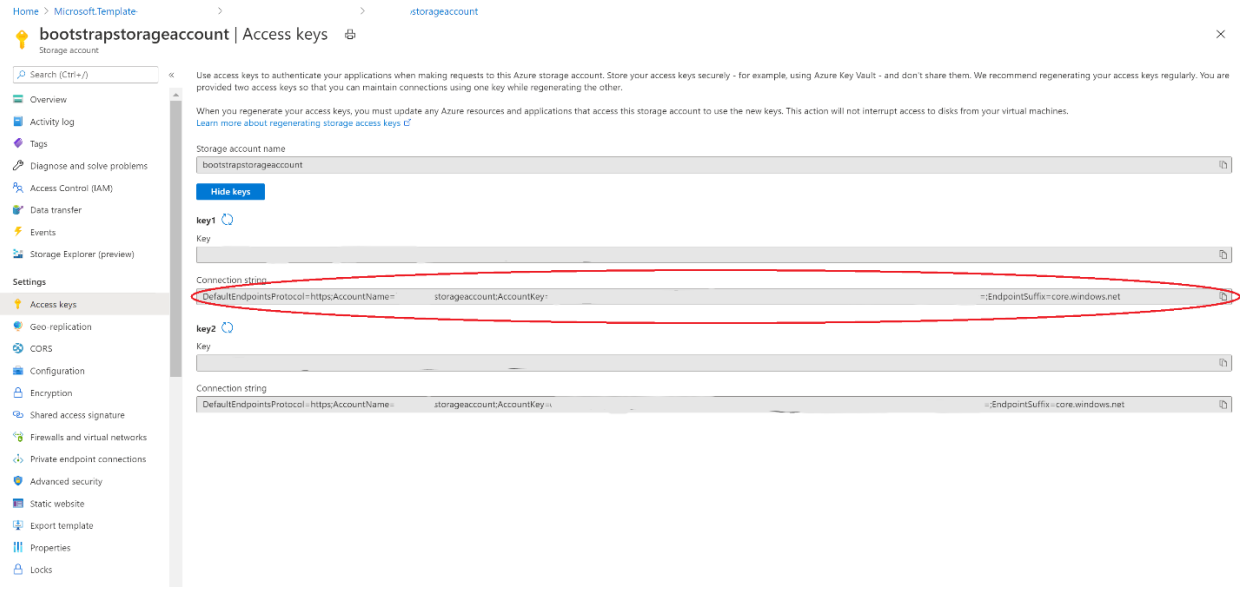
Prerequisites

- **Azure subscription** – You have **contributor access** to an existing Azure subscription. If you don't have an Azure subscription, create a free Azure account before you begin.
- **Azure storage account** – You have an Azure storage account. If you don't have a storage account, follow the steps in Create an Azure storage account to create one.
- **Azure data factory** – Create an Azure Data Factory resource by following the steps in Create a data factory.
- **Finance and Operations app** – Use the Data management framework to export the data in CSV format. For more information, see Data management overview. In this template, customers are exported by using the **CustCustomerV3Entity** table.
- **Dynamics 365 Dataverse** – Use the credentials for the Dataverse admin user to initialize the data.
- **Dual-write** – Dual-write solutions are installed, and reference data is copied by using initial write.

Deployment steps

Set up an Azure storage account

If you don't have an Azure storage account, follow these steps in Create an Azure storage account to create one. In your storage account, create one container that is named **ce-data**. This container will store all data files. You can change the container in your datasets and pipelines as you require. Go to **Access keys**, and copy the **Connection string** value, as shown in the following illustration. This value is required when you import the Azure Data Factory template.



Deploy an Azure Data Factory template

1. Make a note of the name of the Azure data factory that you created.
2. Make a note of the connection string for the Azure storage account.
3. Make a note of the service URI of the Dataverse instance, and the admin user name and password.

   The following table shows the parameters that are required.

PARAMETER NAME	DESCRIPTION	EXAMPLE VALUE
Factory name	The name of your data factory	*BootstrapDataverseDataADF*
Bootstrap blob storage account Linked Service_connection String	The connection string for blob storage	The value that you copied when you created the storage account
Bootstrap Dynamics 365 Linked Service_service Uri	The URI of the Dataverse instance	`https://contosod365.crm4.dynamics.com`
Bootstrap Dynamics 365 Linked Service_properties_type Properties_username	The Dynamics 365 admin user's user ID	`<adminservice@contoso.onmicrosoft.com>`
Bootstrap Dynamics 365 Linked Service_password	The Dynamics 365 admin user's password	********

4. Download the Azure Resource Manager (ARM) template file to your local directory.
5. In the Azure portal, go to Custom deployment.
6. Select **Build your own template in the editor**.
7. Select **Load file**, and find and select the ARM template file that you downloaded earlier. Then select **Save**.
8. Provide the required parameters, select **Review**, and then select **Create**.

Home >

Custom deployment

Deploy from a custom template

Select a template | Basics | Review + create

Template

Customized template
13 resources

Edit template | Edit parameters

Deployment scope

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ

Resource group * ⓘ

Create new

Parameters

Region ⓘ — Central US

Factory Name

Bootstrap blob stroage account Linked Service_connection String *

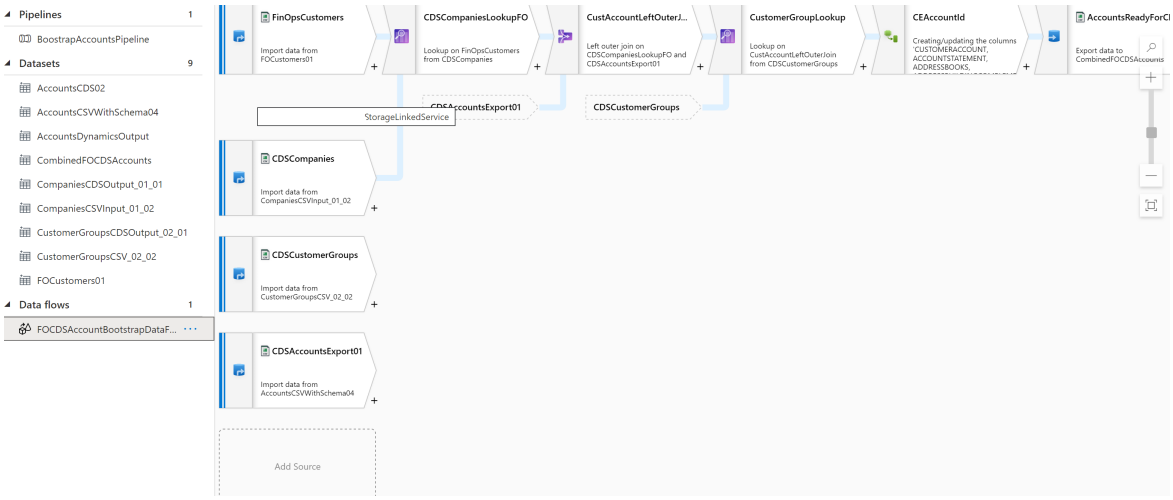
Bootstrap Dynamics 365 Linked Service_service Uri *

Bootstrap Dynamics 365 Linked Service_properties_type Properties_username

Bootstrap Dynamics 365 Linked Service_password *

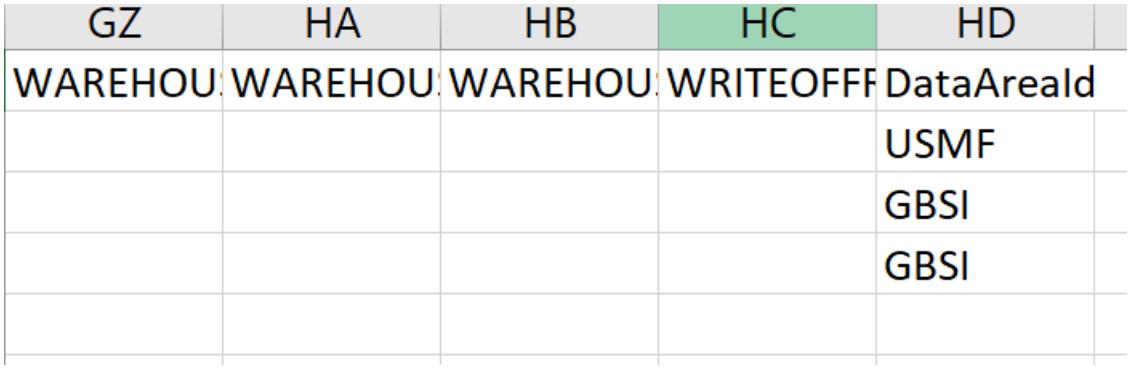
Review + create | < Previous | Next : Review + create >

9. After deployment, you will see **Pipelines**, **Datasets**, and **Data flows** sections in the list pane.



Run the process

1. In the Finance and Operations app, use the Data management framework to export data in CSV format. For more information, see Data management overview. In this template, customer data was exported from the **CustCustomerV3Entity** table. Set up **CustCustomerV3Entity**, and remove the **FullPrimaryAddress** field map from the mapping. Add the **DataAreaId** field to the CSV file. Rename the exported file **01-CustomersV3Export-Customers V3.csv**, and upload it to the Azure storage account that you named **ce-data**.



GZ	HA	HB	HC	HD
WAREHOU	WAREHOU	WAREHOU	WRITEOFFF	DataAreaId
				USMF
				GBSI
				GBSI

2. Download the sample customer file.

3. Run **BootstrapAccountsPipeline** from Azure Data Factory.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Organization hierarchy in Dataverse

2/18/2021 • 3 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

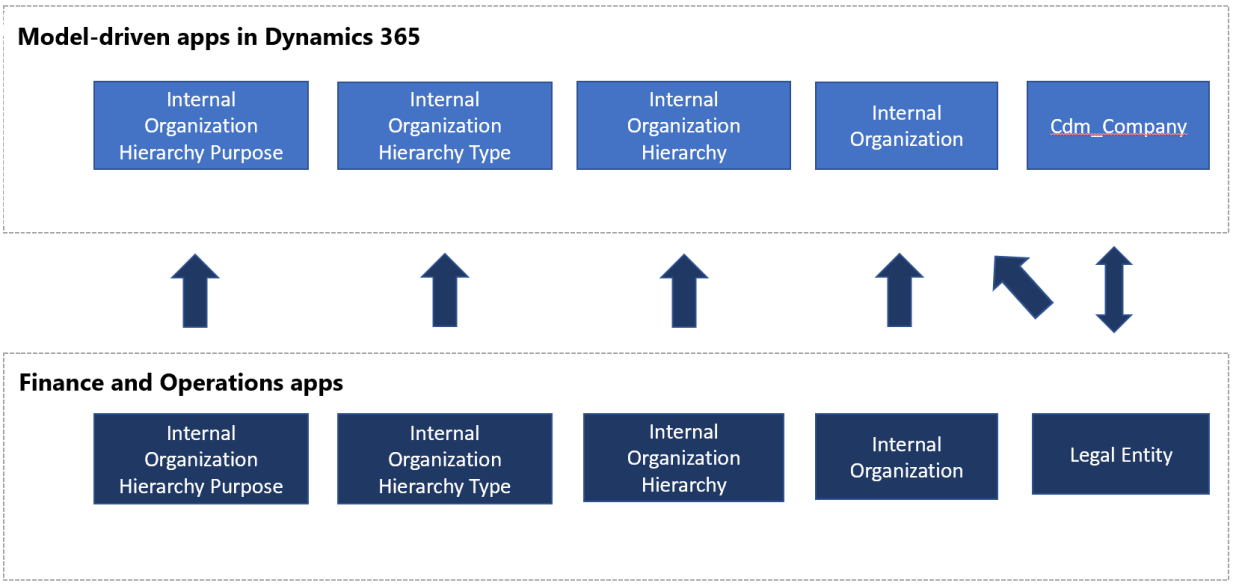
This topic will be updated soon to reflect the latest terminology.

Because Dynamics 365 Finance is a financial system, *organization* is a core concept, and system setup starts with the configuration of an organization hierarchy. Business financials can then be tracked at the organization level and also at any level in the organization hierarchy.

Although Dataverse doesn't have the concept of an organization hierarchy, it does have a few loose concepts, such as total sales revenue. As part of Dataverse integration, the organization hierarchy data structure is added to Dataverse.

Data flow

A business ecosystem that consists of Finance and Operations apps and Dataverse will continue to have an organization hierarchy. This organization hierarchy is built on Finance and Operations apps, but it's exposed in Dataverse for informational and extensibility purposes. The following illustration shows the organization hierarchy information that is exposed in Dataverse as a one-way data flow from Finance and Operations apps to Dataverse.



Organization hierarchy table maps are available for one-way synchronization of data from Finance and Operations apps to Dataverse.

Templates

Product information contains all the information related to the product and its definition, such as the product

dimensions or the tracking and storage dimensions. As the following table shows, a collection of table maps is created to sync products and related information.

FINANCE AND OPERATIONS APPS	OTHER DYNAMICS 365 APPS	DESCRIPTION
Organization hierarchy purposes	msdyn_internalorganizationhierarchypurposes	This template provides one-way synchronization of the Organization Hierarchy Purpose table.
Organization hierarchy type	msdyn_internalorganizationhierarchytypes	This template provides one-way synchronization of the Organization Hierarchy Type table.
Organization hierarchy - published	msdyn_internalorganizationhierarchies	This template provides one-way synchronization of the Organization Hierarchy Published table.
Operating unit	msdyn_internalorganizations	
Legal entities	msdyn_internalorganizations	
Legal entities	cdm_companies	Provides bidirectional synchronization of legal entity (company) information.

Mapping tables

Mapping types

There are several different mapping types. The following table explains the symbols used in the template tables.

SYMBOL	DESCRIPTION
>	One-way
>>	One-way, and data is transformed in the process.
=	Bidirectional
><	Bidirectional, and data is transformed in the process.
<<	One-way, and data is transformed in the process.

Filters

The source filter and reverse source filter determine which rows are synchronized.

Default values

If a synchronized field does not exist in either the Finance and Operations table or the other Dynamics 365 table, then a default value is assigned in the synchronized table. In some cases, the default value is an integer that is a lookup to an attribute value in the Common Data Model. For example, in the Contacts table of the Common Data Model, the default value of **address1_addresstypecode** is **3**. In the Common Data Model, for address1AddressTypeCode the value of **3** is **Primary address**.

Organization hierarchy purposes to msdyn_internalorganizationhierarchypurposes

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
HIERARCHYTYPE	>	msdyn_hierarchypurposetypename	
HIERARCHYTYPE	>	msdyn_hierarchytype.msdyn_name	
HIERARCHYPURPOSE	>>	msdyn_hierarchypurpose	
IMMUTABLE	>>	msdyn_immutable	
SETASDEFAULT	>>	msdyn_setasdefault	

Organization hierarchy type to msdyn_internalorganizationhierarchytypes

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
NAME	>	msdyn_name	

Organization hierarchy - published to msdyn_internalorganizationhierarchies

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
VALIDTO	>	msdyn_validto	
VALIDFROM	>	msdyn_validfrom	
HIERARCHYTYPE	>	msdyn_hierarchytypename	
PARENTORGANIZATIONPARTYNUMBER	>	msdyn_parentpartyid	
CHILDORGANIZATIONPARTYNUMBER	>	msdyn_childpartyid	
HIERARCHYTYPE	>	msdyn_hierarchytypeid.msdyn_name	
CHILDORGANIZATIONPARTYNUMBER	>	msdyn_childid.msdyn_partynumber	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
PARENTORGANIZATIONPARTYNUMBER	>	msdyn_parentid.msdyn_partynumber	

Internal Organization

Internal organization information in Dataverse comes from two tables, **operating unit** and **legal entities**.

Operating unit to msdyn_internalorganizations

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
LANGUAGEID	>	msdyn_languageid	
NAMEALIAS	>	msdyn_namealias	
NAME	>	msdyn_name	
PARTYNUMBER	>	msdyn_partynumber	
OPERATINGUNITTYPE	>>	msdyn_type	

Legal entities to msdyn_internalorganizations

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
NAMEALIAS	>	msdyn_namealias	
LANGUAGEID	>	msdyn_languageid	
NAME	>	msdyn_name	
PARTYNUMBER	>	msdyn_partynumber	
none	>>	msdyn_type	806380000
LEGALENTITYID	>	msdyn_companycode	

Legal entities to cdm_companies

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
NAME	=	cdm_name	
LEGALENTITYID	=	cdm_companycode	

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Access to finance and tax reference data

2/18/2021 • 2 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

Every business works with a basic set of financial data, such as the fiscal calendar year, the currency that business is transacted in, the accounts that the money to run the business comes in to or goes out of, tax rates, and remittance. This data resides in Finance and Operations apps. However, it's exposed to Dataverse so that model-driven apps in Microsoft Dynamics 365 can have a single source for finance and tax data. In this way, data is uniform across the business ecosystem.

Finance and tax data is integrated by using the following mappings:

- Integrated ledger
- Integrated tax master

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Integrated ledger

2/18/2021 • 4 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

In a business application, ledger data defines the core set up for how a company does business. For example, ledger data describes the fiscal year the company follows, the currencies it transacts in, and the accounts it uses. This topic describes the integration of this core financial data.

Templates

Ledger data includes a collection of core financial table maps that work together during data interaction, as shown in the following table.

FINANCE AND OPERATIONS APPS	MODEL-DRIVEN APP IN DYNAMICS 365	DESCRIPTION
Currencies	transactioncurrencies	
FiscalCalendar	msdyn_fiscalcalendars	
FiscalCalendarYear	msdyn_fiscalcalendaryears	
ExchRateType	msdyn_exchangeratetypes	
ExchangeRateCurrencyPair	msdyn_currencyexchangeratepairs	
FiscalPeriodEntity	msdyn_fiscalcalendarperiods	
MainAccountCategory	msdyn_mainaccountcategory	
MainAccount	msdyn_mainaccounts	
Ledger	msdyn_ledgers	
ExchangeRates	msdyn_currencyexchangerates	
FinancialCalendarPeriod	msdyn_fiscalcalendarperiods	
DimensionAttributeEntity	msdyn_dimensionattributes	
DimensionIntegrationFormatEntity	msdyn_financialdimensionformats	

FINANCE AND OPERATIONS APPS	MODEL-DRIVEN APP IN DYNAMICS 365	DESCRIPTION
LedgerChartOfAccounts	msdyn_chartofaccounts	

Mapping tables

Mapping types

There are several different mapping types. The following table explains the symbols used in the template tables.

SYMBOL	DESCRIPTION
>	One-way
>>	One-way, and data is transformed in the process.
=	Bidirectional
><	Bidirectional, and data is transformed in the process.
<<	One-way, and data is transformed in the process.

Filters

The source filter and reverse source filter determine which rows are synchronized.

Default values

If a synchronized field does not exist in either the Finance and Operations table or the other Dynamics 365 table, then a default value is assigned in the synchronized table. In some cases, the default value is an integer that is a lookup to an attribute value in the Common Data Model. For example, in the Contacts table of the Common Data Model, the default value of **address1_addresstypecode** is **3**. In the Common Data Model, for address1AddressTypeCode the value of **3** is **Primary address**.

Currencies to transactioncurrencies

This template synchronizes data between Finance and Operations apps and Dataverse.

Source filter: `((CURRENCYCODE != "999"))`

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
CURRENCYCODE	=	isocurrencycode	
NAME	=	currencyname	
SYMBOL	=	currencysymbol	
none	>>	exchangerate	1

Fiscal calendar integration entity to msdyn_fiscalcalendars

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
CALENDARID	=	msdyn_calendar	
DESCRIPTION	=	msdyn_description	

Fiscal calendar year integration entity to msdyn_fiscalcalendaryears

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
FISCALCALENDAR_CALENDARID	=	msdyn_fiscalcalendarname	
NAME	=	msdyn_name	
STARTDATE	=	msdyn_startdate	
ENDDATE	=	msdyn_enddate	
FISCALCALENDAR_CALENDARID	=	msdyn_calendar.msdyn_calendar	

Exchange rate type to msdyn_exchangeratetypes

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
NAME	=	msdyn_name	
DESCRIPTION	=	msdyn_description	

Exchange rate currency pair to msdyn_currencyexchangeratepairs

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
EXCHANGERATEDISPLAYFACTOR	><	msdyn_displayfactor	
EXCHANGERATETYPENAME	=	msdyn_currencyexchangeratetypeid.msdyn_name	
FROMCURRENCYCODE	=	msdyn_fromtransactioncurrencyid.isocurrencycode	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
TOCURRENCYCODE	=	msdyn_totransactioncurrencyid.isocurrencycode	

Main account categories to msdyn_mainaccountcategories

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
MAINACCOUNTCATEGORY	=	msdyn_mainaccountcategory	
REFERENCEID	=	msdyn_referenceid	
DESCRIPTION	=	msdyn_description	
DISPLAYORDER	=	msdyn_displayorder	
CLOSED	><	msdyn_closed	
MAINACCOUNTTYPE	><	msdyn_mainaccounttypevalue	

Main account to msdyn_mainaccounts

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
MAINACCOUNTID	=	msdyn_accountnumber	
CHARTOFACCOUNTS	=	msdyn_chartofaccounts.msdyn_name	
NAME	=	msdyn_name	
BALANCECONTROL	><	msdyn_balancecontrol	
EXCHANGEADJUSTMENTRATETYPE	=	msdyn_exchangeadjustmentratetype.msdyn_name	
CLOSING	><	msdyn_closing	
REPORTINGEXCHANGEADJUSTMENTRATETYPE	=	msdyn_reportingexchangeadjustmentratetype.msdyn_name	
DEBITCREDITREQUIREMENT	><	msdyn_debitcreditrequirement	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
FINANCIALREPORTINGEXCHANGERATETYPE	=	msdyn_financialreportingexchangeratetype.msdyn_name	
FOREIGNCURRENCYREVALUATION	> <	msdyn_foreigncurrencyrevaluation	
MAINACCOUNTCATEGORY	=	msdyn_mainaccountcategoryname	
MANDATORYPAYMENTREFERENCE	> <	msdyn_mandatorypaymentreference	
MONETARY	> <	msdyn_monetary	
OFFSETACCOUNTDISPLAYVALUE	=	msdyn_offsetaccount	
POSTINGTYPE	> <	msdyn_postingtype	
SRUCODE	=	msdyn_srucode	
VALIDATECURRENCY	> <	msdyn_validatecurrencycode	
VALIDATEUSER	> <	msdyn_validateuser	
DEBITCREDITDEFAULT	> <	msdyn_debitcreditdefault	
DEFAULTCURRENCY	=	msdyn_defaultcurrency.isocurrencycode	
MAINACCOUNTTYPE	> <	msdyn_mainaccounttype	
FINANCIALREPORTINGCURRENCYTRANSLATIONTYPE	> <	msdyn_financialreportingcurrencytrantype	
USER	=	msdyn_user	
VALIDATEPOSTINGTYPE	> <	msdyn_validateposting	

Ledger to msdyn_ledgers

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
LEGALENTITYID	> >	msdyn_company.cdm_companycode	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
DESCRIPTION	>>	msdyn_description	
ACCOUNTINGCURRENCY	>>	msdyn_accountingcurrency.isocurrencycode	
ISBUDGETCONTROLENABLED	>>	msdyn_isbudgetcontrolenabled	
NAME	>>	msdyn_name	
REPORTINGCURRENCY	>>	msdyn_reportingcurrency.isocurrencycode	
BUDGETEXCHANGERATETYPE	>>	msdyn_budgetexchangeratetype.msdyn_name	
CHARTOFACCOUNTS	>>	msdyn_chartofaccounts.msdyn_name	
EXCHANGERATETYPE	>>	msdyn_exchangeratetype.msdyn_name	
FISCALCALENDAR	>>	msdyn_fiscalcalendar.msdyn_calendar	
REPORTINGCURRENCYEXCHANGERATETYPE	>>	msdyn_reportingcurrencyexchangeratetype.msdyn_name	

CDS Exchange Rates to msdyn_currencyexchangerates

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
EXCHANGERATETYPENAME	>>	msdyn_exchangeratetypename	
FROMCURRENCYCODE	>>	msdyn_fromcurrencycode	
TOCURRENCYCODE	>>	msdyn_tocurrencycode	
RATE	>>	msdyn_exchangerate	
VALIDFROM	>>	msdyn_validfrom	
VALIDTO	>>	msdyn_validto	

Fiscal calendar period to msdyn_fiscalcalendarperiods

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
COMMENTS	=	msdyn_comments	
ENDDATE	=	msdyn_enddate	
MONTH	><	msdyn_month	
CALENDAR	=	msdyn_fiscalcalendar.msdyn_calendar	
QUARTER	><	msdyn_quarter	
SHORTNAME	=	msdyn_shortname	
STARTDATE	=	msdyn_startdate	
TYPE	><	msdyn_fiscalperiodtype	
PERIODNAME	=	msdyn_periodname	
FISCALYEAR	=	msdyn_fiscalcalendaryear.msdyn_name	
CALENDAR	=	msdyn_fiscalcalendaryear.msdyn_fiscalcalendarname	

Financial dimensions to msdyn_dimensionattributes

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
DIMENSIONNAME	=	msdyn_dimensionname	
COPYVALUESONCREATE	><	msdyn_copyvaluesoncreate	
REPORTCOLUMNNAME	=	msdyn_reportcolumnname	
GIVEDERIVEDDIMENSIONSPRECEDENCE	><	msdyn_givederiveddimensionsprecedence	
UseValuesFrom	=	msdyn_usevaluesfrom	
DimensionValueMask	=	msdyn_dimensionvaluemask	

Financial dimension format to msdyn_financialdimensionformats

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
DIMENSIONFORMATNAME	=	msdyn_dimensionformatname	
DIMENSIONFORMATTYPE	><	msdyn_dimensionformattype	
FINANCIALDIMENSIONFORMAT	=	msdyn_financialdimensionformat	
ISACTIVE	><	msdyn_isactive	

Chart of accounts to msdyn_chartofaccountses

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
DESCRIPTION	=	msdyn_description	
MAINACCOUNTMASK	=	msdyn_mainaccountmask	
CHARTOFACCOUNTS	=	msdyn_name	

Integrated tax

2/18/2021 • 3 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

Tax setup data defines the setup for both indirect taxes (VAT, GST, Sales tax) and withholding tax. It describes the tax calculation rule, tax rate, tax accounting, settlement, and other concepts.

Templates

Tax data includes a collection of table maps that work together during data interaction, as shown in the following table.

FINANCE AND OPERATIONS APPS	MODEL-DRIVEN APPS IN DYNAMICS 365	DESCRIPTION
Item sales tax group	msdyn_taxitemgroups	
Sales tax authorities	msdyn_taxauthorities	
Sales tax exempt code entity CDS	msdyn_taxexemptcodes	
Sales tax groups	msdyn_taxgroups	
Sales tax ledger posting groups V2	msdyn_taxpostinggroups	
Withholding tax codes	msdyn_withholdingtaxcodes	
Withholding tax groups	msdyn_withholdingtaxgroups	

Mapping tables

Mapping types

There are several different mapping types. The following table explains the symbols used in the template tables.

SYMBOL	DESCRIPTION
>	One-way
>>	One-way, and data is transformed in the process.

SYMBOL	DESCRIPTION
=	Bidirectional
><	Bidirectional, and data is transformed in the process.
<<	One-way, and data is transformed in the process.

Filters

The source filter and reverse source filter determine which rows are synchronized.

Default values

If a synchronized field does not exist in either the Finance and Operations table or the other Dynamics 365 table, then a default value is assigned in the synchronized table. In some cases, the default value is an integer that is a lookup to an attribute value in the Common Data Model. For example, in the Contacts table of the Common Data Model, the default value of **address1_addresstypecode** is **3**. In the Common Data Model, for address1AddressTypeCode the value of **3** is **Primary address**.

Item sales tax group to msdyn_taxitemgroups

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
TAXITEMGROUP	=	msdyn_name	
NAME	=	msdyn_description	

Sales tax authorities to msdyn_taxauthorities

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
TAXAUTHORITYCODE	=	msdyn_taxauthoritycode	
TAXAUTHORITYIDENTIFICATION	=	msdyn_taxauthorityidentificator	
DESCRIPTION	=	msdyn_description	
REPORTLAYOUT	><	msdyn_taxreportlayout	
ROUNDOFFTYPE	><	msdyn_roundofftype	
ROUNDOFF	=	msdyn_roundoff	
EMAIL	=	msdyn_email	
PHONE	=	msdyn_phone	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
URL	=	msdyn_url	

Sales tax exempt code entity CDS to msdyn_taxexemptcodes

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
NAME	=	msdyn_name	
DESCRIPTION	=	msdyn_description	

Sales tax groups to msdyn_taxgroups

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
TAXGROUPCODE	=	msdyn_name	
DESCRIPTION	=	msdyn_description	

Sales tax ledger posting groups V2 to msdyn_taxpostinggroups

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
TAXPOSTINGGROUPCODE	=	msdyn_name	
DESCRIPTION	=	msdyn_description	

Withholding tax codes to msdyn_withholdingtaxcodes

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
WITHHOLDINGCODE	=	msdyn_name	
WITHHOLDINGTAXNAME	=	msdyn_description	
WITHHOLDINGTAXROUNDOFF	=	msdyn_roundoff	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
WITHHOLDINGTAXROUNDOFFTYPE	><	msdyn_roundofftype	
CURRENCYCODEID	=	msdyn_currency.isocurrencycode	
WITHHOLDINGTAXBASE	><	msdyn_taxableamountorigin	

Withholding tax groups to msdyn_withholdingtaxgroups

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
WITHHOLDINGTAXGROUPCODE	=	msdyn_name	
DESCRIPTION	=	msdyn_description	

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Integrate procurement between Supply Chain Management and Field Service

2/18/2021 • 15 minutes to read • Edit Online

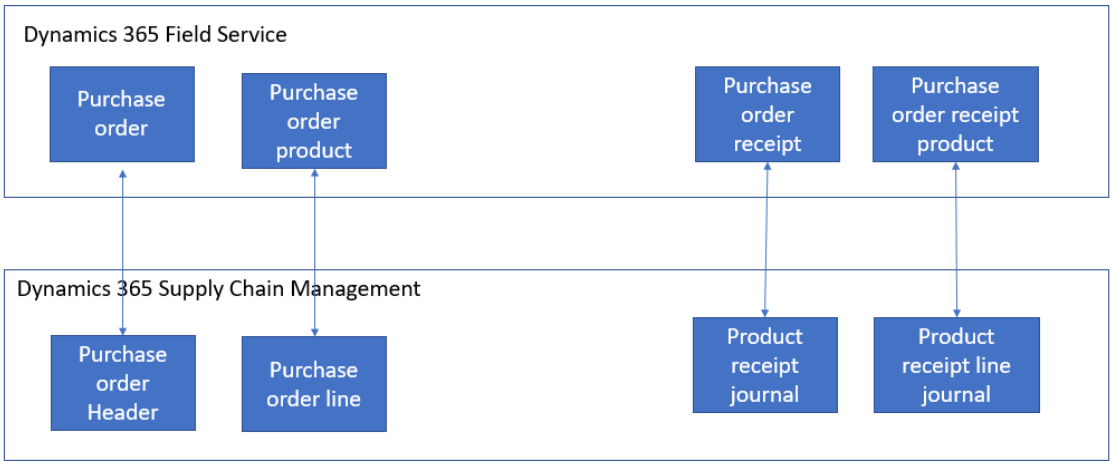
IMPORTANT

Some or all of the functionality noted in this topic is available as part of a preview release. The content and the functionality are subject to change. For more information about preview releases, see Service update availability.

Microsoft Dynamics 365 Supply Chain Management provides robust procurement functionality. Dynamics 365 Field Service offers similar functionality that supports the purchasing processes that are associated with the service process. The functionality in these two apps is integrated through dual-write, and the resulting cross-functional use cases are enabled through table mappings, solution logic, views, and forms.

This integration supports purchase order creation and, in most cases, updates from both apps. However, Supply Chain Management controls pricing, addresses, and product receipt. Several powerful cross-functional use cases are enabled for organizations that use both Field Service and Supply Chain Management. These use cases enable procurements to be initiated and tracked across both systems.

The following illustration shows the tables in both systems and how they are mapped to each other. Purchase orders in Field Service reference an *account* row, whereas purchase orders in Supply Chain Management reference a *vendor* row. To resolve the integration, dual-write uses a reference to link *vendor* rows with *account* rows. For more information, see Integrated vendor master.



Prerequisites

To integrate Supply Chain Management with Field Service, you must install the following components:

- Field Service version 8.8.31.60 or later, for comprehensive purchase order integration
- Supply Chain Management version 10.0.14 or later
- Dual-write, to run the OneFSSCM solution

Installation guidelines

Prerequisites

- **Dual-write** – For more information, see the Dual-write home page.
- **Dynamics 365 Field Service** – For more information, see How to install Dynamics 365 Field Service.

When they are enabled in Microsoft Dataverse, dual-write and Field Service introduce several solution layers that extend the environment with new metadata, forms, views, and logic. These solutions can be enabled in any

order, though you typically install in the order that is given here:

1. **Field Service Common** – Field Service Common is installed when Field Service is installed in the environment.
2. **Field Service (Anchor)** – Field Service (Anchor) is installed when Field Service is installed in the environment.
3. **Supply Chain Management Extended** – Supply Chain Management Extended is automatically installed when dual-write is enabled in an environment.
4. **OneFSSCM solution** – OneFSSCM is automatically installed by whichever solution (Field Service or Supply Chain Management) is installed last.
    - If Field Service is already installed in the environment, and you enable dual-write, which installs Supply Chain Management Extended, OneFSSCM is installed.
    - If Supply Chain Management Extended is already installed in the environment, and you install Field Service, OneFSSCM is installed.

Initial synchronization

To create new purchase orders and work with existing purchase orders, you must sync the reference data between Supply Chain Management and Dataverse. You use the initial write functionality to detect the table relationships and find the tables that you must enable for a given map.

You must sync the following tables:

- Product templates

  When you run the initial write, you get a full list of the tables that are required. Here are some examples of these templates:

  - All products
  - Released products V2
  - Dataverse released distinct products
- Sites
- Warehouses
- Procurement categories templates

  Here are some examples of these templates:

  - Procurement categories
  - Pro
  - Product category hierarchy
  - Product category assignments
- Vendor templates, such as Vendor V2
- Contact person templates, such as Dataverse Contacts V2
- Worker templates, such as Worker

Synchronization of the tables ensures that all documents (purchase orders and product receipts) in Supply Chain Management are available in Dataverse.

Account and Vendor tables

Purchase orders in Field Service rely on the Account table to track vendors. Therefore, the Dataverse tables for purchase orders use accounts to track vendors. To accommodate this key difference, the following four workflows must be activated to keep the accounts and vendors in sync:

- Create Vendors in Accounts table
- Create Vendors in Vendors table
- Update Vendors in Accounts table

- Update Vendors in Vendors table

If OneFSSCM is installed, because both Field Service and Supply Chain Management Extended are installed, these workflows are automatically activated. If Field Service isn't installed, but you want to integrate the purchase order tables with Dataverse, you must activate these workflows. In both cases, unless you start from scratch, you might have to ensure that all vendors are created as accounts in Dataverse before you create purchase orders. Otherwise, errors might occur.

Initial synchronization

After all the prerequisites are in place, if you want existing purchase orders and product receipts to be available in both systems, you must do an initial synchronization of the following templates:

- Purchase Order Header V2
- CDS Purchase Order Line
- CDS Purchase Order Line soft delete
- Purchase Order Receipt
- Purchase Order Receipt Product

Mappings with logic

The procurement integration extends the product mapping with the following logic to ensure that the **Field Service Product Type** column is correctly set in the products table in Dataverse:

- If **Product Type** is set to *Product*, and **Item model group, Stocked product** is set to *True*, **Field Service Product Type** is set to *Inventory*.
- If **Product Type** is set to *Product*, and **Item model group, Stocked product** is set to *False*, **Field Service Product Type** is set to *Non-Inventory*.
- If **Product Type** is set to *Service*, **Field Service Product Type** is set to *Service*.

In addition, Dataverse includes logic that maps vendors with their related accounts. This logic sets the default invoice vendor account. On create, server-side plug-in logic sets the default invoice vendor account from the vendor that is related to the account. The vendor has a reference to the invoice account that is used to set this value.

Supported scenarios

- Purchase orders can be created and updated by Dataverse users. However, the process and data are controlled by Supply Chain Management. The constraints on updates to purchase order columns in Supply Chain Management apply when updates come from Field Service. For example, you can't update a purchase order if it has been finalized.

- If the purchase order is controlled by change management in Supply Chain Management, a Field Service user can update the purchase order only when the Supply Chain Management approval status is *Draft*.

- Several columns are managed only by Supply Chain Management and can't be updated in Field Service. To learn which columns can't be updated, review the mapping tables in the product. For the sake of simplicity, most of these columns are set to read-only on Dataverse pages.

  For example, the columns for price information are managed by Supply Chain Management. Supply Chain Management has trade agreements that Field Service can benefit from. columns such as **Unit price**, **Discount**, and **Net amount** come only from Supply Chain Management. To ensure that the price is synced to Field Service, you should use the **Sync** feature on the **Purchase Order** and **Purchase Order Product** pages in Dataverse when purchase order data has been entered. For more information, see Sync with the Dynamics 365 Supply Chain Management procurement data on demand.

- The **Totals** column is available only in Field Service, because there are no up-to-date totals of the purchase order in Supply Chain Management. The totals in Supply Chain Management are calculated based on multiple parameters that aren't available in Field Service.

- Purchase order lines where only a procurement category is specified, or where the product that is specified is an item of the *Service* product type or Field Service product type, can be initiated only in

Supply Chain Management. The lines are then synced to Dataverse and are visible in Field Service.

- If only Field Service is installed, not Supply Chain Management, the **Warehouse** column is mandatory on the purchase order. However, if Supply Chain Management is installed, this requirement is relaxed, because Supply Chain Management allows for purchase order lines where no warehouse is specified in certain situations.
- Product receipts (purchase order receipts in Dataverse) are managed by Supply Chain Management and can't be created from Dataverse if Supply Chain Management is installed. The product receipts from Supply Chain Management are synced from Supply Chain Management to Dataverse.
- Under-delivery is allowed in Supply Chain Management. The OneFSSCM solution adds logic so that, when the product receipt line (or purchase order receipt product in Dataverse) is created or updated, an inventory journal row is created in Dataverse to adjust the remaining quantity that is on order for under-delivery scenarios.

Unsupported scenarios

- Field Service prevents lines from being added to a canceled purchase order in Supply Chain Management. As a workaround, you can change the system status of the purchase order in Field Service, and then add the new line in either Field Service or Supply Chain Management.
- Although procurement rows affect inventory levels in both systems, this integration doesn't ensure inventory alignment across Supply Chain Management and Field Service. Both Field Service and Supply Chain Management have other processes that update inventory levels. Those processes are outside the scope of procurement.

Status management

The statuses of purchase orders in Field Service differ from the statuses in Supply Chain Management.

Field Service purchase order and purchase order product statuses

HEADER – SYSTEM STATUS	HEADER - APPROVAL STATUS	ITEM STATUS
• Draft • Submitted • Cancelled • Product received • Billed	• Null • Approved • Rejected	• Pending • Received • Cancelled

Supply Chain Management purchase order and purchase order line statuses

Line approval statuses are active only when there is a line workflow.

HEADER – DOCUMENTS STATUS	HEADER - APPROVAL STATUS	LINE STATUS	LINE APPROVAL STATUS
• Open Order (Back order) • Received • Invoiced • Cancelled	• Draft • In Review • Approved • Rejected • In External Review • Confirmed • Finalized	• Open Order (back order) • Received • Invoiced • Cancelled	• Not Submitted • In Review • Approved • Rejected

The following rules are applied to the status columns:

- The status in Supply Chain Management can't be updated from Field Service. However, in some cases, the status in Field Service will be updated when the purchase order status in Supply Chain Management is changed.
- If a purchase order in Supply Chain Management is under change management, and a change is being processed, the approval status is *Draft* or *In Review*. In this case, the Field Service approval status will be set

to *Null*.

- If the purchase order approval status in Supply Chain Management is set to *Approved*, *In External review*, *Confirmed*, or *Finalized*, the Field Service purchase order approval status will be set to *Approved*.
- If the purchase order approval status in Supply Chain Management is set to *Rejected*, the Field Service purchase order approval status will be set to *Rejected*.
- If the document header status in Supply Chain Management is changed to *Open order (Back order)*, and the Field Service purchase order status is *Draft* or *Cancelled*, the Field Service purchase order status will be changed to *Submitted*.
- If the document header status in Supply Chain Management is changed to *Cancelled*, and no purchase order receipt products in Field Service are associated with the purchase order (via purchase order products), the Field Service system status is set to *Cancelled*.
- If purchase order line status in Supply Chain Management is *Cancelled*, the purchase order product status in Field Service is set to *Cancelled*. In addition, if the purchase order line status in Supply Chain Management is changed from *Cancelled* to *Back Order*, the purchase order product item status in Field Service is set to *Pending*.

Sync with the Supply Chain Management procurement data on demand

Supply Chain Management includes procurement data that handles trade agreements, discounts, and other scenarios that rely on secondary processes in Supply Chain Management. The procurement engine uses complex rules to determine the best price for a given purchase order. When you use dual-write, data isn't always kept synchronous across the two environments, especially in scenarios where the row was created or updated from Dataverse and might trigger follow-on processes in Supply Chain Management.

Sync the procurement data from Supply Chain Management

1. In Dataverse, go to **Inventory** > **Purchase Order**.
2. Select **New** to create a new purchase order, or select the row for an existing purchase order.
3. From the purchase order or purchase order line.
4. On the Action Pane, select **Sync**.

All columns from Dataverse and Field Service that are shared by Supply Chain Management are synced.

Here are the situations where you might use the **Sync** function:

- If you make multiple successive changes to the same row from Dataverse, run the **Sync** function.
- If you aren't sure whether a change might be the second successive change from Dataverse, it might make sense to run the **Sync** function.
- If you receive an error message about updating a value from Supply Chain Management, run the **Sync** function, and then retry the update in Dataverse.

Cancelling the posting process

If the product receipt posting process is cancelled during processing, then dual-write might create a product receipt row in Dataverse, but not create a product receipt row in Supply Chain Management. This situation happens because dual-write does not support distributed transactions.

Templates

The following templates are available for the integration of procurement-related documents.

SUPPLY CHAIN MANAGEMENT	FIELD SERVICE	DESCRIPTION
Purchase order header V2	msdyn_Purchaseorders	This table contains the columns that represent the purchase order header.

SUPPLY CHAIN MANAGEMENT	FIELD SERVICE	DESCRIPTION
Purchase order line entity	msdyn_PurchaseOrderProducts	This table contains the rows that represent lines on a purchase order. The product number is used for synchronization. This identifies the product as a stock keeping unit (SKU), including product dimensions. For more information about product integration with Dataverse, see Unified product experience.
Product receipt header	msdyn_purchaseorderreceipts	This table contains the product receipt headers that are created when a product receipt is posted in Supply Chain Management.
Product receipt line	msdyn_purchaseorderreceiptproducts	This table contains the product receipt lines that are created when a product receipt is posted in Supply Chain Management.
Purchase order line soft deleted entity	msdyn_purchaseorderproducts	This table contains information about purchase order lines that are soft-deleted. A purchase order line in Supply Chain Management can be soft-deleted only when the purchase order has been confirmed or approved, if change management is turned on. The row exists in the Supply Chain Management database and is marked as **IsDeleted**. Because Dataverse doesn't have a concept of soft-deletion, it's important that this information be synced to Dataverse. In this way, lines that are soft-deleted in Supply Chain Management can automatically be deleted from Dataverse. In this case, the logic for deleting a line in Dataverse is located in Supply Chain Management Extended.

Mapping tables

Mapping types

There are several different mapping types. The following table explains the symbols used in the template tables.

SYMBOL	DESCRIPTION
>	One-way
>>	One-way, and data is transformed in the process.
=	Bidirectional
><	Bidirectional, and data is transformed in the process.
<<	One-way, and data is transformed in the process.

Filters

The source filter and reverse source filter determine which rows are synchronized.

Default values

If a synchronized field does not exist in either the Finance and Operations table or the other Dynamics 365 table, then a default value is assigned in the synchronized table. In some cases, the default value is an integer that is a lookup to an attribute value in the Common Data Model. For example, in the Contacts table of the Common Data Model, the default value of **address1_addresstypecode** is **3**. In the Common Data Model, for address1AddressTypeCode the value of **3** is **Primary address**.

Product receipt header to msdyn_purchaseorderreceipts

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
RECORDID	>>	`msdyn_recordid`	
PRODUCTRECEIPTDATE	>>	`msdyn_datereceived`	
DELIVERYADDRESSLATITUDE	>>	`msdyn_deliveryaddresslatitude`	
DELIVERYADDRESSLONGITUDE	>>	`msdyn_deliveryaddresslongitude`	
PURCHASEORDERNUMBER	>>	`msdyn_purchaseorder.msdyn_name`	
DELIVERYMODEID	>>	`msdyn_shipvia.msdyn_name`	
DELIVERYTERMSID	>>	`msdyn_deliveryterm.msdyn_termscode`	
ORDERVENDORACCOUNTNUMBER	>>	`msdyn_ordervendor.msdyn_vendoraccountnumber`	
REQUESTERPERSONNELNUMBER	>>	`msdyn_requesterpersonnel.cdm_workernumber`	
ISDELIVERYADDRESSPRIVATE	>>	`msdyn_isdeliveryaddressprivate`	
DELIVERYADDRESSCOUNTRYREGIONID	>>	`msdyn_deliveryaddresscountryregionid`	
DELIVERYADDRESSCOUNTYID	>>	`msdyn_deliveryaddresscountyid`	
DELIVERYADDRESSSTATEID	>>	`msdyn_deliveryaddressstateid`	
DELIVERYADDRESSZIPCODE	>>	`msdyn_deliveryaddresszipcode`	
DELIVERYADDRESSNAME	>>	`msdyn_deliveryaddressname`	
DELIVERYADDRESSTIMEZONE	>>	`msdyn_deliveryaddresstimezone`	
DELIVERYADDRESSPOSTBOX	>>	`msdyn_deliveryaddresspostbox`	
DELIVERYADDRESSSTREETNUMBER	>>	`msdyn_deliveryaddressstreetnumber`	
DELIVERYADDRESSSTREET	>>	`msdyn_deliveryaddressstreet`	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
PRODUCTRECEIPTNUMBER	>>	msdyn_name	
ATTENTIONINFORMATION	>>	msdyn_note	
DELIVERYCITYINKANA	>>	msdyn_deliverycityinkana	
DELIVERYSTREETINKANA	>>	msdyn_deliverystreetinkana	
FORMATTEDDELIVERYADDRESS	>>	msdyn_formatteddeliveryaddress	
DELIVERYADDRESSLOCATIONID	>>	msdyn_deliveryaddresslocationid	
DELIVERYADDRESSCITY	>>	msdyn_deliveryaddresscity	
DELIVERYADDRESSDESCRIPTION	>>	msdyn_deliveryaddressdescription	
DELIVERYADDRESSDISTRICTNAME	>>	msdyn_deliveryaddressdistrictname	
DELIVERYBUILDINGCOMPLIMENT	>>	msdyn_deliverybuildingcompliment	
DELIVERYADDRESSDUNSNUMBER	>>	msdyn_deliveryaddressdunsnumber	

Product receipt line to msdyn_purchaseorderreceiptproducts

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
RECORDID	>>	msdyn_recordid	
DELIVERYADDRESSCOUNTRYREGIONID	>>	msdyn_deliveryaddresscountryregionid	
DELIVERYADDRESSCOUNTYID	>>	msdyn_deliveryaddresscountyid	
DELIVERYADDRESSSTATEID	>>	msdyn_deliveryaddressstateid	
EXPECTEDDELIVERYDATE	>>	msdyn_expecteddeliverydate	
EXTERNALITEMNUMBER	>>	msdyn_externalitemnumber	
ITEMBATCHNUMBER	>>	msdyn_itembatchnumber	
ITEMSERIALNUMBER	>>	msdyn_itemserialnumber	
LINEDESCRIPTION	>>	msdyn_linedescription	
ORDEREDPURCHASEQUANTITY	>>	msdyn_orderedpurchasequantity	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
PROCUREMENTPRODUCTCATEGORYNAME	>>	msdyn_procurementproductcategory.msdyn_name	
PROCUREMENTPRODUCTCATEGORYHIERARCHYNAME	>>	msdyn_procurementproductcategory.msdyn_hierarchy.msdyn_name	
PRODUCTRECEIPTDATE	>>	msdyn_productreceiptdate	
PRODUCTRECEIPTHEADERRECORDID	>>	msdyn_purchaseorderreceipt.msdyn_recordid	
PRODUCTRECEIPTNUMBER	>>	msdyn_productreceiptnumber	
PURCHASEORDERLINENUMBER	>>	msdyn_purchaseorderproduct.msdyn_lineorder	
PURCHASEORDERNUMBER	>>	msdyn_purchaseorderproduct.msdyn_purchaseorder.msdyn_name	
PURCHASEORDERNUMBER	>>	msdyn_purchaseorder.msdyn_name	
PURCHASEUNITSYMBOL	>>	msdyn_purchaseunitsymbol.msdyn_symbol	
RECEIVEDINVENTORYQUANTITY	>>	msdyn_receivedinventoryquantity	
RECEIVEDINVENTORYSTATUSID	>>	msdyn_receivedinventorystatusid	
RECEIVEDPURCHASEQUANTITY	>>	msdyn_quantity	
RECEIVINGSITEID	>>	msdyn_receivingsiteid.msdyn_siteid	
RECEIVINGWAREHOUSEID	>>	msdyn_associatetowarehouse.msdyn_warehouseidentifier	
RECEIVINGWAREHOUSELOCATIONID	>>	msdyn_receivingwarehouselocation.msdyn_warehouselocationid	
RECEIVINGWAREHOUSEID	>>	msdyn_receivingwarehouselocation.msdyn_warehouse.msdyn_warehouseiden	
REMAININGINVENTORYQUANTITY	>>	msdyn_remaininginventoryquantity	
REMAININGPURCHASEQUANTITY	>>	msdyn_remainingpurchasequantity	
PRODUCTNUMBER	>>	msdyn_product.msdyn_productnumber	

Purchase order headers V2 to msdyn_purchaseorders

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
DELIVERYADDRESSNAME		msdyn_addressname	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
REQUESTEDDELIVERYDATE		`msdyn_dateexpected`	
PURCHASEORDERNUMBER		`msdyn_name`	
PAYMENTTERMSNAME		`msdyn_paymentterm.msdyn_name`	
DEFAULTRECEIVINGWAREHOUSEID		`msdyn_receivetowarehouse.msdyn_warehouseidentifier`	
ACCOUNTINGDATE		`msdyn_accountingdate`	
AREPRICESINCLUDINGSALESTAX		`msdyn_arepriceincludingsalestax`	
ATTENTIONINFORMATION		`msdyn_attentioninformation`	
CASHDISCOUNTCODE		`msdyn_cashdiscountcode`	
CASHDISCOUNTPERCENTAGE		`msdyn_cashdiscountpercentage`	
CONTACTPERSONID		`msdyn_contactpersonid.msdyn_contactpersonid`	
CURRENCYCODE		`transactioncurrencyid.isocurrencycode`	
DEFAULTRECEIVINGSITEID		`msdyn_defaultreceivingsiteid.msdyn_siteid`	
DELIVERYTERMSID		`msdyn_deliveryterm.msdyn_termscode`	
EMAIL		`msdyn_email`	
ISCHANGEMANAGEMENTACTIVE		`msdyn_ischangemanagementactive`	
ISDELIVEREDDIRECTLY		`msdyn_isdeliverydirectly`	
LANGUAGEID		`msdyn_language`	
ORDERERPERSONNELNUMBER		`msdyn_ordererpersonnelnumber.cdm_workernumber`	
REASONCODE		`msdyn_reasoncode`	
REASONCOMMENT		`msdyn_reasoncomment`	
TOTALDISCOUNTPERCENTAGE		`msdyn_totaldiscountpercentage`	
URL		`msdyn_url`	
VENDORORDERREFERENCE		`msdyn_vendororderreference`	
VENDORPAYMENTMETHODNAME		`msdyn_vendorpaymentmethod.msdyn_name`	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
VENDORPAYMENTMETHODSPECIFICATIONNAME		`msdyn_vendorpaymentmethodspecificationname`	
ORDERVENDORACCOUNTNUMBER		`msdyn_vendor.accountnumber`	
DELIVERYMODEID		`msdyn_shipvia.msdyn_name`	
INVOICEVENDORACCOUNTNUMBER		`msdyn_invoicevendoraccount.accountnumber`	
DOCUMENTAPPROVALSTATUS		`msdyn_documentapprovalstatus`	
PURCHASEORDERSTATUS		`msdyn_purchaseorderstatus`	
DELIVERYADDRESSCITY		`msdyn_city`	
DELIVERYADDRESSCOUNTRYREGIONISOCODE		`msdyn_country`	
DELIVERYADDRESSSTATEID		`msdyn_stateorprovince`	
DELIVERYADDRESSZIPCODE		`msdyn_postalcode`	
DELIVERYADDRESSSTREET		`msdyn_address1`	
INVOICEADDRESSSTREETNUMBER		`msdyn_invoiceaddressstreetnumber`	
CONFIRMEDDELIVERYDATE		`msdyn_confirmeddeliverydate`	
DELIVERYADDRESSLONGITUDE		`msdyn_longitude`	
DELIVERYADDRESSLATITUDE		`msdyn_latitude`	
DELIVERYADDRESSCOUNTRYREGIONID		`msdyn_deliveryaddresscountryregionid`	
DELIVERYADDRESSCOUNTYID		`msdyn_deliveryaddresscountyid`	
DELIVERYADDRESSDESCRIPTION		`msdyn_deliveryaddressdescription`	
DELIVERYADDRESSDISTRICTNAME		`msdyn_deliveryaddressdistrictname`	
DELIVERYADDRESSDUNSNUMBER		`msdyn_deliveryaddressdunsnumber`	
DELIVERYADDRESSLOCATIONID		`msdyn_deliveryaddresslocationid`	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
DELIVERYADDRESSPOSTBOX		`msdyn_deliveryaddresspostbox`	
DELIVERYADDRESSTIMEZONE		`msdyn_deliveryaddresstimezone`	
DELIVERYBUILDINGCOMPLIMENT		`msdyn_deliverybuildingcompliment`	
FORMATTEDDELIVERYADDRESS		`msdyn_formatteddeliveryaddress`	
FORMATTEDINVOICEADDRESS		`msdyn_formattedinvoiceaddress`	
INVOICEADDRESSCITY		`msdyn_invoiceaddresscity`	
INVOICEADDRESSCOUNTRYREGIONID		`msdyn_invoiceaddresscountryregionid`	
INVOICEADDRESSCOUNTY		`msdyn_invoiceaddresscounty`	
INVOICEADDRESSSTATE		`msdyn_invoiceaddressstate`	
INVOICEADDRESSSTREET		`msdyn_invoiceaddressstreet`	
DELIVERYADDRESSSTREETNUMBER		`msdyn_address2`	
INVOICEADDRESSZIPCODE		`msdyn_invoiceaddresszipcode`	

purchase order line soft deleted table to msdyn_purchaseorderproducts

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
LINENUMBER		`msdyn_lineorder`	
PURCHASEORDERNUMBER		`msdyn_purchaseorder.msdyn_name`	
ISDELETED		`msdyn_issoftdeletedinscm`	

purchase order line table to msdyn_purchaseorderproducts

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
REQUESTEDDELIVERYDATE		`msdyn_dateexpected`	
LINEDESCRIPTION		`msdyn_description`	
LINENUMBER		`msdyn_lineorder`	
PRODUCTNUMBER		`msdyn_product.msdyn_productnumber`	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
PURCHASEORDERNUMBER		`msdyn_purchaseorder.msdyn_name`	
ORDEREDPURCHASEQUANTITY		`msdyn_quantity`	
LINEAMOUNT		`msdyn_lineamount`	
PURCHASEPRICE		`msdyn_unitcost`	
BARCODE		`msdyn_barcode`	
CATCHWEIGHTUNITSYMBOL		`msdyn_catchweightunitsymbol.msdyn_symbol`	
CONFIRMEDSHIPPINGDATE		`msdyn_confirmedshippingdate`	
CUSTOMERREFERENCE		`msdyn_customerreference`	
CUSTOMERREQUISITIONNUMBER		`msdyn_customerrequisitionnumber`	
EXTERNALITEMNUMBER		`msdyn_externalitemnumber`	
ISPARTIALDELIVERYPREVENTED		`msdyn_ispartialdeliveryprevented`	
LINEDISCOUNTAMOUNT		`msdyn_linediscountamount`	
LINEDISCOUNTPERCENTAGE		`msdyn_linediscountpercentage`	
ORDEREDCATCHWEIGHTQUANTITY		`msdyn_orderedcatchweightquantity`	
PURCHASEORDERLINESTATUS		`msdyn_purchaseorderlinestatus`	
PURCHASEPRICEQUANTITY		`msdyn_purchasepricequantity`	
RECEIVINGSITEID		`msdyn_receivingsiteid.msdyn_siteid`	
REQUESTEDSHIPPINGDATE		`msdyn_requestedshippingdate`	
REQUESTERPERSONNELNUMBER		`msdyn_requesterpersonnelnumber.cdm_workernumber`	
PROCUREMENTPRODUCTCATEGORYNAME		`msdyn_procurementproductcategory.msdyn_name`	
PROCUREMENTPRODUCTCATEGORYHIERACHYNAME		`msdyn_procurementproductcategory.msdyn_hierarchy.msdyn_name`	
CURRENCYCODE		`transactioncurrencyid.isocurrencycode`	
CONFIRMEDDELIVERYDATE		`msdyn_confirmeddeliverydate`	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
DELIVERYADDRESSCITY		`msdyn_deliveryaddresscity`	
DELIVERYADDRESSCOUNTRYREGIONID		`msdyn_deliveryaddresscountryregionid`	
DELIVERYADDRESSSTATEID		`msdyn_deliveryaddressstate`	
DELIVERYADDRESSSTREET		`msdyn_deliveryaddressstreet`	
DELIVERYADDRESSSTREETNUMBER		`msdyn_deliveryaddressstreetnumber`	
DELIVERYADDRESSZIPCODE		`msdyn_deliveryaddresszipcode`	
FORMATTEDDELVERYADDRESS		`msdyn_formatteddeliveryaddress`	
PURCHASEUNITSYMBOL		`msdyn_unit.msdyn_symbol`	
RECEIVINGWAREHOUSEID		`msdyn_associatetowarehouse.msdyn_warehouseidentifier`	
DELIVERYADDRESSDESCRIPTION	>	`msdyn_deliveryaddressdescription`	
DELIVERYADDRESSNAME	>	`msdyn_deliveryaddressname`	

Sync on-demand with the Supply Chain Management pricing engine

2/18/2021 • 2 minutes to read • Edit Online

Microsoft Dynamics 365 Supply Chain Management includes a pricing engine that handles trade agreements, price lists, customer loyalty programs, promotions, and discounts. The pricing engine uses complex rules to determine the best price for a given quotation or order. When you use dual-write, you use either static pricing or the pricing engine from Dynamics 365 Supply Chain Management on the Quote and Order pages in Dynamics 365 Sales.

Use the pricing engine from Supply Chain Management in Sales

1. In Sales, go to **Sales > Orders**.
2. Select **New** to create a new order, or select an existing order in the **My Orders** list.
3. Add a new order line.
4. If you're creating a new order, select **Price Order** on the Action Pane. If you're updating an existing order, select **Recalculate** on the Action Pane.

   The following columns are automatically filled in:

   - Detail Amount
   - Discount %
   - Discount
   - Pre-Freight Amount
   - Freight Amount
   - Total Tax
   - Total Amount
5. To ensure that the system considers trade and sales agreements to calculate the price:

   a. Navigate to your Supply Chain Management environment.
   b. Navigate to **Accounts receivable > Setup > Accounts receivable parameters**.
   c. Select the **Prices** tab in the side navigation bar.
   d. Under the **Trade agreement evaluation** fastab, uncheck the **Manual entry** option.

How it works

When you select **Price Order** in Sales, the **Totals** function on the **Sales Order > View** tab in Supply Chain Management is called for the associated sales order. The values in the order total in Sales are used to fill in the corresponding columns in Supply Chain Management.

When the sales order total is calculated in Supply Chain Management, the calculation evaluates the existing trade agreements and sales agreements for the customer and the products that are listed in the sales order. This information is used to calculate the totals. When **Price Order** is selected, Sales automatically reflects all the setup that has been done in Supply Chain Management.

Limitations

When the columns in Sales are filled in, the following limitations apply:

- The setup of charges and charge allocations in Supply Chain Management isn't replicated in Sales.
- Pricing doesn't consider special retail pricing that is specified in the **Retail Channel** column on the sales order line page in Supply Chain Management.
- Discounts that are defined in the **Trade Allowance Management** section of Supply Chain Management aren't considered.

Prospect-to-cash in dual-write

2/18/2021 • 14 minutes to read • Edit Online

An important goal of most businesses is to convert prospects to customers and then maintain an ongoing business relationship with those customers. In Microsoft Dynamics 365 apps, the prospect-to-cash process occurs through quotations or order processing workflows, and the financials are reconciled and recognized. Integration of prospect-to-cash with dual-write creates a workflow that takes a quotation and an order that originate in either Dynamics 365 Sales or Dynamics 365 Supply Chain Management, and makes the quotation and order available in both apps.

In the app interfaces, you can access the processing statuses and invoice information in real time. Therefore, you can more easily manage functions such as product stocking, inventory handling, and fulfillment in Supply Chain Management, without having to re-create the quotations and orders.

For information about customer and contact integration, see Integrated customer master. For information about product integration, see Unified product experience.

> **NOTE**
> In Dynamics 365 Sales, both prospect and customer refer to a record in the **Account** table where the **RelationshipType** column is either **Prospect** or **Customer**. If your business logic includes an **Account** qualification process where the **Account** record is created and qualified as a prospect first and then as a customer, that record synchronizes to the Finance and Operations app only when it is a customer (`RelationshipType=Customer`). If you want the **Account** row to synchronize as a prospect, then you need a custom map to integrate the prospect data.

Prerequisites and mapping setup

Before you can sync sales quotations, you must update the following settings.

Setup in Sales

In Sales, go to **Settings** > **Administration** > **System settings** > **Sales**, and make sure that the following settings are used:

- The **Use system prizing calculation** system option is set to **Yes**.
- The **Discount calculation method** column is set to **Line item**.

Sites and warehouses

In Supply Chain Management, the **Site** and **warehouse** columns are required for quotation lines and order lines. If you set the site and warehouse in the default order settings, those columns will automatically be set when you add a product to a quotation line or an order line.

Number sequences for quotations and orders

The number sequences for Supply Chain Management and Sales aren't connected when quotations and orders are created and synced in Sales and Supply Chain Management. If a sales order that is created in Sales is synced to Supply Chain Management, it has the same sales order number in Supply Chain Management. To help ensure that the sales order number isn't duplicated, you must use different number sequence systems in the two apps.

For example, the number sequence in Supply Chain Management is **1, 2, 3, 4, 5, ...**, and the number sequence in Sales is **100, 99, 98, ...**. If you create 100 sales orders in Sales, an order number will eventually be generated

that already exists in Supply Chain Management. In other words, the two number sequences will eventually overlap as sales orders are created in Supply Chain Management and Sales. Instead, you might use a number sequence such as **F1, F2, F3, ...** in Supply Chain Management and a number sequence such as **C1, C2, C3, ...** in Sales. These number sequences will never produce duplicate sales order numbers.

Sales quotations

Sales quotations can be created in either Sales or Supply Chain Management. If you create a quotation in Sales, it's synced to Supply Chain Management in real time. Likewise, if you create a quotation in Supply Chain Management, it's synced to Sales in real time. Note the following points:

- You can add a discount to the product on the quotation. In this case, the discount will be synced to Supply Chain Management. The **Discount**, **Charges**, and **Tax** columns on the header are controlled by a setup in Supply Chain Management. This setup doesn't support integration mapping. Instead, the **Price**, **Discount**, **Charge**, and **Tax** columns are maintained and handled in Supply Chain Management.
- The **Discount %**, **Discount**, and **Freight Amount** columns on the sales quotation header are read-only columns.
- The **Freight terms**, **Delivery terms**, **Shipping method**, and **Delivery mode** columns aren't part of the default mappings. To map these columns, you must set up a value mapping that is specific to the data in the organizations that the table is synced between.

If you are also using the Field Service solution, make sure to re-enable the **Quote Line Quick Create** parameter. Re-enabling the parameter lets you continue creating quote lines using the quick create function.

1. Navigate to your Dynamics 365 Sales application.
2. Select the settings icon in the top navigation bar.
3. Select **Advanced Settings**.
4. Choose the **Customize the System** option.
5. Select the **Quote Line** menu item.
6. Go to the **Data Services** section and select the **Allow quick create** checkbox.

Sales orders

Sales orders can be created in either Sales or Supply Chain Management. If you create a sales order in Sales, it's synced to Supply Chain Management in real time. Likewise, if you create a sales order in Supply Chain Management, it's synced to Sales in real time. Note the following points:

- Write-in products on Dynamics 365 Sales will appear as product categories in Dynamics 365 Supply Chain Management.
- Discount calculation and rounding:
    - The discount calculation model in Sales differs from the discount calculation model in Supply Chain Management. In Supply Chain Management, the final discount amount on a sales line can be the result of a combination of discount amounts and discount percentages. If this final discount amount is divided by the quantity on the line, rounding can occur. However, this rounding isn't considered if a rounded per-unit discount amount is synced to Sales. To help ensure that the full discount amount from a sales line in Supply Chain Management is correctly synced to Sales, the full amount must be synced without being divided by the line quantity. Therefore, you must define the discount calculation method as **Line item** in Sales.
    - When a sales order line is synced from Sales to Supply Chain Management, the full line discount amount is used. Because Supply Chain Management has no column that can store the full discount amount for a line, the amount is divided by the quantity and stored in the **Line discount** column. Any rounding that occurs during this division is stored in the **Sales charges** column on the sales line.

Example: Synchronization from Sales to Supply Chain Management

You have the following sales order:

- **Sales:** Quantity = 3, per-line discount = $10.00
- **Supply Chain Management:** Quantity = 3, line discount amount = $3.33, sales charge = –$0.01

If you sync from Supply Chain Management to Sales, you get the following result:

- **Supply Chain Management:** Quantity = 3, line discount amount = $3.33, sales charge = –$0.01
- **Sales:** Quantity = 3, per-line discount = (3 × $3.33) + $0.01 = $10.00

Dual-write solution for Sales

New columns have been added to the **Order** table and appear on the page. Most of these columns appear on the **Integration** tab in Sales. To learn more about how the status columns are mapped, see Set up the mapping for sales order status columns.

- The **Create Invoice** and **Cancel Order** buttons on the **Sales order** page are hidden in Sales.
- The **Sales order status** value will remain **Active** to help ensure that changes from Supply Chain Management can flow to the sales order in Sales. To control this behavior, set the default **Statecode [Status]** value to **Active**.

Invoices

Sales invoices are created in Supply Chain Management and synced to Sales. Note the following points:

- An **Invoice number** column has been added to the **Invoice** table and appears on the page.
- The **Create invoice** button on the **Sales order** page is hidden, because invoices will be created in Supply Chain Management and synced to Sales. The **Invoice** page can't be edited, because invoices will be synced from Supply Chain Management.
- The **Sales order status** value is automatically changed to **Invoiced** when the related invoice from Supply Chain Management has been synced to Sales. Additionally, the owner of the sales order that the invoice was created from is assigned as the owner of the invoice. Therefore, the owner of the sales order can view the invoice.
- The **Freight terms**, **Delivery terms**, and **Delivery mode** columns aren't included in the default mappings. To map these columns, you must set up a value mapping that is specific to the data in the organizations that the table is synced between.

Templates

Prospect-to-cash includes a collection of core table maps that work together during data interaction, as shown in the following table.

FINANCE AND OPERATIONS APPS	CUSTOMER ENGAGEMENT APPS	DESCRIPTION
Sales invoice headers V2	invoices	The Sales invoice headers V2 table in the Finance and Operations app contains invoices for sales orders and free text invoices. A filter is applied in Dataverse for dual-write that will filter out any free text invoice documents.
Sales invoice lines V2	invoicedetails	
CDS sales order headers	salesorders	

FINANCE AND OPERATIONS APPS	CUSTOMER ENGAGEMENT APPS	DESCRIPTION
CDS sales order lines	salesorderdetails	
Sales order origin codes	msdyn_salesorderorigins	
CDS sales quotation header	quotes	
CDS sales quotation lines	quotedetails	

Here are the related core table maps for prospect-to-cash:

- Customers V3 to accounts
- CDS Contacts V2 to contacts
- Customers V3 to contacts
- Released products V2 to msdyn_sharedproductdetails
- All products to msdyn_globalproducts
- Pricelist

Limitations

- Return orders are not supported.
- Credit notes are not supported.
- Financial dimensions must be set for the master data, for example, customer and vendor. When a customer is added to a quotation or sales order, the financial dimensions associated with the customer record flow to the order automatically. Currently dual-write does not include financial dimensions data for master data.

Mapping tables

Mapping types

There are several different mapping types. The following table explains the symbols used in the template tables.

SYMBOL	DESCRIPTION
>	One-way
>>	One-way, and data is transformed in the process.
=	Bidirectional
><	Bidirectional, and data is transformed in the process.
<<	One-way, and data is transformed in the process.

Filters

The source filter and reverse source filter determine which rows are synchronized.

Default values

If a synchronized field does not exist in either the Finance and Operations table or the other Dynamics 365 table, then a default value is assigned in the synchronized table. In some cases, the default value is an integer that is a

lookup to an attribute value in the Common Data Model. For example, in the Contacts table of the Common Data Model, the default value of **address1_addresstypecode** is **3**. In the Common Data Model, for address1AddressTypeCode the value of **3** is **Primary address**.

Sales invoice headers V2 to invoices

This template synchronizes data between Finance and Operations apps and Dataverse.

Source filter: `(SalesOrderNumber != "")`

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
none	>>	ispricelocked	False
none	>>	statuscode	4
CONTACTPERSONID	>>	msdyn_contactperson.msdyn_contactpersonid	
CURRENCYCODE	>>	transactioncurrencyid.isocurrencycode	
CUSTOMERSORDERREFERENCE	>>	description	
INVOICEADDRESSCITY	>>	billto_city	
INVOICEADDRESSCOUNTRYREGIONISOCODE	>>	billto_country	
INVOICEADDRESSSTATE	>>	billto_stateorprovince	
INVOICEADDRESSSTREET	>>	billto_line1	
INVOICEADDRESSSTREETNUMBER	>>	billto_line2	
INVOICEADDRESSZIPCODE	>>	billto_postalcode	
INVOICECUSTOMERACCOUNTNUMBER	>>	customerid.Account(accountnumber).Contact(msdyn_contactpersonid)	
INVOICEDATE	>>	msdyn_invoicedate	
INVOICENUMBER	>>	msdyn_invoicenumber	
LEDGERVOUCHER	>>	msdyn_ledgervoucher	
PAYMENTTERMSNAME	>>	msdyn_paymentterms.msdyn_name	
SALESORDERNUMBER	>>	salesorderid.msdyn_salesordernumber	

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
TOTALCHARGEAMOUNT	>>	freightamount	
TOTALDISCOUNTAMOUNT	>>	totaldiscountamount	
TOTALDISCOUNTCUSTOMERGROUPCODE	>>	discountamount	
TOTALINVOICEAMOUNT	>>	totalamount	
TOTALTAXAMOUNT	>>	totaltax	

Sales invoice lines V2 to invoicedetails

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
CONFIRMEDSHIPPINGDATE	>>	msdyn_confirmedshippingdate	
CURRENCYCODE	>>	transactioncurrencyid.isocurrencycode	
INVENTORYSITEID	>>	msdyn_inventorysite.msdyn_siteid	
INVENTORYWAREHOUSEID	>>	msdyn_inventorywarehouse.msdyn_warehouseidentifier	
INVOICEDATE	>>	invoiceid.msdyn_invoicedate	
INVOICEDQUANTITY	>>	quantity	
INVOICENUMBER	>>	invoiceid.msdyn_invoicenumber	
LEDGERVOUCHER	>>	invoiceid.msdyn_ledgervoucher	
LINEAMOUNT	>>	extendedamount	
LINECREATIONSEQUENCENUMBER	>>	sequencenumber	
LINETOTALCHARGEAMOUNT	>>	msdyn_totalchargeamount	
LINETOTALDISCOUNTAMOUNT	>>	manualdiscountamount	

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
LINETOTALTAXAMOUNT	>>	tax	
PRODUCTNAME	>>	description	
PRODUCTNUMBER	>>	productid.msdyn_productnumber	
SALESPRICE	>>	priceperunit	
SALESPRODUCTCATEGORYHIERARCHYNAME	>>	msdyn_salesproductcategory.msdyn_hierarchy.msdyn_name	
SALESPRODUCTCATEGORYNAME	>>	msdyn_salesproductcategory.msdyn_name	
SALESUNITSYMBOL	>>	uomid.msdyn_symbol	
none	>>	producttypecode	1
none	>>	propertyconfigurationstatus	2
none	>>	ispriceoverridden	True

CDS sales order headers to salesorders

This template synchronizes data between Finance and Operations apps and Dataverse.

Reversed source filter: msdyn_ordertype eq 192350000

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
SALESORDERNUMBER	><	msdyn_salesordernumber	
ORDERINGCUSTOMERACCOUNTNUMBER	><	customerid.Account(accountnumber).Contact(msdyn_contactpersonid)	
CURRENCYCODE	><	transactioncurrencyid.isocurrencycode	
DELIVERYADDRESSCITY	><	shipto_city	
DELIVERYADDRESSCOUNTRYREGIONISOCODE	><	shipto_country	
DELIVERYADDRESSSTREETNUMBER	><	shipto_line2	
DELIVERYADDRESSZIPCODE	><	shipto_postalcode	

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
DELIVERYADDRESSSTREET	> <	shipto_line1	
DELIVERYADDRESSSTATEID	> <	shipto_stateorprovince	
SALESORDERNAME	> >	name	
INVOICEADDRESSCITY	> >	billto_city	
INVOICEADDRESSSTREET	> >	billto_line1	
INVOICEADDRESSSTREETNUMBER	> >	billto_line2	
INVOICEADDRESSCOUNTRYREGIONISOCODE	> >	billto_country	
INVOICEADDRESSSTATEID	> >	billto_stateorprovince	
INVOICEADDRESSZIPCODE	> >	billto_postalcode	
ORDERTOTALAMOUNT	> >	totalamount	
TOTALDISCOUNTAMOUNT	> >	discountamount	
ORDERTOTALTAXAMOUNT	> >	totaltax	
ORDERTOTALCHARGESAMOUNT	> >	freightamount	
AREPRICESINCLUDINGSALESTAX	> <	msdyn_arepricesincludingsalestax	
CONFIRMEDRECEIPTDATE	=	msdyn_confirmedreceiptdate	
CONFIRMEDSHIPPINGDATE	=	msdyn_confirmedshippingdate	
CONTACTPERSONID	=	msdyn_contactperson.msdyn_contactpersonid	
CUSTOMERREQUISITIONNUMBER	=	msdyn_customerrequisitionnumber	
DEFAULTSHIPPINGSITEID	=	msdyn_defaultshippingsite.msdyn_siteid	
DEFAULTSHIPPINGWAREHOUSEID	=	msdyn_defaultshippingwarehouse.msdyn_warehouseidentifier	

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
DELIVERYADDRESSCOUNTYID	=	msdyn_deliveryaddresscountyid	
DELIVERYADDRESSDESCRIPTION	=	msdyn_deliveryaddressdescription	
DELIVERYADDRESSDISTRICTNAME	=	msdyn_deliveryaddressdistrictname	
DELIVERYADDRESSDUNSNUMBER	=	msdyn_deliveryaddressdunsnumber	
DELIVERYADDRESSLATITUDE	=	msdyn_deliveryaddresslatitude	
DELIVERYADDRESSLOCATIONID	=	msdyn_deliveryaddresslocationid	
DELIVERYADDRESSLONGITUDE	=	msdyn_deliveryaddresslongitude	
DELIVERYADDRESSNAME	=	msdyn_deliveryaddressname	
DELIVERYADDRESSPOSTBOX	=	msdyn_deliveryaddresspostbox	
DELIVERYBUILDINGCOMPLIMENT	=	msdyn_deliverybuildingcompliment	
FORMATTEDDELVERYADDRESS	>>	msdyn_formatteddeliveryaddress	
EMAIL	=	emailaddress	
FORMATTEDINVOICEADDRESS	>>	msdyn_formattedinvoiceaddress	
INVOICEADDRESSCOUNTYID	>>	msdyn_invoiceaddresscountyid	
INVOICEADDRESSDISTRICTNAME	>>	msdyn_invoiceaddressdistrictname	
INVOICEADDRESSLATITUDE	>>	msdyn_invoiceaddresslatitude	
INVOICEADDRESSLONGITUDE	>>	msdyn_invoiceaddresslongitude	
INVOICEADDRESSPOSTBOX	>>	msdyn_invoiceaddresspostbox	

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
INVOICEBUILDINGCOMPLIMENT	> >	msdyn_invoicebuildingcompliment	
INVOICECUSTOMERACCOUNTNUMBER	=	msdyn_invoicecustomer.accountnumber	
ISDELIVERYADDRESSORDERSPECIFIC	> <	msdyn_isdeliveryaddressorderspecific	
ISDELIVERYADDRESSPRIVATE	> <	msdyn_isdeliveryaddressprivate	
ISINVOICEADDRESSPRIVATE	> >	msdyn_isinvoiceaddressprivate	
ISONETIMECUSTOMER	> <	msdyn_isonetimecustomer	
ISSALESPROCESSINGSTOPPED	> <	msdyn_issalesprocessingstopped	
PAYMENTTERMSBASEDATE	=	msdyn_paymenttermsbasedate	
PAYMENTTERMSNAME	=	msdyn_paymentterms.msdyn_name	
PRICECUSTOMERGROUPCODE	=	msdyn_pricecustomergroup.msdyn_groupcode	
QUOTATIONNUMBER	=	msdyn_quotationnumber	
REQUESTEDRECEIPTDATE	=	msdyn_requestedreceiptdate	
REQUESTEDSHIPPINGDATE	=	requestdeliveryby	
SALESORDERPROMISINGMETHOD	> <	msdyn_salesorderpromisingmethod	
URL	=	msdyn_url	
none	> >	statecode	0
none	> >	statuscode	1
SALESORDERPROCESSINGSTATUS	> <	msdyn_processingstatus	
LANGUAGEID	> <	msdyn_language	
CUSTOMERSORDERREFERENCE	> <	msdyn_customersorderreference	

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
DELIVERYMODECODE	=	msdyn_deliverymode.msdyn_name	
DELIVERYTERMSCODE	=	msdyn_deliveryterms.msdyn_termscode	
SALESORDERORIGINCODE	=	msdyn_salesorderorigin.msdyn_origincode	

CDS sales order lines to salesorderdetails

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
CURRENCYCODE	> >	transactioncurrencyid.isocurrencycode	
DELIVERYADDRESSCITY	> <	shipto_city	
DELIVERYADDRESSCOUNTRYREGIONISOCODE	> <	shipto_country	
DELIVERYADDRESSZIPCODE	> <	shipto_postalcode	
DELIVERYADDRESSSTATEID	> <	shipto_stateorprovince	
DELIVERYADDRESSSTREET	> <	shipto_line1	
DELIVERYADDRESSSTREETNUMBER	> <	shipto_line2	
LINEAMOUNT	> >	extendedamount	
LINECREATIONSEQUENCENUMBER	> <	sequencenumber	
ORDEREDSALESQUANTITY	> <	quantity	
PRODUCTNAME	> <	description	
PRODUCTNUMBER	> <	productid.msdyn_productnumber	
SALESORDERNUMBER	> <	salesorderid.msdyn_salesordernumber	
SALESUNITSYMBOL	> <	uomid.msdyn_symbol	

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
LINEDISCOUNTAMOUNT	><	manualdiscountamount	
TOTALTAXAMOUNT	>>	tax	
SALESPRICE	><	priceperunit	
LINEAMOUNT	>>	baseamount	
SALESPRODUCTCATEGORYNAME	=	msdyn_salesproductcategory.msdyn_name	
SALESPRODUCTCATEGORYHIERARCHYNAME	>>	msdyn_salesproductcategory.msdyn_hierarchy.msdyn_name	
ISDELIVERYADDRESSORDERSPECIFIC	><	msdyn_isdeliveryaddressspecific	
ISDELIVERYADDRESSPRIVATE	><	msdyn_isdeliveryaddressprivate	
ISLINESTOPPED	><	msdyn_islinestopped	
ALLOWEDOVERDELIVERYPERCENTAGE	=	msdyn_allowedoverdeliverypercentage	
ALLOWEDUNDERDELIVERYPERCENTAGE	=	msdyn_allowedunderdeliverypercentage	
CONFIRMEDSHIPPINGDATE	=	msdyn_confirmedshippingdate	
CONFIRMEDRECEIPTDATE	=	msdyn_confirmedreceiptdate	
DELIVERYADDRESSCOUNTYID	=	msdyn_deliveryaddresscountyid	
DELIVERYADDRESSDESCRIPTION	=	msdyn_deliveryaddressdescription	
DELIVERYADDRESSDISTRICTNAME	=	msdyn_deliveryaddressdistrictname	
DELIVERYADDRESSDUNSNUMBER	=	msdyn_deliveryaddressdunsnumber	
DELIVERYADDRESSLATITUDE	=	msdyn_deliveryaddresslatitude	
DELIVERYADDRESSLOCATIONID	=	msdyn_deliveryaddresslocationid	

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
DELIVERYADDRESSLONGITUDE	=	msdyn_deliveryaddresslongitude	
DELIVERYADDRESSNAME	=	msdyn_deliveryaddressname	
DELIVERYADDRESSPOSTBOX	=	msdyn_deliveryaddresspostbox	
DELIVERYBUILDINGCOMPLIMENT	=	msdyn_deliverybuildingcompliment	
EXTERNALITEMNUMBER	=	msdyn_externalitemnumber	
FIXEDPRICECHARGES	=	msdyn_fixedpricecharges	
FORMATTEDDELIVERYADDRESS	=	msdyn_formatteddeliveryaddress	
LINEDESCRIPTION	=	msdyn_linedescription	
LINEDISCOUNTAMOUNT	><	msdyn_linediscountamount	
LINEDISCOUNTPERCENTAGE	><	msdyn_linediscountpercentage	
MULTILINEDISCOUNTAMOUNT	><	msdyn_multilinediscountamount	
MULTILINEDISCOUNTPERCENTAGE	><	msdyn_multilinediscountpercentage	
REQUESTEDRECEIPTDATE	=	msdyn_requestedreceiptdate	
REQUESTEDSHIPPINGDATE	=	requestdeliveryby	
SALESORDERLINESTATUS	>>	msdyn_linestatus	
SALESORDERPROMISINGMETHOD	><	msdyn_salesorderpromisingmethod	
SALESPRICEQUANTITY	=	msdyn_salespricequantity	
SHIPPINGSITEID	=	msdyn_shippingsite.msdyn_siteid	
SHIPPINGWAREHOUSEID	=	msdyn_shippingwarehouse.msdyn_warehouseidentifier	
none	>>	ispriceoverridden	true

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
TOTALCHARGESAMOUNT	>>	msdyn_totalchargesamount	
DELIVERYMODECODE	=	msdyn_deliverymode.msdyn_name	
DELIVERYTERMSID	=	msdyn_deliveryterms.msdyn_termscode	

Sales order origin codes to msdyn_salesorderorigins

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
ORIGINCODE	=	msdyn_origincode	
ORIGINDESCRIPTION	=	msdyn_origindescription	

CDS sales quotation header to quotes

This template synchronizes data between Finance and Operations apps and Dataverse.

Reversed source filter: statecode eq 0

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
SALESQUOTATIONNUMBER	=	msdyn_quotenumber	
REQUESTINGCUSTOMERACCOUNTNUMBER	=	customerid.Account(accountnumber).Contact(msdyn_contactpersonid)	
CURRENCYCODE	=	transactioncurrencyid.isocurrencycode	
CUSTOMERSREFERENCE	=	msdyn_customersreference	
DELIVERYADDRESSCITY	=	shipto_city	
DELIVERYADDRESSCOUNTRYREGIONISOCODE	><	shipto_country	
DELIVERYADDRESSSTREETNUMBER	=	shipto_line2	
DELIVERYADDRESSZIPCODE	=	shipto_postalcode	
DELIVERYADDRESSSTREET	=	shipto_line1	

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
DELIVERYADDRESSSTATEID	=	shipto_stateorprovince	
SALESQUOTATIONNAME	=	name	
INVOICEADDRESSCITY	>>	billto_city	
INVOICEADDRESSSTREET	>>	billto_line1	
INVOICEADDRESSSTREETNUMBER	>>	billto_line2	
INVOICEADDRESSCOUNTRYREGIONISOCODE	>>	billto_country	
INVOICEADDRESSSTATEID	>>	billto_stateorprovince	
INVOICEADDRESSZIPCODE	>>	billto_postalcode	
QUOTATIONTOTALAMOUNT	>>	totalamount	
TOTALDISCOUNTAMOUNT	>>	discountamount	
QUOTATIONTOTALTAXAMOUNT	>>	totaltax	
QUOTATIONTOTALCHARGESAMOUNT	>>	freightamount	
AREPRICESINCLUDINGSALESTAX	><	msdyn_arepricesincludingsalestax	
CONTACTPERSONID	=	msdyn_contactperson.msdyn_contactpersonid	
CUSTOMERREQUISITIONNUMBER	=	msdyn_customerrequisitionnumber	
DEFAULTSHIPPINGSITEID	=	msdyn_defaultshippingsite.msdyn_siteid	
DEFAULTSHIPPINGWAREHOUSEID	=	msdyn_defaultshippingwarehouse.msdyn_warehouseidentifier	
DELIVERYADDRESSCOUNTYID	=	msdyn_deliveryaddresscountyid	
DELIVERYADDRESSDESCRIPTION	=	msdyn_deliveryaddressdescription	

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
DELIVERYADDRESSDISTRICTNAME	=	msdyn_deliveryaddressdistrictname	
DELIVERYADDRESSDUNSNUMBER	=	msdyn_deliveryaddressdunsnumber	
DELIVERYADDRESSLATITUDE	=	msdyn_deliveryaddresslatitude	
DELIVERYADDRESSLOCATIONID	=	msdyn_deliveryaddresslocationid	
DELIVERYADDRESSLONGITUDE	=	msdyn_deliveryaddresslongitude	
DELIVERYADDRESSNAME	=	msdyn_deliveryaddressname	
DELIVERYADDRESSPOSTBOX	=	msdyn_deliveryaddresspostbox	
DELIVERYBUILDINGCOMPLIMENT	=	msdyn_deliverybuildingcompliment	
FORMATTEDDELIVERYADDRESS	>>	msdyn_formatteddeliveryaddress	
FORMATTEDINVOICEADDRESS	>>	msdyn_formattedinvoiceaddress	
GENERATEDSALESORDERNUMBER	=	msdyn_generatedsalesordernumber.msdyn_salesordernumber	
INVOICEADDRESSCOUNTRYREGIONID	>	msdyn_invoiceaddresscountryregionid	
INVOICEADDRESSCOUNTYID	>>	msdyn_invoiceaddresscountyid	
INVOICEADDRESSDISTRICTNAME	>>	msdyn_invoiceaddressdistrictname	
INVOICEADDRESSLATITUDE	>>	msdyn_invoiceaddresslatitude	
INVOICEADDRESSLONGITUDE	>>	msdyn_invoiceaddresslongitude	
INVOICEADDRESSPOSTBOX	>>	msdyn_invoiceaddresspostbox	

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
INVOICEBUILDINGCOMPLIMENT	> >	msdyn_invoicebuildingcompliment	
INVOICECUSTOMERACCOUNTNUMBER	=	msdyn_invoicecustomer.accountnumber	
ISDELIVERYADDRESSORDERSPECIFIC	> <	msdyn_isdeliveryaddressorderspecific	
ISDELIVERYADDRESSPRIVATE	> <	msdyn_isdeliveryaddressprivate	
ISINVOICEADDRESSPRIVATE	> >	msdyn_isinvoiceaddressprivate	
LANGUAGEID	> <	msdyn_language	
PAYMENTTERMSNAME	=	msdyn_paymentterms.msdyn_name	
PRICECUSTOMERGROUPCODE	=	msdyn_pricecustomergroup.msdyn_groupcode	
RECEIPTDATEREQUESTED	=	msdyn_requestedreceiptdate	
REQUESTEDSHIPPINGDATE	=	requestdeliveryby	
SALESORDERPROMISINGMETHOD	> <	msdyn_salesorderpromisingmethod	
SALESQUOTATIONCONFIRMATIONDATE	=	msdyn_salesquotationconfirmationdate	
SALESQUOTATIONEXPIRYDATE	=	msdyn_salesquotationexpirydate	
SALESQUOTATIONFOLLOWUPDATE	=	msdyn_salesquotationfollowupdate	
SALESQUOTATIONSTATUS	> <	msdyn_salesquotationstatus	
TOTALDISCOUNTPERCENTAGE	=	msdyn_totaldiscountpercentage	
URL	=	msdyn_url	
EMAIL	=	emailaddress	

CDS sales quotation lines to quotedetails

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
ALLOWEDOVERDELIVERYPERCENTAGE	=	msdyn_allowedoverdeliverypercentage	
ALLOWEDUNDERDELIVERYPERCENTAGE	=	msdyn_allowedunderdeliverypercentage	
SALESQUOTATIONNUMBER	=	quoteid.msdyn_quotenumber	
LINECREATIONSEQUENCENUMBER	=	sequencenumber	
CURRENCYCODE	>>	transactioncurrencyid.isocurrencycode	
DELIVERYADDRESSCITY	=	shipto_city	
DELIVERYADDRESSCOUNTRYREGIONISOCODE	><	shipto_country	
DELIVERYADDRESSCOUNTYID	=	msdyn_deliveryaddresscountyid	
DELIVERYADDRESSDESCRIPTION	=	msdyn_deliveryaddressdescription	
DELIVERYADDRESSDISTRICTNAME	=	msdyn_deliveryaddressdistrictname	
DELIVERYADDRESSDUNSNUMBER	=	msdyn_deliveryaddressdunsnumber	
DELIVERYADDRESSLATITUDE	=	msdyn_deliveryaddresslatitude	
DELIVERYADDRESSLOCATIONID	=	msdyn_deliveryaddresslocationid	
DELIVERYADDRESSLONGITUDE	=	msdyn_deliveryaddresslongitude	
DELIVERYADDRESSNAME	=	msdyn_deliveryaddressname	
DELIVERYADDRESSPOSTBOX	=	msdyn_deliveryaddresspostbox	
DELIVERYADDRESSSTATEID	=	shipto_stateorprovince	
DELIVERYADDRESSSTREET	=	shipto_line1	

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
DELIVERYADDRESSSTREETNUMBER	=	shipto_line2	
DELIVERYADDRESSZIPCODE	=	shipto_postalcode	
DELIVERYBUILDINGCOMPLIMENT	=	msdyn_deliverybuildingcompliment	
EXTERNALITEMNUMBER	=	msdyn_externalitemnumber	
FIXEDPRICECHARGES	=	msdyn_fixedpricecharges	
FORMATTEDDELIVERYADDRESS	=	msdyn_formatteddeliveryaddress	
ISDELIVERYADDRESSPRIVATE	><	msdyn_isdeliveryaddressprivate	
ISDELIVERYADDRESSORDERSPECIFIC	><	msdyn_isdeliveryaddressspecific	
LINEAMOUNT	>>	baseamount	
LINEAMOUNT	>>	extendedamount	
LINEDESCRIPTION	=	msdyn_linedescription2	
LINEDISCOUNTAMOUNT	=	msdyn_linediscountamount	
LINEDISCOUNTPERCENTAGE	=	msdyn_linediscountpercentage	
MULTILINEDISCOUNTAMOUNT	=	msdyn_multilinediscountamount	
MULTILINEDISCOUNTPERCENTAGE	=	msdyn_multilinediscountpercentage	
PRODUCTNAME	=	description	
PRODUCTNUMBER	=	productid.msdyn_productnumber	
REQUESTEDRECEIPTDATE	=	msdyn_requestedreceiptdate	
REQUESTEDSALESQUANTITY	=	quantity	
REQUESTEDSHIPPINGDATE	=	requestdeliveryby	

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
SALESPRICE	=	priceperunit	
SALESPRICEQUANTITY	=	msdyn_salespricequantity	
SALESQUOTATIONPROMISINGMETHOD	><	msdyn_salesquotationpromisingmethod	
SALESQUOTATIONSTATUS	><	msdyn_salesquotationstatus	
SALESUNITSYMBOL	=	uomid.msdyn_symbol	
SHIPPINGSITEID	=	msdyn_shippingsite.msdyn_siteid	
SHIPPINGWAREHOUSEID	=	msdyn_shippingwarehouse.msdyn_warehouseidentifier	
TOTALCHARGESAMOUNT	>>	msdyn_totalchargesamount	
TOTALDISCOUNTAMOUNT	=	manualdiscountamount	
TOTALTAXAMOUNT	>>	tax	
SALESPRODUCTCATEGORYHIERARCHYNAME	>	msdyn_salesproductcategory.msdyn_hierarchy.msdyn_name	
SALESPRODUCTCATEGORYNAME	=	msdyn_salesproductcategory.msdyn_name	
none	>>	ispriceoverridden	true

Use the Dynamics 365 Commerce pricing engine with Dynamics 365 Sales

2/18/2021 • 2 minutes to read • Edit Online

This topic describes how to use the Microsoft Dynamics 365 Commerce pricing engine to create sales quotations in Dynamics 365 Sales.

The Dynamics 365 Commerce pricing engine supports most business-to-consumer (B2C) pricing scenarios, such as store-level pricing, affiliation-based and loyalty-based pricing, mix-and-match discounts, quantity discounts, and threshold discounts. The pricing engine uses complex rules to determine the best price for a given quotation or order.

When you use dual-write, you have three options for your pricing needs. You can use the static pricing that comes from the price list in Dynamics 365 Sales, the pricing engine in Dynamics 365 Supply Chain Management, or the pricing engine in Dynamics 365 Commerce. Among these options, the Commerce pricing engine is best suited to B2C scenarios.

Use the Commerce pricing engine in Sales

> **NOTE**
> Currently, the Commerce pricing engine can be used only for quotation creation in the Sales. Sales order integration with the Commerce pricing engine isn't yet available.

When users initiate a quotation in Sales, the dual-write framework copies the quotation details to Commerce. Any changes to existing quotation lines or any newly added quotation lines in Sales are copied to Commerce. When users want to use the Commerce pricing engine to price the quotation, they can select **Price quote** to update the prices of the quotation, based on the Commerce pricing engine. Prices are then automatically updated in both Sales and Commerce.

Prerequisites

- Before you can use the Commerce pricing engine in Sales, you must follow the steps in Prospect-to-cash in dual-write.
- You must turn off trade agreement evaluation for manual entry by following these steps:
  1. In your Commerce environment, go to **Accounts receivable** > **Setup** > **Accounts receivable parameters**.
  2. On the **Prices** tab, on the **Trade agreement evaluation** FastTab, clear the **Manual entry** check box.

Additional resources

Prospect-to-cash in dual-write

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

In-house assets for servicing

2/18/2021 • 4 minutes to read • Edit Online

Microsoft Dynamics 365 Field Service is designed to service customer assets. Asset management for Dynamics 365 Supply Chain Management is designed to maintain in-house assets. Integration of these two apps lets you use Field Service to service both customer assets and in-house assets. You can also classify the assets, based on functional location or hierarchy, and track the servicing at a detailed level.

For more information, see Integrate Dynamics 365 Field Service and Supply Chain Management.

Templates

In-house-assets include a collection of core table maps that work together during data interaction, as shown in the following table.

FINANCE AND OPERATIONS APPS	MODEL-DRIVEN APPS IN DYNAMICS 365	DESCRIPTION
Asset management asset lifecycle models	msdyn_assetlifecyclemodels	
Asset management asset lifecycle states	msdyn_assetlifecyclestates	
Asset management assets	msdyn_customerassets	
Asset management asset types	msdyn_customerassetcategories	
Asset management functional location lifecycle models	msdyn_functionallocationlifecyclemodels	
Asset management functional location lifecycle states	msdyn_functionallocationlifecyclestates	
Asset management functional locations	msdyn_functionallocations	
Asset management functional location types	msdyn_functionallocationtypes	
Asset management manufacturers	msdyn_manufacturers	
Asset management models	msdyn_models	
Asset management warranty	msdyn_warranties	

Mapping tables

Mapping types

There are several different mapping types. The following table explains the symbols used in the template tables.

SYMBOL	DESCRIPTION
>	One-way
> >	One-way, and data is transformed in the process.
=	Bidirectional
> <	Bidirectional, and data is transformed in the process.
< <	One-way, and data is transformed in the process.

Filters

The source filter and reverse source filter determine which rows are synchronized.

Default values

If a synchronized field does not exist in either the Finance and Operations table or the other Dynamics 365 table, then a default value is assigned in the synchronized table. In some cases, the default value is an integer that is a lookup to an attribute value in the Common Data Model. For example, in the Contacts table of the Common Data Model, the default value of **address1_addresstypecode** is **3**. In the Common Data Model, for address1AddressTypeCode the value of **3** is **Primary address**.

Asset management asset lifecycle models to msdyn_assetlifecyclemodels

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
ACTIVELIFECYCLESTATEID	=	msdyn_activelifecyclestate.msdyn_assetlifecyclestate_id	
INBOUNDLIFECYCLESTATEID	=	msdyn_inboundlifecyclestate.msdyn_assetlifecyclestate_id	
INSTORAGELIFECYCLESTATEID	=	msdyn_instoragelifecyclestate.msdyn_assetlifecyclestate_id	
LIFECYCLEMODELID	=	msdyn_assetlifecyclemodel_id	
NAME	=	msdyn_name	
ONLOANLIFECYCLESTATEID	=	msdyn_onloanlifecyclestate.msdyn_assetlifecyclestate_id	
OUTBOUNDLIFECYCLESTATEID	=	msdyn_outboundlifecyclestate.msdyn_assetlifecyclestate_id	

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
RECEIVEDLIFECYCLESTATEID	=	msdyn_receivedlifecyclestate.msdyn_assetlifecyclestate_id	

Asset management asset lifecycle states to msdyn_assetlifecyclestates

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
DELETEOPENCALENDARLINES	><	msdyn_deleteopencalendarlines	
LIFECYCLESTATEID	=	msdyn_assetlifecyclestate_id	
LINE	=	msdyn_line	
MAINTENANCEASSETACTIVE	><	msdyn_maintenanceassetactive	
NAME	=	msdyn_name	

Asset management assets to msdyn_customerassets

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
ACQUISITIONCOST	=	msdyn_acquisitioncost	
ACQUISITIONDATE	=	msdyn_acquisitiondate	
ACTIVEFROM	=	msdyn_activefrom	
ACTIVETO	=	msdyn_activeto	
FUNCTIONALLOCATIONID	=	msdyn_functionallocation.msdyn_functionallocation_id	
MAINTENANCEASSETLIFECYCLESTATEID	=	msdyn_assetlifecyclestate.msdyn_assetlifecyclestate_id	
MODELID	=	msdyn_model.msdyn_model_id	
MODELPRODUCTID	=	msdyn_model.msdyn_manufacturer.msdyn_manufacturer_id	

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
MODELYEAR	=	msdyn_modelyear	
NAME	=	msdyn_name	
NOTES	=	msdyn_notes	
PURCHASEORDERID	=	msdyn_purchaseorderid	
REPLACEMENTDATE	><	msdyn_replacementdate	
REPLACEMENTVALUE	=	msdyn_replacementvalue	
SERIALID	=	msdyn_serialid	
VENDACCOUNT	=	msdyn_vendaccount	
WARRANTYDATEFROMVEND	=	msdyn_warrantydatefromvend	
WARRANTYID	=	msdyn_warranty.msdyn_warranty_id	
WRKCTRID	=	msdyn_wrkctrid	
FIXEDASSETID	=	msdyn_fixedassetid	
MAINTENANCEASSETID	=	msdyn_maintenanceassetid	
PARENTMAINTENANCEASSETID	=	msdyn_parentasset.msdyn_maintenanceassetid	
MAINTENANCEASSETTYPEID	=	msdyn_customerassetcategory.msdyn_maintenanceassettypeid	
PRODUCTID	=	msdyn_manufacturer.msdyn_manufacturer_id	

Asset management asset types to msdyn_customerassetcategories

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
LIFECYCLEMODELID	=	msdyn_lifecyclemodel.msdyn_assetlifecyclemodel_id	
MAINTENANCEASSETTYPEID	=	msdyn_maintenanceassettypeid	

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
NAME	=	msdyn_name	
CALCULATEKPITOTAL	><	msdyn_calculatekpitotal	

Asset management functional location lifecycle models to msdyn_functionallocationlifecyclemodels

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
LIFECYCLEMODELID	=	msdyn_functionallocationlifecyclemodel_id	
NAME	=	msdyn_name	

Asset management functional location lifecycle states to msdyn_functionallocationlifecyclestates

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
ALLOWDELETELOCATION	><	msdyn_allowdeletelocation	
ALLOWNEWSUBLOCATIONS	><	msdyn_allownewsublocations	
ALLOWRENAMELOCATION	><	msdyn_allowrenamelocation	
FUNCTIONALLOCATIONACTIVE	><	msdyn_functionallocationactive	
LIFECYCLESTATEID	=	msdyn_functionallocationlifecyclestate_id	
NAME	=	msdyn_name	
ALLOWINSTALLMAINTENANCEASSETS	><	msdyn_allowinstallmaintenanceassets	
CREATELOCATIONMAINTENANCEASSET	><	msdyn_createlocationmaintenanceasset	
MAINTENANCEASSETLIFECYCLESTATEID	=	msdyn_assetlifecyclestate.msdyn_assetlifecyclestate_id	

Asset management functional locations to msdyn_functionallocations

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
FUNCTIONALLOCATIONID	=	msdyn_functionallocation_id	
INVENTORYLOCATIONID	=	msdyn_inventorylocationid	
INVENTORYSITEID	=	msdyn_inventorysiteid	
NAME	=	msdyn_name	
NOTES	=	msdyn_notes	
PARENTFUNCTIONALLOCATIONID	=	msdyn_parentfunctionallocation.msdyn_functionallocation_id	
FUNCTIONALLOCATIONLIFECYCLESTATEID	=	msdyn_functionallocationlifecyclestate.msdyn_functionallocationlifecyclestate_id	
FUNCTIONALLOCATIONTYPEID	=	msdyn_functionallocationtype.msdyn_functionallocationtype_id	

Asset management functional location types to msdyn_functionallocationtypes

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
ALLOWMULTIPLEINSTALLEDASSETS	><	msdyn_allowmultipleinstalledassets	
FUNCTIONALLOCATIONTYPEID	=	msdyn_functionallocationtype_id	
NAME	=	msdyn_name	
UPDATEASSETDIMENSION	><	msdyn_updateassetdimension	
LIFECYCLEMODELID	=	msdyn_lifecyclemodelid.msdyn_functionallocationlifecyclemodel_id	
MAINTENANCEASSETTYPEID	=	msdyn_maintenanceassettype.msdyn_maintenanceassettypeid	

Asset management manufacturers to msdyn_manufacturers

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
DESCRIPTION	=	msdyn_description	
PRODUCTID	=	msdyn_manufacturer_id	

Asset management models to msdyn_models

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
DESCRIPTION	=	msdyn_description	
MODELID	=	msdyn_model_id	
PRODUCTID	=	msdyn_manufacturer.msdyn_manufacturer_id	
MAINTENANCEASSETTYPEID	=	msdyn_maintenanceassettype.msdyn_maintenanceassettypeid	

Asset management warranty to msdyn_warranties

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
NAME	=	msdyn_name	
WARRANTYID	=	msdyn_warranty_id	

Integrated worker, job, and position

2/18/2021 • 5 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

Worker data can be mastered in more than one Microsoft Dynamics 365 app. For example, Human resources (HR) data can be managed in Dynamics 365 Human Resources, Dynamics 365 Commerce, and Dynamics 365 Supply Chain Management. Regardless of where the data originates, it's integrated behind the scenes. The ability to integrate data about workers gives you the flexibility to master worker data in any Dynamics 365 app. It also provides a comprehensive view of the information in Dynamics 365 apps.

Human resources

The integration of HR data involves just mapping the HR data between Finance and Operations apps and model-driven apps in Dynamics 365.

Templates

HR data includes information about employees and contractors, positions, and jobs. A collection of table maps works together during data interaction, as shown in the following table.

FINANCE AND OPERATIONS APPS	MODEL-DRIVEN APPS IN DYNAMICS 365	DESCRIPTION
Compensation job function	cdm_jobfunctions	
Compensation job type	cdm_jobtypes	
Employment	cdm_employments	
Employment detail	cdm_employments	
Jobs	cdm_jobs	
Job detail	cdm_jobs	
Position details	cdm_jobpositions	
Position durations	cdm_jobpositions	
Position hierarchies	cdm_jobpositions	

FINANCE AND OPERATIONS APPS	MODEL-DRIVEN APPS IN DYNAMICS 365	DESCRIPTION
Position type	cdm_positiontypes	
Position worker assignments	cdm_positionworkerassignmentmaps	
Worker	cdm_workers	In Dynamics 365 Finance and Supply Chain Management data, workers are classified as either employees or contractors. Dataverse can also classify workers as volunteers. Volunteers will become contractors when the data is transformed back into Finance and Supply Chain Management.

Mapping tables

Mapping types

There are several different mapping types. The following table explains the symbols used in the template tables.

SYMBOL	DESCRIPTION
>	One-way
>>	One-way, and data is transformed in the process.
=	Bidirectional
><	Bidirectional, and data is transformed in the process.
<<	One-way, and data is transformed in the process.

Filters

The source filter and reverse source filter determine which rows are synchronized.

Default values

If a synchronized field does not exist in either the Finance and Operations table or the other Dynamics 365 table, then a default value is assigned in the synchronized table. In some cases, the default value is an integer that is a lookup to an attribute value in the Common Data Model. For example, in the Contacts table of the Common Data Model, the default value of **address1_addresstypecode** is **3**. In the Common Data Model, for address1AddressTypeCode the value of **3** is **Primary address**.

Compensation job function to cdm_jobfunctions

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
JOBFUNCTIONID	=	cdm_name	
DESCRIPTION	=	cdm_description	

Compensation job type to cdm_jobtypes

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
JOBTYPEID	=	cdm_name	
DESCRIPTION	=	cdm_description	
EXEMPTSTATUS	><	cdm_exemptstatus	

Employment to cdm_employments

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
PERSONNELNUMBER	=	cdm_workerid.cdm_workernumber	
LEGALENTITYID	=	cdm_companyid.cdm_companycode	
WORKERTYPE	>>	cdm_workertype	
EMPLOYMENTSTARTDATE	=	cdm_employmentstartdate	
EMPLOYMENTENDDATE	=	cdm_employmentenddate	

Employment detail to cdm_employments

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
PERSONNELNUMBER	=	cdm_workerid.cdm_workernumber	
LEGALENTITYID	=	cdm_companyid.cdm_companycode	
ADJUSTEDWORKERSTARTDATE	=	cdm_adjustedworkerstartdate	
EMPLOYERNOTICEAMOUNT	=	cdm_employernoticeamount	
EMPLOYERUNITOFNOTICE	=	cdm_employerunitofnotice	
EMPLOYMENTENDDATE	=	cdm_employmentenddate	

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
EMPLOYMENTSTARTDATE	=	cdm_employmentstartdate	
LASTDATEWORKED	=	cdm_lastdateworked	
TRANSITIONDATE	=	cdm_transitiondate	
EMPLOYMENTTYPE	=	cdm_workertype	
VALIDFROM	=	cdm_validfrom	
VALIDTO	=	cdm_validto	
WORKERNOTICEAMOUNT	=	cdm_workernoticeamount	
WORKERUNITOFNOTICE	=	cdm_workerunitofnotice	

Jobs to cdm_jobs

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
JOBID	=	cdm_name	
MAXIMUMNUMBEROFPOSITIONS	=	cdm_maximumnumberofpositions	
ALLOWUNLIMITEDPOSITIONS	><	cdm_allowunlimitedpositions	
DESCRIPTION	=	cdm_description	
JOBDESCRIPTION	=	cdm_jobdescription	
JOBTYPEID	=	cdm_jobtypeid.cdm_name	
FUNCTIONID	=	cdm_jobfunctionid.cdm_name	
EFFECTIVE	=	cdm_validfrom	
EXPIRATION	=	cdm_validto	
FULLTIMEEQUIVALENT	=	cdm_defaultfulltimeequivalent	

Job detail to cdm_jobs

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
JOBID	=	cdm_name	
JOBTYPEID	=	cdm_jobtypeid.cdm_name	
FUNCTIONID	=	cdm_jobfunctionid.cdm_name	
VALIDFROM	=	cdm_validfrom	
VALIDTO	=	cdm_validto	
FULLTIMEEQUIVALENT	=	cdm_defaultfulltimeequivalent	

Position details to cdm_jobpositions

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
POSITIONID	=	cdm_jobpositionnumber	
JOBID	=	cdm_jobid.cdm_name	
DESCRIPTION	=	cdm_description	
DEPARTMENTNUMBER	=	cdm_departmentid.cdm_departmentnumber	
AVAILABLEFORASSIGNMENT	=	cdm_availableforassignment	
VALIDFROM	=	cdm_validfrom	
VALIDTO	=	cdm_validto	
FULLTIMEEQUIVALENT	=	cdm_fulltimeequivalent	
POSITIONTYPEID	=	cdm_positiontypeid.cdm_name	

Position durations to cdm_jobpositions

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
POSITIONID	=	cdm_jobpositionnumber	

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
VALIDFROM	=	cdm_activation	
VALIDTO	=	cdm_retirement	

Position hierarchies to cdm_jobpositions

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
POSITIONID	=	cdm_jobpositionnumber	
PARENTPOSITIONID	=	cdm_parentjobpositionid.cdm_jobpositionnumber	
HIERARCHYTYPENAME	<<	none	Line
VALIDFROM	<<	cdm_validfrom	
VALIDTO	<<	cdm_validto	

Position type to cdm_positiontypes

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
POSITIONTYPEID	=	cdm_name	
DESCRIPTION	=	cdm_description	
CLASSIFICATION	><	cdm_classification	

Position worker assignments to cdm_positionworkerassignmentmaps

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
PERSONNELNUMBER	=	cdm_workerid.cdm_workernumber	
POSITIONID	=	cdm_jobpositionid.cdm_jobpositionnumber	
VALIDFROM	=	cdm_validfrom	

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
VALIDTO	=	cdm_validto	

Worker to cdm_workers

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS APPS	MAP TYPE	CUSTOMER ENGAGEMENT APPS	DEFAULT VALUE
PERSONNELNUMBER	=	cdm_workernumber	
FIRSTNAME	=	cdm_firstname	
MIDDLENAME	=	cdm_middlename	
LASTNAME	=	cdm_lastname	
WORKERTYPE	>>	cdm_type	
WORKERSTATUS	>>	cdm_status	
PRIMARYCONTACTEMAIL	=	cdm_primaryemailaddress	
PRIMARYCONTACTPHONE	=	cdm_primarytelephone	
PRIMARYCONTACTFACEBOOK	=	cdm_facebookidentity	
PRIMARYCONTACTTWITTER	=	cdm_twitteridentity	
PRIMARYCONTACTLINKEDIN	=	cdm_linkedinidentity	
PRIMARYCONTACTURL	=	cdm_websiteurl	
GENDER	><	cdm_gender	
BIRTHDATE	=	cdm_birthdate	
NAME	>>	cdm_fullname	

Inventory availability in dual-write

2/18/2021 • 3 minutes to read • Edit Online

By using inventory availability, you can check your inventory before you add a product to the **Quotations**, **Orders**, or **Invoices** page in Microsoft Dynamics 365 Sales. For example, you check inventory and determine a fulfillment date as one key task in the prospect-to-cash process.

If you don't have enough inventory, you can estimate a delivery date, based on projected inventory receipts and issues. You can also check the product's available-to-promise (ATP) information, where you can find the ATP quantity in the predefined time fence.

On-hand inventory

In Dynamics 365 Sales, a new **On-hand Inventory** button has been added to the header of the **Quotes**, **Orders**, and **Invoices** pages. When you select this button, a dialog box appears, where you can specify the company and the product that you want to check the on-hand inventory for. This dialog box shows the same information as On-hand inventory.

The dialog box returns the inventory information from Dynamics 365 Supply Chain Management. This information includes the following quantities:

- On-hand quantity
- Reserved on-hand quantity
- Available on-hand quantity
- Ordered quantity
- On-order quantity
- Reserved ordered quantity
- Total available quantity

ATP information

In Sales, a new **ATP Information** button has been added to line items on the **Quotes**, **Orders**, and **Invoices** pages. When you select this button, a dialog box appears, where you can specify the company, product, inventory site, inventory warehouse, and order quantity. This dialog box has the same settings that are described in Order promising.

The dialog box returns the ATP information from Supply Chain Management. This information includes the following quantities:

- ATP quantity
- Receipt quantity
- Issue quantity
- On-hand quantity

How it works

When you select the **On-hand Inventory** button on the **Quotes**, **Orders**, or **Invoices** page, a live dual-write call is made to the **Onhand inventory** API. The API calculates the on-hand inventory for the given product. The result is stored in the **InventCDSInventoryOnHandRequestEntity** and **InventCDSInventoryOnHandEntryEntity** tables, and then is written to Dataverse by dual-write. To use this

functionality, you need to run the following dual-write maps. Skip initial synchronization when you run the maps.

- CDS inventory on-hand entries (msdyn_inventoryonhandentries)
- CDS inventory on-hand requests (msdyn_inventoryonhandrequests)

Templates

The following templates are available for the exposing the onhand inventory data.

FINANCE AND OPERATIONS APPS	CUSTOMER ENGAGEMENT APP	DESCRIPTION
CDS inventory on-hand entries	msdyn_inventoryonhandentries	
CDS inventory on-hand requests	msdyn_inventoryonhandrequests	

Mapping tables

Mapping types

There are several different mapping types. The following table explains the symbols used in the template tables.

SYMBOL	DESCRIPTION
>	One-way
>>	One-way, and data is transformed in the process.
=	Bidirectional
><	Bidirectional, and data is transformed in the process.
<<	One-way, and data is transformed in the process.

Filters

The source filter and reverse source filter determine which rows are synchronized.

Default values

If a synchronized field does not exist in either the Finance and Operations table or the other Dynamics 365 table, then a default value is assigned in the synchronized table. In some cases, the default value is an integer that is a lookup to an attribute value in the Common Data Model. For example, in the Contacts table of the Common Data Model, the default value of **address1_addresstypecode** is **3**. In the Common Data Model, for address1AddressTypeCode the value of **3** is **Primary address**.

CDS inventory on-hand entries (msdyn_inventoryonhandentries)

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
`REQUESTID`	=	`msdyn_request.msdyn_requestid`	
`INVENTORYSITEID`	=	`msdyn_inventorysite.msdyn_siteid`	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
INVENTORYWAREHOUSEID	=	msdyn_inventorywarehouse.msdyn_warehouseidentifier	
AVAILABLEONHANDQUANTITY	>	msdyn_availableonhandquantity	
AVAILABLEORDEREDQUANTITY	>	msdyn_availableorderedquantity	
ONHANDQUANTITY	>	msdyn_onhandquantity	
ONORDERQUANTITY	>	msdyn_onorderquantity	
ORDEREDQUANTITY	>	msdyn_orderedquantity	
RESERVEDONHANDQUANTITY	>	msdyn_reservedonhandquantity	
RESERVEDORDEREDQUANTITY	>	msdyn_reservedorderedquantity	
TOTALAVAILABLEQUANTITY	>	msdyn_totalavailablequantity	
ATPDATE	=	msdyn_atpdate	
ATPQUANTITY	>	msdyn_atpquantity	
PROJECTEDISSUEQUANTITY	>	msdyn_projectedissuequantity	
PROJECTEDONHANDQUANTITY	>	msdyn_projectedonhandquantity	
PROJECTEDRECEIPTQUANTITY	>	msdyn_projectedreceiptquantity	
ORDERQUANTITY	>	msdyn_orderquantity	
UNAVAILABLEONHANDQUANTITY	>	msdyn_unavailableonhandquantity	

CDS inventory on-hand requests (msdyn_inventoryonhandrequests)

This template synchronizes data between Finance and Operations apps and Dataverse.

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
REQUESTID	=	msdyn_requestid	
PRODUCTNUMBER	<	msdyn_product.msdyn_productnumber	
ISATPCALCULATION	<<	msdyn_isatpcalculation	
ORDERQUANTITY	<	msdyn_orderquantity	
INVENTORYSITEID	<	msdyn_inventorysite.msdyn_siteid	

FINANCE AND OPERATIONS FIELD	MAP TYPE	CUSTOMER ENGAGEMENT FIELD	DEFAULT VALUE
`INVENTORYWAREHOUSEID`	<	`msdyn_inventorywarehouse.msdyn_warehouseidentifier`	
`REFERENCENUMBER`	<	`msdyn_referencenumber`	
`LINECREATIONSEQUENCENUMBER`	<	`msdyn_linecreationsequencenumber`	

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Support for Field Service and Project Service Automation solutions

2/18/2021 • 2 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

Support for Field Service solutions

Microsoft supports dual-write on top of existing Dataverse environments that are based on Field Service solutions.

For more information, see Integrate Dynamics 365 Field Service and Supply Chain Management.

Support for Project Service Automation solutions

Microsoft supports dual-write on top of existing Dataverse environments that are based on Project Service Automation solutions.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Migrate Prospect to cash data from Data Integrator to dual-write

2/18/2021 • 4 minutes to read • Edit Online

To migrate your Prospect to cash data from Data Integrator to dual-write, follow these steps.

1. Run the Prospect to cash Data Integrator jobs to do one final full synchronization. In this way, you ensure that both systems (Finance and Operations apps and customer engagement apps) have all the data.
2. To help prevent potential data loss, export the Prospect to cash data from Microsoft Dynamics 365 Sales to an Excel file or a comma-separated values (CSV) file. Export data from the following entities:
   - Account
   - Contact
   - Invoice
   - Invoice products
   - Order
   - Order products
   - Products
   - Quote
   - Quote products
3. Uninstall the Prospect to cash solution from the Sales environment. This step removes the columns and corresponding data that the Prospect to cash solution introduced.
4. Install the dual-write solution.
5. Create a dual-write connection between the Finance and Operations app and the customer engagement app for one or more legal entities.
6. Enable dual-write table maps, and run the initial synchronization for the required reference data. (For more information, see Considerations for initial synchronization.) Examples of required data include customer groups, payment terms, and payment schedules. Don't enable the dual-write maps for tables that require initialization, such as the account, quote, quote line, order, and order line tables.
7. In the customer engagement app, go to **Advanced Settings > System Settings > Data Management > Duplicate detection rules**, and disable all the rules.
8. Initialize the tables that are listed in step 2. For instructions, see the remaining sections of this topic.
9. Open the Finance and Operations app, and enable the table maps, such as the account, quote, quote line, order, and order line table maps. Then run the initial synchronization. (For more information, see Considerations for initial synchronization.) This process will sync additional information from the Finance and Operations app, such as processing status, shipping and billing addresses, sites, and warehouses.

Account table

1. In the **Company** column, enter the company name, such as **USMF**.
2. In the **Relationship Type** column, enter **Customer** as a static value. You might not want to classify every account record as a customer in your business logic.
3. In the **Customer Group ID** column, enter the customer group number from the Finance and Operations

app. The default value from the Prospect to cash solution is **10**.

4. If you're using the Prospect to cash solution without any customization of **Account Number**, enter an **Account Number** value in the **Party Number** column. If there are customizations, and you don't know the party number, pull this information from the Finance and Operations app.

Contact table

1. In the **Company** column, enter the company name, such as **USMF**.
2. Set the following columns, based on the **IsActiveCustomer** value in the CSV file:
   - If **IsActiveCustomer** is set to **Yes** in the CSV file, set the **Sellable** column to **Yes**. In the **Customer Group ID** column, enter the customer group number from the Finance and Operations app. The default value from the Prospect to cash solution is **10**.
   - If **IsActiveCustomer** is set to **No** in the CSV file, set the **Sellable** column to **No**, and set the **Contact For** column to **Customer**.
3. If you're using the Prospect to cash solution without any customization of **Contact Number**, set the following columns:
   - Migrate the contact number from the CSV file (**msdynce_contactnumber**) to the contact number in the **Contact** table (**msdyn_contactnumber**).
   - Use values from the **Contact Number** table in the **Party Number** column.
   - Use values from the **Contact Number** table in the **Account Number/Contact Person ID** column.

Invoice table

Because data from the **Invoice** table is designed to flow one way, from the Finance and Operations app to the customer engagement app, initialization isn't required. Run the initial synchronization to migrate all the required data from the Finance and Operations app to the customer engagement app. For more information, see Considerations for initial synchronization.

Order table

1. In the **Company** column, enter the company name, such as **USMF**.
2. Copy the value of the **Order ID** column in the CSV file to the **Sales Order Number** column.
3. Copy the value of the **Customer** column in the CSV file to the **Invoice customer number** column.
4. Copy the value of the **Ship To Country/Region** column in the CSV file to the **Ship To Country/Region** column. Examples of this value include **US** and **United States**.
5. Set the **Requested Receipt Date** column. If you aren't using a receipt date, use the **Requested Delivery Date**, **Date Fulfilled**, and **Date Submitted** columns in the CSV file. An example of this value is **2020-03-27T00:00:00Z**.
6. Set the **Language** column. An example of this value is **en-us**.
7. Set the **Order Type** column by using the **Item-based** column.

Order products table

- In the **Company** column, enter the company name, such as **USMF**.

Products table

Because data from the **Products** table is designed to flow one way, from the Finance and Operations app to the customer engagement app, initialization isn't required. Run the initial synchronization to migrate all the required data from the Finance and Operations app to the customer engagement app. For more information, see

Considerations for initial synchronization.

Quote and Quote product tables

For the **Quote** table, follow the instructions in the Order table section earlier in this topic. For the **Quote product** table, follow the instructions in the Order products table section.

Verify dual-write configuration in Finance and Operations apps and Dataverse

2/18/2021 • 2 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

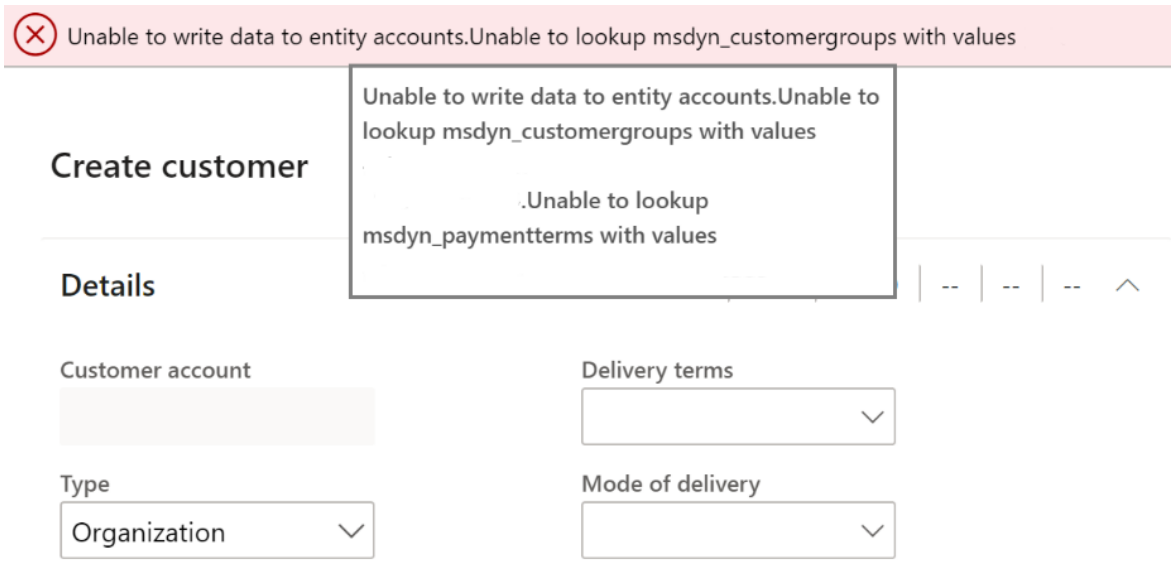
This topic provides troubleshooting information for dual-write integration between Finance and Operations apps and Dataverse. Specifically, it explains how you can determine whether dual-write is configured in Finance and Operations apps and in Dataverse.

Verify that dual-write is configured in a Finance and Operations app

To determine whether the errors that you see when you try to save rows for update come from dual-write, first verify that dual-write is configured.

- If you have admin privileges in the Finance and Operations app, go to **Workspaces** > **Data management**, and select the **Dual-write** tile. If the details of the linked environments and the list of table maps that are running are shown, dual-write is configured.

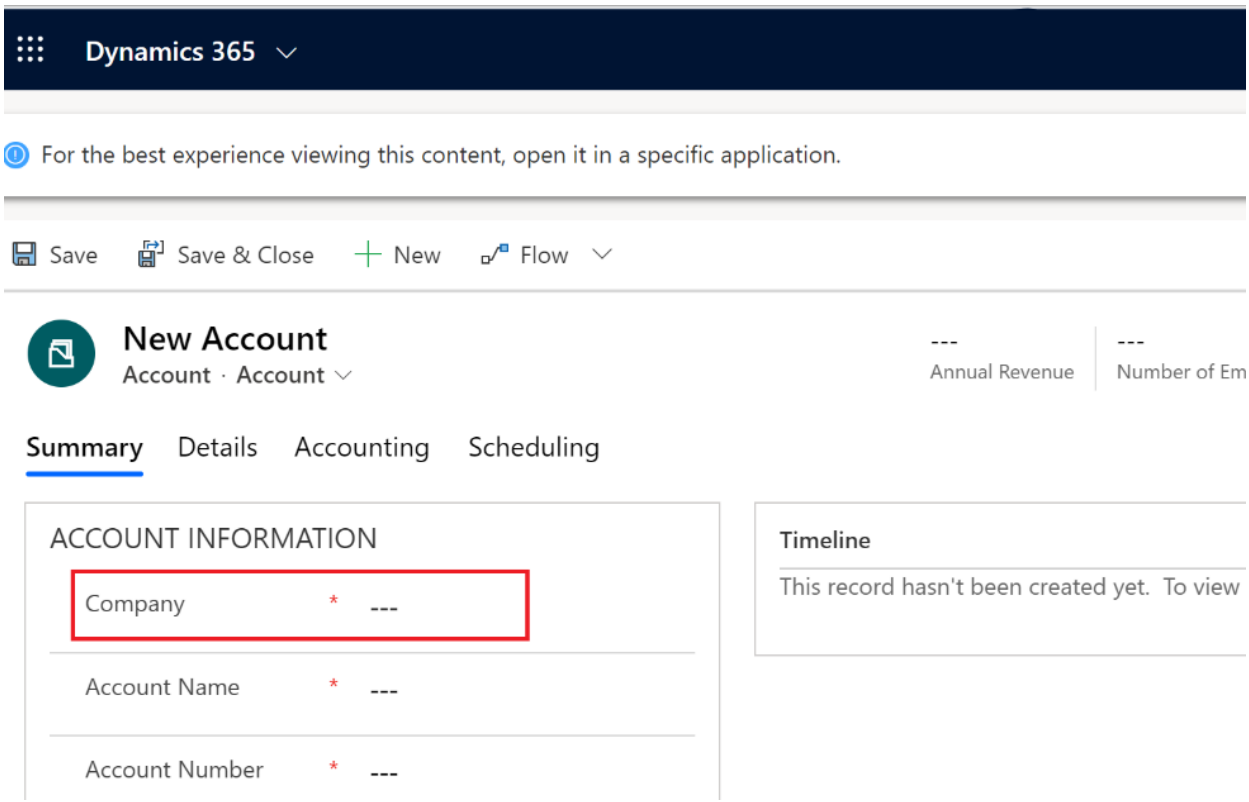
- If you don't have admin privileges, you will receive an error message, *Unable to write data to entity <entity name>*. In the example in the following illustration, you can't create a customer row in the Finance and Operations app, because dual-write is configured, but the customer group and payment terms reference data don't exist in Dataverse.



For information about how to fix issues when you create data in Finance and Operations apps, see Troubleshoot live synchronization issues.

Verify that dual-write is configured in Dataverse

When you create data, if you see the **Company** column on pages in Dataverse, dual-write is configured.



For information about how to fix issues when you create data in Dataverse, see Troubleshoot live synchronization issues.

For information about how to view error details if you encounter any errors while you create data in Dataverse, see Enable and view the plug-in trace log in Dataverse to view error details.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Troubleshoot issues during initial setup

2/18/2021 • 3 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

This topic provides troubleshooting information for dual-write integration between Finance and Operations apps and Dataverse. Specifically, it provides information that can help you fix issues that might occur during the initial setup of dual-write integration.

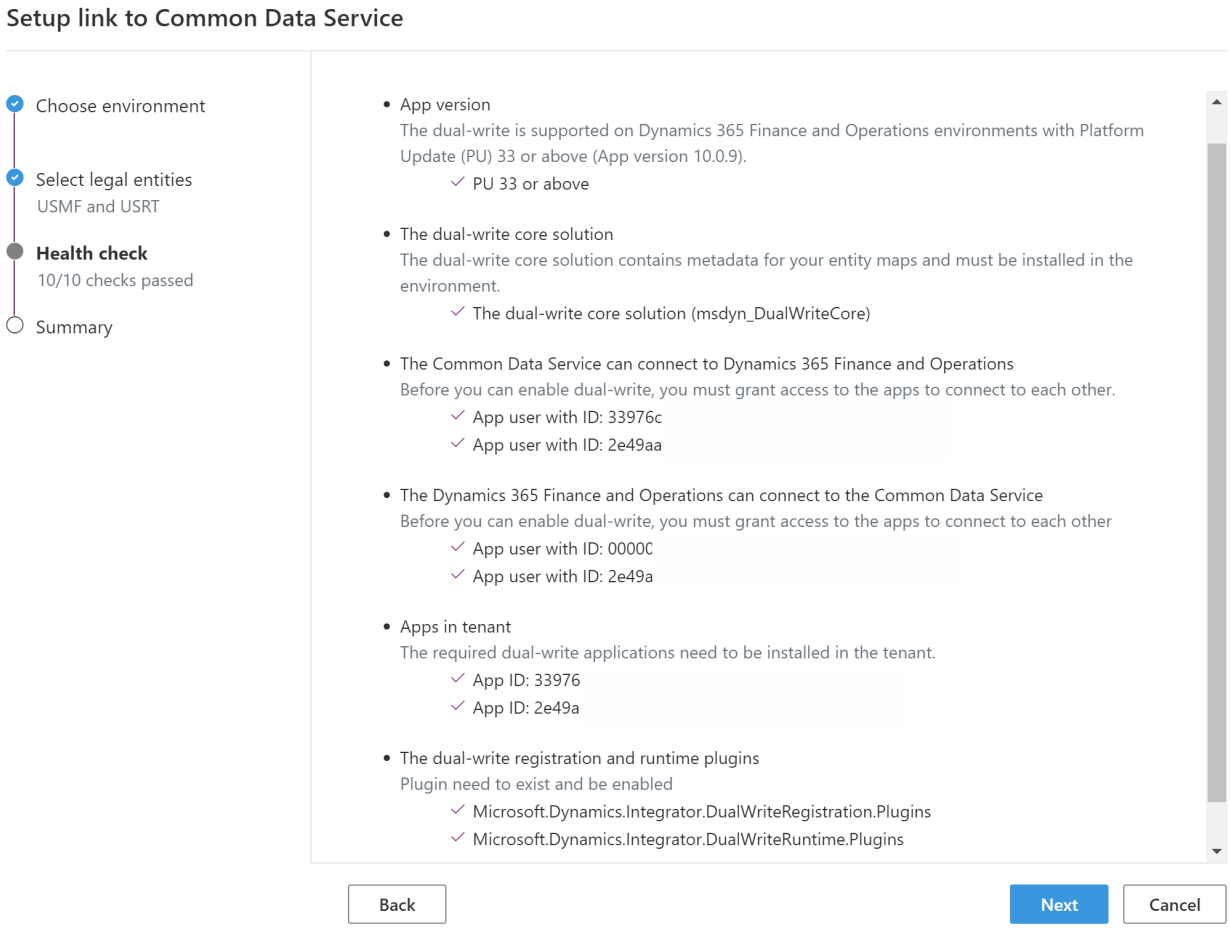
IMPORTANT

Some of the issues that this topic addresses might require either the system admin role or Microsoft Azure Active Directory (Azure AD) tenant admin credentials. The section for each issue explains whether a specific role or credentials are required.

You can't link a Finance and Operations app to Dataverse

Required role to set up dual-write: System administrator in Finance and Operations apps and Dataverse.

Errors on the **Setup link to Dataverse** page are usually caused by incomplete setup or permissions issues. Make sure that the whole health check passes on the **Setup link to Dataverse** page, as shown in the following illustration. You can't link dual-write unless the whole health check passes.



You must have Azure AD tenant admin credentials to link the Finance and Operations and Dataverse environments. After you link the environments, users can sign in by using their account credentials and update an existing table map.

Error when you open the Link to Dataverse page

Required credentials to fix the issue: Azure AD tenant admin

You might receive the following error message when you open the **Link to Dataverse** page in a Finance and Operations app:

Response status code does not indicate success: 404 (Not Found).

This error occurs when the consent step hasn't been completed. To validate whether the consent step has been completed, sign in to portal.Azure.com by using the Azure AD tenant admin account, and see whether the third-party app that has the ID **33976c19-1db5-4c02-810e-c243db79efde** appears in the Azure AD **Enterprise applications** list. If it doesn't, you must provide app consent.

To provide app consent, follow these steps.

1. Open the following URL by using your admin credentials. You should be prompted for consent.

   https://login.microsoftonline.com/common/oauth2/authorize?client_id=33976c19-1db5-4c02-810e-c243db79efde&response_type=code&prompt=admin_consent

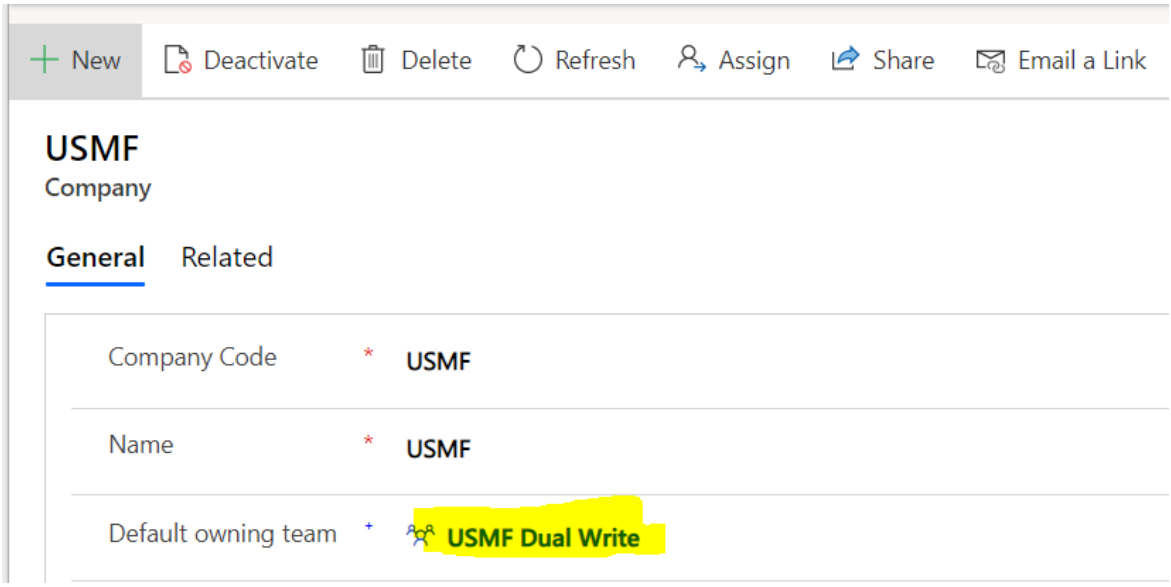
2. Select **Accept** to indicate that you're giving your consent to install the app that has the ID **33976c19-1db5-4c02-810e-c243db79efde** in your tenant.

> **TIP**
> This app is required to link Dataverse and Finance and Operations apps. If you have trouble with this step, open your browser in incognito mode (in Google Chrome) or InPrivate mode (in Microsoft Edge).

Verify that company data and dual-write teams are set up correctly during linking

To ensure that dual-write works correctly, the companies that you select during configuration are created in the Dataverse environment. By default, these companies are read-only, and the **IsDualWriteEnable** property is set to **True**. In addition, the default owning business unit owner and team are created and include the company name. Before you enable the maps, verify that the default team owner is specified. To find the **Companies (CDM_Company)** table, follow these steps.

1. In the model-driven app in Dynamics 365, select the filter in the upper-right corner.
2. In the drop-down list, select **Company**.
3. Select **Run** to see the results.
4. Select the company that was linked when you configured dual-write.
5. Verify that the **Default owning team** column has a value. In the following illustration, the **Default owning team** column is set to **USMF Dual Write**.



Find the limit on the number of legal tables or companies that can be linked for dual-write

You might receive the following error message when you try to enable maps:

Dual write failure - Plugin registration failed: [(Unable to get partition map for project DWM-1ae35e60-4bc2-4905-88ea-69efd3b29260-7f12cb89-1550-42e2-858e-4761fc1443ea. Error Exceeds the maximum partitions allowed for mapping DWM-1ae35e60-4bc2-4905-88ea-69efd3b29260-7f12cb89-1550-42e2-858e-4761fc1443ea)], One or more errors occurred.

The current limit when you link the environments is approximately 40 legal tables. This error occurs if you try to enable maps, and more than 40 legal tables are linked between the environments.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Troubleshoot issues during initial synchronization

2/18/2021 • 7 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

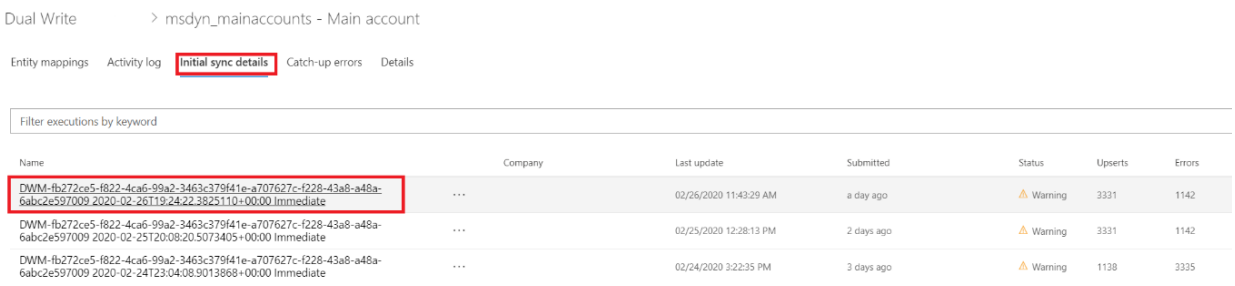
This topic provides troubleshooting information for dual-write integration between Finance and Operations apps and Dataverse. Specifically, it provides information that can help you fix issues that might occur during initial synchronization.

IMPORTANT

Some of the issues that this topic addresses might require either the system admin role or Microsoft Azure Active Directory (Azure AD) tenant admin credentials. The section for each issue explains whether a specific role or credentials are required.

Check for initial synchronization errors in a Finance and Operations app

After you enable the mapping templates, the status of the maps should be **Running**. If the status is **Not running**, errors occurred during initial synchronization. To view the errors, select the **Initial sync details** tab on the **Dual-write** page.



You can't complete initial synchronization: 400 Bad Request

Required role to fix the issue: System admin

You might receive the following error message when you try to run the mapping and initial synchronization:

([Bad Request], The remote server returned an error: (400) Bad Request.), AX export encountered an error

Here is an example of the full error message.

```
Dual write Initial Sync completed with status: Error. Following are the details:
Executed leg: From AX Financial dimensions to CRM msdyn_dimensionattributes
with exported records count: 0, ImportRecordsErrorCount: 0,
ImportRecordsInsertedCount: 0 and ImportRecordsUpdatedCount: 0
ErrorsDetails:
Dual write Initial sync failed
Message: ([Bad Request], The remote server returned an error: (400) Bad Request.), AX export encountered an
error
Stacktrace: at
Microsoft.Dynamics.Integrator.QueryGenerator.AxClient.\<ExportAxPackage\>d__16.MoveNext()
in X:\\bt\\1024532\\repo\\src\\Core\\QueryGenerator\\AxClient.cs:line 265
\--- End of stack trace from previous location where exception was thrown ---
at System.Runtime.ExceptionServices.ExceptionDispatchInfo.Throw()
at System.Runtime.CompilerServices.TaskAwaiter.HandleNonSuccessAndDebuggerNotification(Task task)
at Microsoft.D365.ServicePlatform.Context.ServiceContext.Activity.\<ExecuteAsync\>d__11\`2.MoveNext()
\--- End of stack trace from previous location where exception was thrown ---
```

If this error occurs consistently, and you can't complete the initial synchronization, follow these steps to fix the issue.

1. Sign in to the virtual machine (VM) for the Finance and Operations app.
2. Open Microsoft Management Console.
3. In the **Services** pane, make sure that the Microsoft Dynamics 365 Data import export framework service is running. Restart it if it has been stopped, because the initial synchronization requires it.

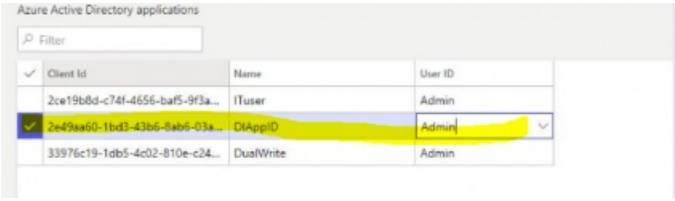
Initial synchronization error: 403 Forbidden

You might receive the following error message during initial synchronization:

([Forbidden], The remote server returned an error: (403) Forbidden.), AX export encountered an error

To fix the issue, follow these steps.

1. Sign in to the Finance and Operations app.
2. On the **Azure Active Directory applications** page, delete the **DtAppID** client, and then add it again.



Self-reference or circular reference failures during initial synchronization

You might receive an error messages if any of your mappings have self-references or circular references. The errors fall into these categories:

- Errors in the Vendors V2–to–msdyn_vendors table mapping
- Errors in the Customers V3–to–Accounts table mapping

Resolve errors in the Vendors V2–to–msdyn_vendors table mapping

You might encounter initial synchronization errors for the mapping of **Vendors V2** to **msdyn_vendors** if the tables have existing rows where there are values in the **PrimaryContactPersonId** and **InvoiceVendorAccountNumber** columns. These errors occur because **InvoiceVendorAccountNumber** is a self-referencing column, and **PrimaryContactPersonId** is a circular reference in the vendor mapping.

The error messages that you receive will have the following form.

Couldn't resolve the guid for the field: <field>. The lookup was not found: <value>. Try this URL(s) to check if the reference data exists:

```
https://focdsdevtest2.crm.dynamics.com/api/data/v9.0/<entity>?$select=<field>&$filter=<field> eq <value>
```

Here are some examples:

- *Couldn't resolve the guid for the field: msdyn_vendorprimarycontactperson.msdyn_contactpersonid. The lookup was not found: 000056. Try this URL(s) to check if the reference data exists:*

  ```
  https://focdsdevtest2.crm.dynamics.com/api/data/v9.0/contacts?
  $select=msdyn_contactpersonid.contactid&$filter=msdyn_contactpersonid eq '000056'
  ```

- *Couldn't resolve the guid for the field: msdyn_invoicevendoraccountnumber.msdyn_vendoraccountnumber. The lookup was not found: V24-1. Try this URL(s) to check if the reference data exists:*

  ```
  https://focdsdevtest2.crm.dynamics.com/api/data/v9.0/msdn_vendors?
  $select=msdyn_vendoraccountnumber,msdyn_vendorid&$filter=msdyn_vendoraccountnumber eq 'V24-1'
  ```

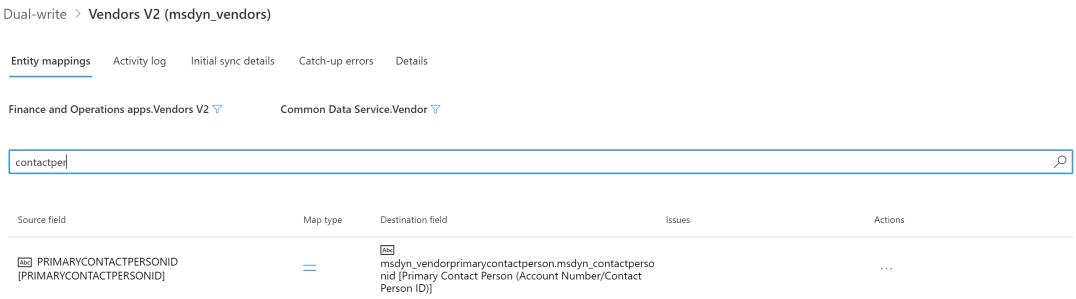
If any rows in the vendor table have values in the **PrimaryContactPersonId** and **InvoiceVendorAccountNumber** columns, follow these steps to complete the initial synchronization.

1. In the Finance and Operations app, delete the **PrimaryContactPersonId** and **InvoiceVendorAccountNumber** columns from the mapping, and then save the mapping.

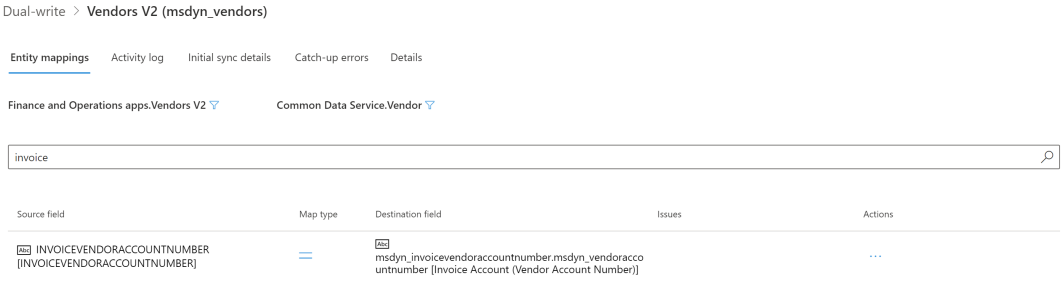
   a. On the dual-write mapping page for **Vendors V2 (msdyn_vendors)**, on the **Table mappings** tab, in the left filter, select **Finance and Operations apps.Vendors V2**. In the right filter, select **Sales.Vendor**.

   b. Search for **primarycontactperson** to find the **PrimaryContactPersonId** source column.

   c. Select **Actions**, and then select **Delete**.

   d. Repeat these steps to delete the **InvoiceVendorAccountNumber** column.

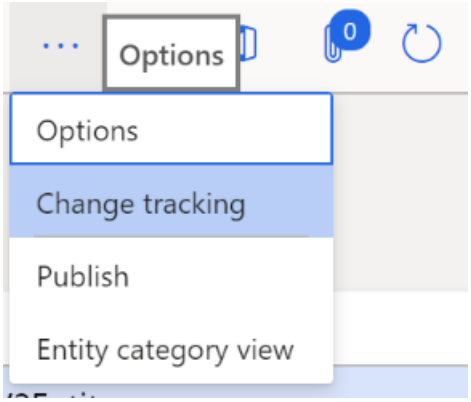
   e. Save your changes to the mapping.

2. Turn off change tracking for the **Vendors V2** table.

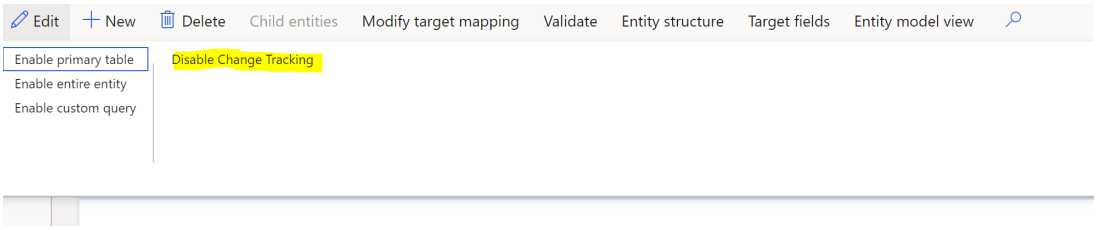
   a. In the **Data management** workspace, select the **Data tables** tile.

   b. Select the **Vendors V2** table.

   c. On the Action Pane, select **Options**, and then select **Change tracking**.



d. Select **Disable Change Tracking**.



3. Run initial synchronization for the **Vendors V2 (msdyn_vendors)** mapping. The initial synchronization should run successfully, without any errors.
4. Run initial synchronization for the **CDS Contacts V2 (contacts)** mapping. You must sync this mapping if you want to sync the primary contact column on the vendors table, because initial synchronization must also be done for the contact rows.
5. Add the **PrimaryContactPersonId** and **InvoiceVendorAccountNumber** columns back to the **Vendors V2 (msdyn_vendors)** mapping, and then save the mapping.
6. Run initial synchronization again for the **Vendors V2 (msdyn_vendors)** mapping. Because change tracking is turned off, all the rows will be synced.
7. Turn change tracking back on for the **Vendors V2** table.

Resolve errors in the Customers V3–to–Accounts table mapping

You might encounter initial synchronization errors for the mapping of **Customers V3** to **Accounts** if the tables have existing rows where there are values in the **ContactPersonID** and **InvoiceAccount** columns. These errors occur because **InvoiceAccount** is a self-referencing column, and **ContactPersonID** is a circular reference in the vendor mapping.

The error messages that you receive will have the following form.

Couldn't resolve the guid for the field: <field>. The lookup was not found: <value>. Try this URL(s) to check if the reference data exists:

```
https://focdsdevtest2.crm.dynamics.com/api/data/v9.0/<entity>?$select=<field>&$filter=<field> eq <value>
```

Here are some examples:

- *Couldn't resolve the guid for the field: primarycontactid.msdyn_contactpersonid. The lookup was not found: 000056. Try this URL(s) to check if the reference data exists:*

  ```
  https://focdsdevtest2.crm.dynamics.com/api/data/v9.0/contacts?
  $select=msdyn_contactpersonid.contactid&$filter=msdyn_contactpersonid eq '000056'
  ```

- *Couldn't resolve the guid for the field: msdyn_billingaccount.accountnumber. The lookup was not found: 1206-1. Try this URL(s) to check if the reference data exists:*

  ```
  https://focdsdevtest2.crm.dynamics.com/api/data/v9.0/accounts?
  $select=accountnumber.account&$filter=accountnumber eq '1206-1'
  ```

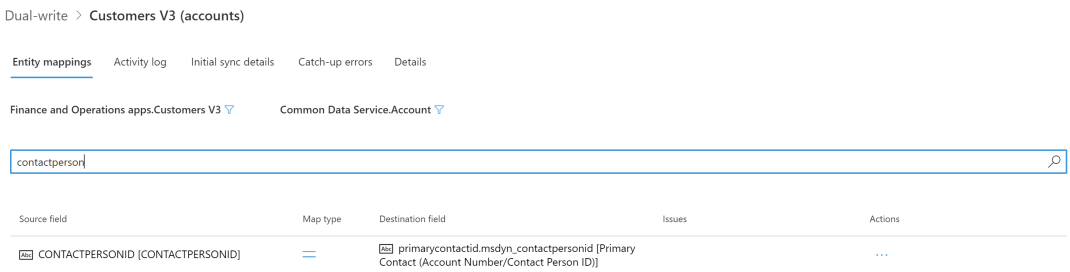
If any rows in the customer table have values in the **ContactPersonID** and **InvoiceAccount** columns, follow these steps to complete the initial synchronization. You can use this approach for any out-of-box tables, such **Accounts** and **Contacts**.

1. In the Finance and Operations app, delete the **ContactPersonID** and **InvoiceAccount** columns from the **Customers V3 (accounts)** mapping, and then save the mapping.

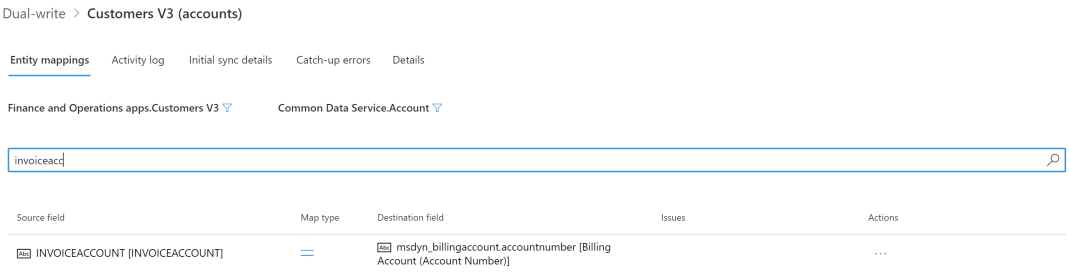
    a. On the dual-write mapping page for **Customers V3 (accounts)**, on the **Table mappings** tab, in the left filter, select **Finance and Operations app.Customers V3**. In the right filter, select **Dataverse.Account**.

    b. Search for **contactperson** to find the **ContactPersonID** source column.

    c. Select **Actions**, and then select **Delete**.



    d. Repeat these steps to delete the **InvoiceAccount** column.



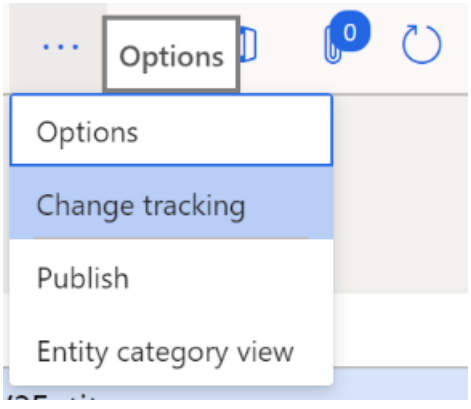
    e. Save your changes to the mapping.

2. Turn off change tracking for the **Customers V3** table.

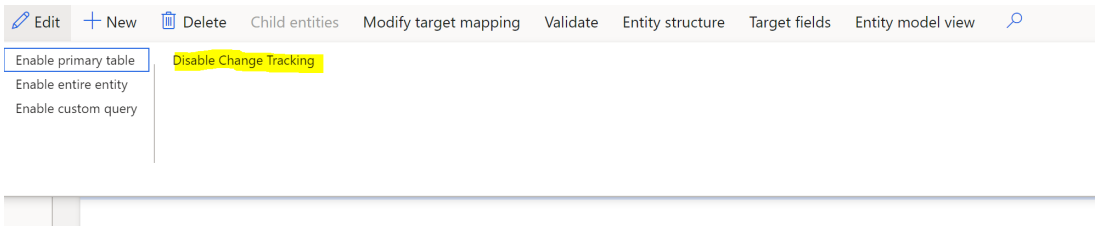
    a. In the **Data management** workspace, select the **Data tables** tile.

    b. Select the **Customers V3** table.

    c. On the Action Pane, select **Options**, and then select **Change tracking**.



    d. Select **Disable Change Tracking**.



3. Run initial synchronization for the **Customers V3 (Accounts)** mapping. The initial synchronization should run successfully, without any errors.

4. Run initial synchronization for the **CDS Contacts V2 (contacts)** mapping.

   > **NOTE**
   >
   > There are two maps that have the same name. Be sure to select the map that has the following description on the **Details** tab: **Dual-write template for sync between FO.CDS Vendor Contacts V2 to CDS.Contacts. Requires new package [Dynamics365SupplyChainExtended].**

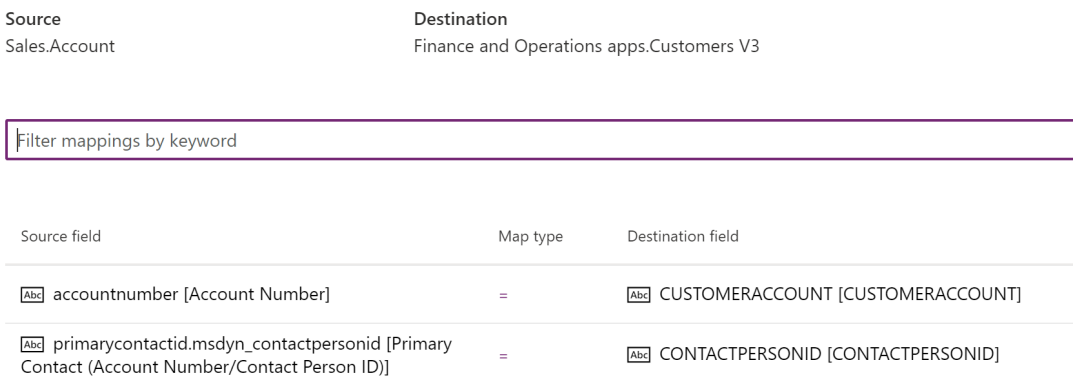
5. Add the **InvoiceAccount** and **ContactPersonId** columns back to the **Customers V3 (Accounts)** mapping, and then save the mapping. Both the **InvoiceAccount** column and the **ContactPersonId** column are now part of live synchronization mode again. In the next step, you will do the initial synchronization for these columns.

6. Run initial synchronization again for the **Customers V3 (Accounts)** mapping. Because change tracking is turned off, the data for **InvoiceAccount** and **ContactPersonId** will be synced from the Finance and Operations app to Dataverse.

7. To sync the data for **InvoiceAccount** and **ContactPersonId** from Dataverse to the Finance and Operations app, you must use a data integration project.

   a. In Power Apps, create a data integration project between the **Sales.Account** and **Finance and Operations apps.Customers V3** tables. The data direction must be from Dataverse to the Finance and Operations app. Because **InvoiceAccount** is a new attribute in dual-write, you might want to skip initial synchronization for it. For more information, see Integrate data into Dataverse.

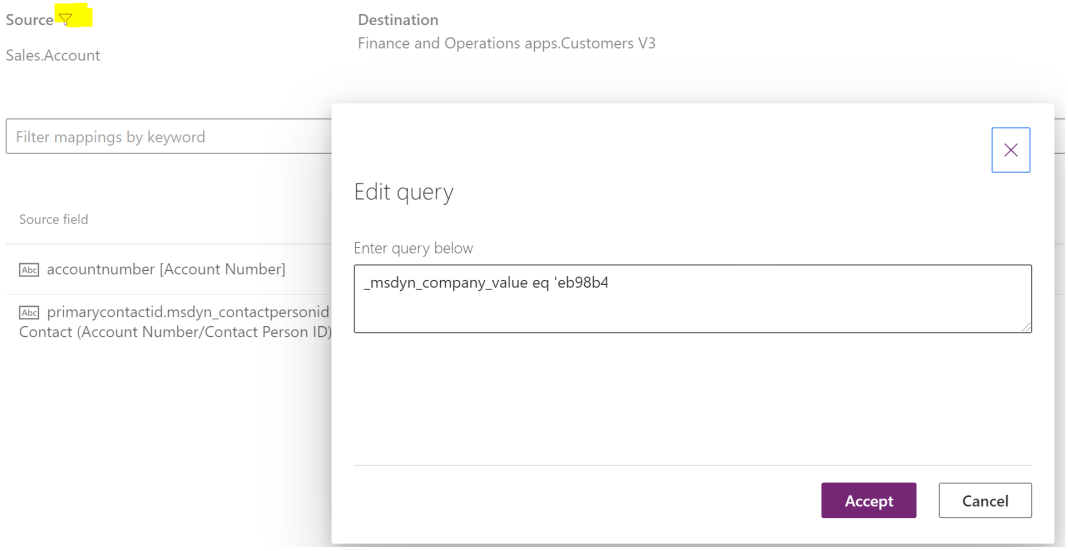
   The following illustration shows a project that updates **CustomerAccount** and **ContactPersonId**.



   b. Add the company criteria in the filter on the Dataverse side, so that only rows that match the filter criteria will be updated in the Finance and Operations app. To add a filter, select the filter button. Then, in the **Edit query** dialog box, you can add a filter query such as **_msdyn_company_value eq '<guid>'**.

> [NOTE] If the filter button isn't present, create a support ticket to ask the data integration team to enable the filter capability on your tenant.

If you don't enter a filter query for **_msdyn_company_value**, all the rows will be synced.



The initial synchronization of the rows is now completed.

8. In the Finance and Operations app, turn change tracking back on for the **Customers V3** table.

Troubleshoot dual-write issues in Finance and Operations apps

2/18/2021 • 3 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

This topic provides troubleshooting information for dual-write integration between Finance and Operations apps and Dataverse. Specifically, it provides information that can help you fix issues with the **Dual-write** module in Finance and Operations apps.

IMPORTANT

Some of the issues that this topic addresses might require either the system admin role or Microsoft Azure Active Directory (Azure AD) tenant admin credentials. The section for each issue explains whether a specific role or credentials are required.

You can't load the dual-write module in a Finance and Operations app

If you can't open the **Dual-write** page by selecting the **Dual Write** tile in the **Data management** workspace, the data integration service is probably down. Create a support ticket to request a restart of the data integration service.

Error when you try to create a new table map

Required credentials to fix the issue: The same user that setup dual-write.

You might receive the following error message when you try to configure a new table for dual-write. The only user that can create a map is the user who setup the dual-write connection.

Response status code does not indicate success: 401 (Unauthorized)

Error when you open the dual-write user interface

You might receive the following error message when you try to access dual-write from the **Data management** workspace:

login.microsoftonline.com refused to connect.

To fix the issue, sign in by using an InPrivate window in Microsoft Edge, an incognito window in Chromium, or an incognito window in Google Chrome. You must also unblock or clear third-party cookies.

Error when you link the environment for dual-write or add a new

table mapping

Required role to fix the issue: System administrator in both Finance and Operations apps and Dataverse.

You might encounter the following error when linking or creating maps:

Response status code does not indicate success: 403 (tokenexchange).
Session ID: <your session id>
Root activity ID: <your root activity id>

This error can occur if you don't have sufficient permissions to link dual-write or create maps. This error can also occur if the Dataverse environment was reset without unlinking dual-write. Any user with system administrator role in both Finance and Operations apps and Dataverse can link the environments. Only the user who setup the dual-write connection can add new table maps. After setup, any user with system administrator role can monitor the status and edit the mappings.

Error when you stop the table mapping

You might receive the following error message when you try to stop the table mappings:

[Forbidden], [{"status":403,"source":"","message":"Error from token exchange: User is not allowed to access connection dynamicscrmonline/xxxxxx-xxxx-xxxx-xxxxxxxx"}], The remote server returned an error: (403) Forbidden.

This error occurs when the linked Dataverse environment isn't available.

To fix the issue, create a ticket for the Data Integration team. Attach the network trace so that the Data Integration team can mark the maps as **Not running** in the back end.

Error while trying to start a table mapping

You might receive an error like the following when you try to set that state of a mapping to **Running**:

Unable to complete initial data sync. Error: dual-write failure - plugin registration failed: Unable to build dual-write lookup metadata. Error object reference not set to an instance of an object.

The fix for this error depends on the cause of the error:

- If the mapping has dependent mappings, then make sure to enable the dependent mappings of this table mapping.
- The mapping might be missing source or destination columns. If a column in the Finance and Operations app is missing, then follow the steps in the section Missing table columns issue on maps. If a column in Dataverse is missing, then click **Refresh tables** button on the mapping so that the columns are automatically populated back into the mapping.

Troubleshoot live synchronization issues

2/18/2021 • 4 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

This topic provides troubleshooting information for dual-write integration between Finance and Operations apps and Dataverse. Specifically, it provides information that can help you fix issues with live synchronization.

IMPORTANT

Some of the issues that this topic addresses might require either the system admin role or Microsoft Azure Active Directory (Azure AD) tenant admin credentials. The section for each issue explains whether a specific role or credentials are required.

Live synchronization throws a 403 Forbidden error when you create a row in a Finance and Operations app

You might receive the following error message when you create a row in a Finance and Operations app:

[{\"error\":{\"code\":\"0x80072560\",\"message\":\"The user is not a member of the organization.\"}}], The remote server returned an error: (403) Forbidden."}}".

To fix the issue, follow the steps in System requirements and prerequisites. To complete those steps, the dual-write application users who are created in Dataverse must have the system admin role. The default owning team must also have the system admin role.

Live synchronization for any table consistently throws a similar error when you create a row in a Finance and Operations app

Required role to fix the issue: System admin

You might receive an error message like the following every time that you try to save table data in a Finance and Operations app:

Cannot save the changes to the database. Unit of Work can not commit transaction. Unable to write data to entity uoms. Writes to UnitOfMeasureEntity failed with error message Unable to sync with entity uoms.

To fix the issue, you must make sure that the prerequisite reference data exists in both the Finance and Operations app and Dataverse. For example, if the customer that you're in the Finance and Operations app belongs to a specific customer group, make sure that the customer group exists in Dataverse.

If data exists on both sides, and you've confirmed that the issue isn't data-related, follow these steps.

1. Stop the related table.

2. Sign in to the Finance and Operations app, and make sure that rows for the failing table exist in the DualWriteProjectConfiguration and DualWriteProjectFieldConfiguration tables. For example, here is what the query looks like if the **Customers** table is failing.

```
Select projectname, externalenvironmentURL ,\*
from DUALWRITEPROJECTCONFIGURATION
where INTERNALENTITYNAME = 'Customers V3' and
    EXTERNALENTITYNAME = 'accounts'
```

3. If there are rows for the failing table even after you stop the table mapping, delete the rows that are related to the failing table. Make a note of the **projectname** column in the DualWriteProjectConfiguration table, and fetch the row in the DualWriteProjectFieldConfiguration table by using the project name to delete the row.
4. Start the table mapping. Validate whether the data is synced without any issues.

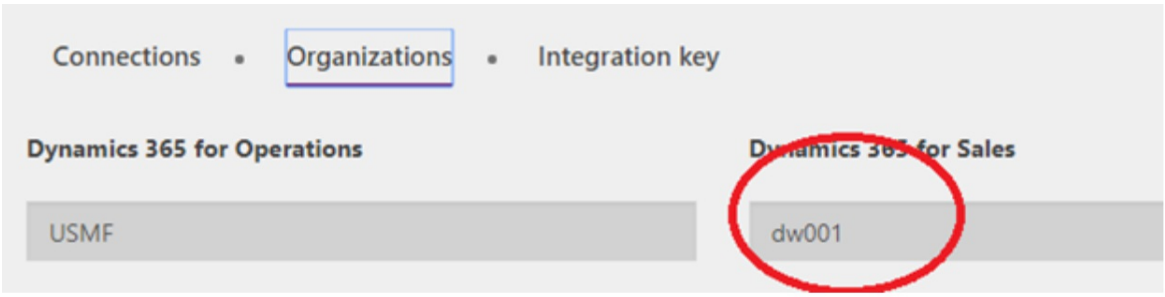
Handle read or write privilege errors when you create data in a Finance and Operations app

You might receive a "Bad Request" error message that resembles the following example when you create data in a Finance and Operations app.

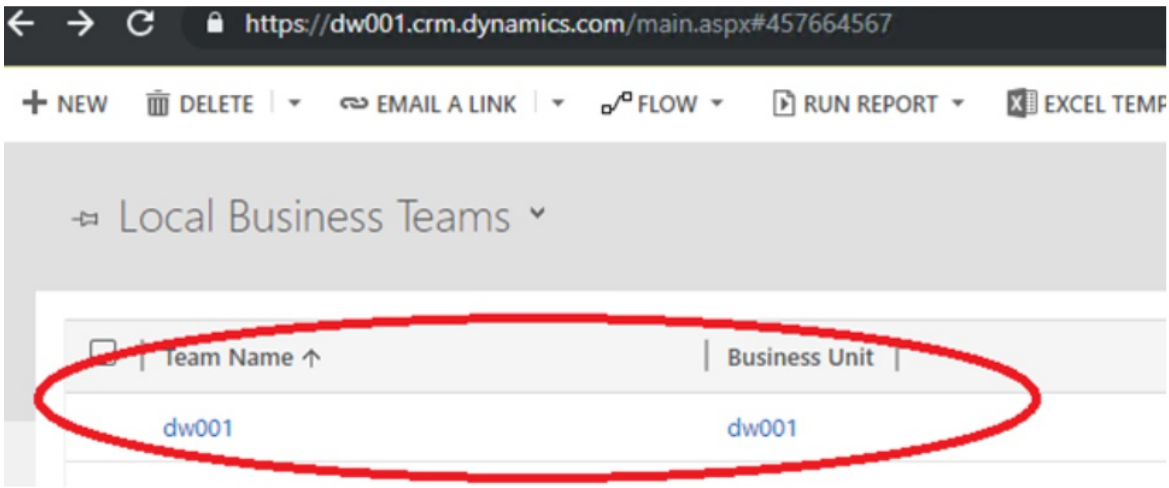
RECORD ID IN SOURCE	SOURCE FIELD WITH ISSUE	ERROR MESSAGE
	POST	Bad Request {"error":{"code":"0x80042f0a","message":"Principal team

To fix the issue, you must assign the correct security role to the team of the mapped Dynamics 365 Sales or Dynamics 365 Customer Service business unit to enable the missing privilege.

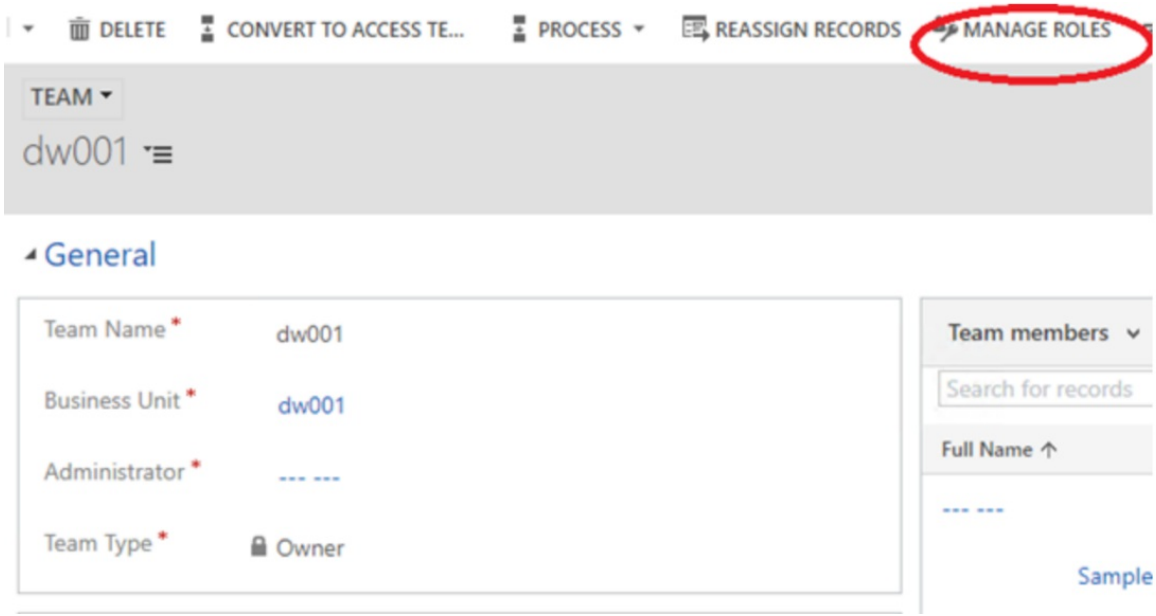
1. In the Finance and Operations app, find the business unit that is mapped in the Data Integration connection set.



2. Sign in to the environment in the model-driven app in Dynamics 365, navigate to **Setting > Security**, and find the team of the mapped business unit.



3. Open the page for the team for editing, and then select **Manage roles** to open the **Manage Team Roles** dialog box.



4. Assign the role that has the read/write privilege for the relevant tables, and then select **OK**.

Fix synchronization issues in an environment that has a recently changed Dataverse environment

Required role to fix the issue: System admin

You might receive the following error message when you create data in a Finance and Operations app:

*{"entityName":"CustCustomerV3Entity","executionStatus":2,"fieldResponses":[],"recordResponses":[{"errorMessage":"**Unable to generate payload for entity CustCustomerV3Entity**","logDateTime":"2019-08-27T18:51:52.5843124Z","verboseError":"Payload creation failed with error Invalid URI: The URI is empty."}],"isErrorCountUpdated":true}*

Here is what the error looks like in the model-driven app in Dynamics 365:

An unexpected error occurred from ISV code. (ErrorType = ClientError) Unexpected exception from plug-in (Execute): Microsoft.Dynamics.Integrator.DualWriteRuntime.Plugins.PostCommitPlugin: System.Exception: failed to process entity account - (A connection attempt failed because the connected party did not properly respond after a period of time, or established connection failed because connected host has failed to respond

This error occurs when the Dataverse environment is incorrectly reset at the same time that you try to create

data in the Finance and Operations app.

To fix the issue, follow these steps.

1. Sign in to the Finance and Operations virtual machine (VM), open SQL Server Management Studio (SSMS), and look for rows in the DUALWRITEPROJECTCONFIGURATIONENTITY table where **internalentityname** equals **Customers V3** and **externalentityname** equals **accounts**. Here is what the query looks like.

   ```
   select projectname, externalenvironmentURL ,\*
   from DUALWRITEPROJECTCONFIGURATION
   where INTERNALENTITYNAME = 'Customers V3' and EXTERNALENTITYNAME = 'accounts'
   ```

2. Use the project name from the results of the previous query to run the following query.

   ```
   select \*
   from DUALWRITEPROJECTFIELDCONFIGURATION
   where projectname = <project name from previous query>
   ```

3. Make sure that the **externalenvironmentURL** column has the correct Dataverse or app URL. Delete any duplicate rows that point to the wrong Dataverse URL. Delete the corresponding rows in the DUALWRITEPROJECTFIELDCONFIGURATION and DUALWRITEPROJECTCONFIGURATION tables.

4. Stop the table mapping, and then restart it

Troubleshoot issues related to solution awareness

2/18/2021 • 2 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

This topic provides troubleshooting information for dual-write integration between Finance and Operations apps and Dataverse. Specifically, it provides information that can help you fix issues that are related to solution awareness.

IMPORTANT

Some of the issues that this topic addresses might require either the system admin role or Microsoft Azure Active Directory (Azure AD) tenant admin credentials. The section for each issue explains whether a specific role or credentials are required.

Error on the Dual-write page

On the **Dual-write** page, you might receive an error message that resembles the following example:

The entity with a name 'msdyn_dualwriteentitymap' with namemapping='Logical' was not found in the MetadataCache.

To fix the issue, make sure that the dual-write core solution is installed in Dataverse. The dual-write core solution is a prerequisite for solution awareness.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Troubleshoot issues from upgrades of Finance and Operations apps

2/18/2021 • 3 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

This topic provides troubleshooting information for dual-write integration between Finance and Operations apps and Dataverse. Specifically, it provides information that can help you fix issues that are related to upgrades of Finance and Operations apps.

IMPORTANT

Some of the issues that this topic addresses might require either the system admin role or Microsoft Azure Active Directory (Azure AD) tenant admin credentials. The section for each issue explains whether a specific role or credentials are required.

Database synchronization errors

Required role to fix the issue: System admin

You might receive an error message that resembles the following example when you try to use the **DualWriteProjectConfiguration** table to update a Finance and Operations app to Platform update 30.

```
Infolog diagnostic message: 'Cannot select a row in Dual write project sync (DualWriteProjectConfiguration).
The SQL database has issued an error.' on category 'Error'. 10/28/2019 15:18:20: Infolog diagnostic message:
'Object Server Database Synchronizer: ' on category 'Error'. 10/28/2019 15:18:20: Infolog diagnostic
message: '[Microsoft][ODBC Driver 17 for SQL Server][SQL Server]Invalid column name 'ISDELETE'.' on category
'Error'. 10/28/2019 15:18:20: Infolog diagnostic message: 'SELECT
T1.PROJECTNAME,T1.EXTERNALENTITYNAME,T1.INTERNALENTITYNAME,T1.EXTERNALENVIRONMENTURL,T1.STATUS,T1.ENABLEBATC
HLOOKUP,T1.PARTITIONMAP,T1.QUERYFILTEREXPRESSION,T1.INTEGRATIONKEY,T1.ISDELETE,T1.ISDEBUGMODE,T1.RECVERSION,
T1.PARTITION,T1.RECID FROM DUALWRITEPROJECTCONFIGURATION T1 WHERE (PARTITION=5637144576)' on category
'Error'. 10/28/2019 15:18:20: Infolog diagnostic message: 'session 1043 (Admin)' on category 'Error'.
10/28/2019 15:18:20: Infolog diagnostic message: 'Stack trace: Call to TTSCOMMIT without first calling
TTSBEGIN.' on category 'Error'.
10/28/2019 15:18:20: Application configuration sync failed.
Microsoft.Dynamics.AX.Framework.Database.TableSyncException: Custom action threw exception(s), please
investigate before synchronizing again: 'InfoException:Stack trace: Call to TTSCOMMIT without first calling
TTSBEGIN."
```

To fix the issue, follow these steps.

1. Sign in to the virtual machine (VM) for the Finance and Operations app.
2. Open Visual Studio as an admin, and open the Application Object Tree (AOT).
3. Search for **DualWriteProjectConfiguration**.

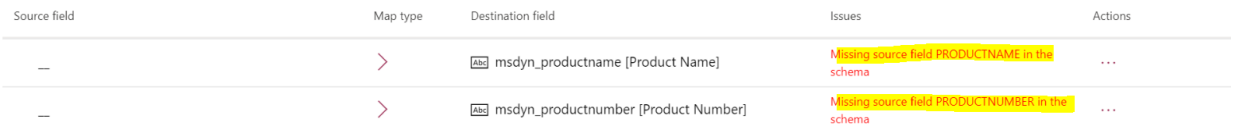
4. In the AOT, right-click **DualWriteProjectConfiguration**, and select **Add to new project**. Select **OK** to create the new project that uses default options.
5. In Solution Explorer, right-click **Project properties**, and set **Synchronize Database on Build** to **True**.
6. Build the project, and confirm that the build is successful.
7. On the **Dynamics 365** menu, select **Synchronize database**.
8. Select **Synchronize** to do a full database synchronization.
9. After the full database synchronization is successful, rerun the database synchronization step in Microsoft Dynamics Lifecycle Services (LCS) and use the manual upgrade scripts as applicable, so that you can proceed with the update.

Missing table columns issue on maps

Required role to fix the issue: System admin

On the **Dual-write** page, you might receive an error message that resembles the following example:

Missing source field <field name> in the schema.



To fix the issue, first follow these steps to make sure that the columns are in the table.

1. Sign in to the VM for the Finance and Operations app.
2. Go to **Workspaces > Data management**, select the **Framework parameters** tile, and then, on the **Table settings** tab, select **Refresh table list** to refresh the tables.
3. Go to **Workspaces > Data management**, select the **Data tables** tab, and make sure that the table is listed. If the table isn't listed, sign in to the VM for the Finance and Operations app, and make sure the table is available.
4. Open the **Table mapping** page from the **Dual-write** page in the Finance and Operations app.
5. Select **Refresh table list** to automatically fill the columns in the table mappings.

If the issue still isn't fixed, follow these steps.

> **IMPORTANT**
> These steps guide you through the process of deleting a table and then adding it again. To avoid issues, be sure to follow the steps exactly.

1. In the Finance and Operations app, go to **Workspaces > Data management**, and select the **Data tables** tile.
2. Find the table that is missing the attribute. Click **Modify target mapping** in the toolbar.
3. On the **Map staging to target** pane, click **Generate mapping**.
4. Open the **Table mapping** page from the **Dual-write** page in the Finance and Operations app.
5. If the attribute is not auto-populated on the map, add it manually by clicking **Add attribute** button and then clicking **Save**.
6. Select the map and click **Run**.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

General troubleshooting

2/18/2021 • 4 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

This topic provides general troubleshooting information for dual-write integration between Finance and Operations apps and Dataverse.

IMPORTANT

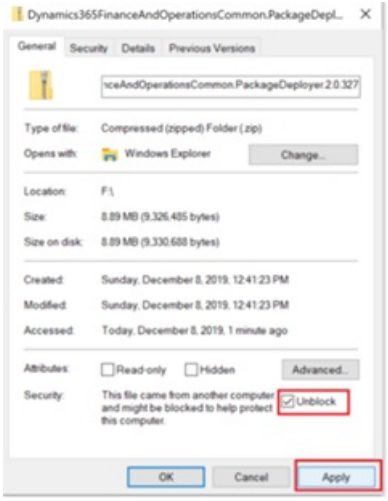
Some of the issues that this topic addresses might require either the system admin role or Microsoft Azure Active Directory (Azure AD) tenant admin credentials. The section for each issue explains whether a specific role or credentials are required.

When you try to install the dual-write package by using the package deployer tool, no available solutions are shown

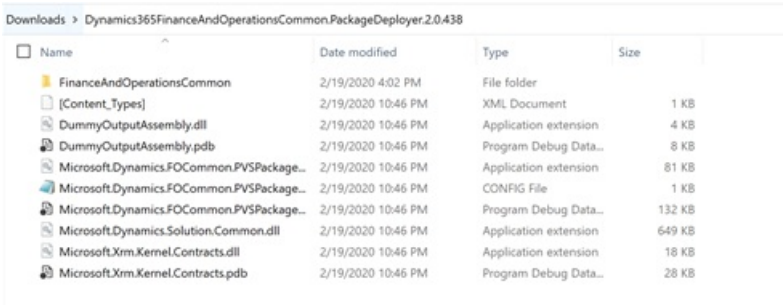
Some versions of the package deployer tool are incompatible with the dual-write solution package. To successfully install the package, be sure to use version 9.1.0.20 or later of the package deployer tool.

After you install the package deployer tool, install the solution package by following these steps.

1. Download the latest solution package file from Yammer.com. After the package zip file is downloaded, right-click it, and select **Properties**. Select the **Unblock** check box, and then select **Apply**. If you don't see the **Unblock** check box, the zip file is already unblocked, and you can skip this step.

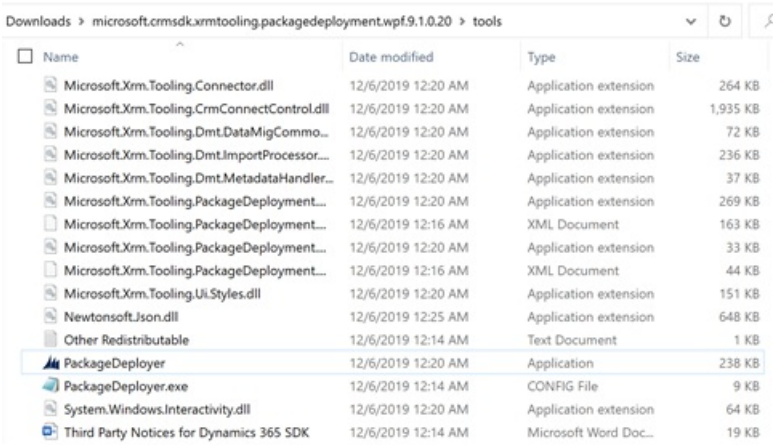


2. Extract the package zip file, and copy all the files in the **Dynamics365FinanceAndOperationsCommon.PackageDeployer.2.0.438** folder.



3. Paste all the copied files into the **Tools** folder of the package deployer tool.

4. Run **PackageDeployer.exe** to select the Dataverse environment and install the solutions.



Enable and view the plug-in trace log in Dataverse to view error details

Required role to turn on the trace log and view errors: System admin

To turn on the trace log, follow these steps.

1. Sign in to the model-driven app in Dynamics 365, open the **Settings** page, and then, under **System**, select **Administration**.
2. On the **Administration** page, select **System Settings**.
3. On the **Customization** tab, in the **Plug-in and custom workflow activity tracing** column, select **All** to enable the plug-in trace log. If you want to log trace logs only when exceptions occur, you can select **Exception** instead.

To view the trace log, follow these steps.

1. Sign in to the model-driven app in Dynamics 365, open the **Settings** page, and then, under **Customization**, select **Plug-in Trace Log**.
2. Find the trace logs where the **Type Name** column is set to **Microsoft.Dynamics.Integrator.DualWriteRuntime.Plugins.PreCommmitPlugin**.
3. Double-click an item to view the full log, and then, on the **Execution** FastTab, review the **Message Block** text.

Enable debug mode to troubleshoot live synchronization issues in Finance and Operations apps

Required role to view the errors: System admin Dual-write errors that originate in Dataverse can appear in the Finance and Operations app. In some cases, the full text of the error message isn't available because the message is too long or contains personally identifying information (PII). You can turn on verbose logging for

errors by following these steps.

1. All project configurations in Finance and Operations apps have an **IsDebugMode** property in the **DualWriteProjectConfiguration** table. Open the **DualWriteProjectConfiguration** table by using the Excel add-in.

   > **TIP**
   > An easy way to open the table is to turn on **Design** mode in the Excel add-in and then add **DualWriteProjectConfigurationEntity** to the worksheet. For more information, see Open table data in Excel and update it by using the Excel add-in.

2. Set the **IsDebugMode** property to **Yes** for the project.
3. Run the scenario that is generating errors.
4. The verbose logs are available in the DualWriteErrorLog table. To look up data in the table browser, use the following URL (replace **XXX** as appropriate):

   ```
   https://XXXaos.cloudax.dynamics.com/?mi=SysTableBrowser&tableName=DualWriteErrorLog
   ```

Check synchronization errors on the virtual machine for the Finance and Operations app

Required role to view the errors: System administrator

1. Sign in to Microsoft Dynamics Lifecycle Services (LCS).
2. Open the LCS project that you chose to do the dual-write testing for.
3. Select the **Cloud-hosted environments** tile.
4. Use Remote Desktop to sign in to the virtual machine (VM) for the Finance and Operations app. Use the local account that is shown in LCS.
5. Open Event viewer.
6. Select **Applications and Services Logs > Microsoft > Dynamics > AX-DualWriteSync > Operational**.
7. Review the list of recent errors.

Unlink and link another Dataverse environment from a Finance and Operations app

Required role to unlink the environment: System administrator for either Finance and Operations app or Dataverse.

1. Sign in to the Finance and Operations app.
2. Go to **Workspaces > Data management**, and select the **Dual Write** tile.
3. Select all running mappings, and then select **Stop**.
4. Select **Unlink environment**.
5. Select **Yes** to confirm the operation.

You can now link a new environment.

Unable to view the sales order line Information form

When you create a sales order in Dynamics 365 Sales, clicking on **+ Add products** might redirect you to the Dynamics 365 Project Operations order line form. There is no way from that form to view the sales order line

Information form. The option for **Information** does not appear in the dropdown below **New Order Line**. This happens because Project Operations has been installed in your environment.

To re-enable the **Information** form option, follow these steps:

1. Navigate to the **Order Line** table.
2. Find the **Information** form under the forms node.
3. Select the **Information** form and click **Enable security roles**.
4. Change the security setting to **Display to everyone**.

Dual-write FAQ

2/18/2021 • 9 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

This topic lists frequently asked questions about dual-write and provides brief answers to help you quickly get the information that you require.

Dual-write setup

Do you plan to enable dual-write to use Dataverse as a hub between multiple Finance and Operations environments? If Dataverse is used as a hub, data can be synced between two or more Finance and Operations environments.

The current plan of row is to restrict dual-write to a one-to-one (1:1) mapping between a single Finance and Operations environment and a single Dataverse environment.

Can I control the sequencing of maps in dual-write, as I can in Data integrator?

Dual-write is transaction-based. For example, if a change in a Finance and Operations app triggers synchronization of multiple maps with Dataverse, by default, those changes will be sequenced in the order in which they are updated in the database. This pattern makes more sense in the context of initial synchronization. The system provides related table maps in a specified order, and you can reorder the list so that it best suits your environment.

Do application users require any special permissions to enable or configure dual-write?

You must have two Azure Active Directory (Azure AD) applications set up for the Finance and Operations environment and two application users set up in the Dataverse environment. These application users should contain the appropriate application IDs. For the connection to work properly, you must give the applications the relevant table permissions by using a security role. For more information, see Verify requirements and grant access.

Do end users require any special permissions to enable or configure dual-write?

End users who are configuring dual-write mappings should have System Administrator security roles assigned in both Dataverse and Finance and Operations environments.

Dual-write mappings can be accessed by multiple users, as long as all the users and environments belong to a single tenant, and the user has the required security and licenses assignment.

I have multiple legal entities. Some of my maps are legal table–specific or valid for only some of the legal entities. What is the best way to address this requirement? Can I apply a filter such as Company = USMF to address it?

Legal table mapping can be done when the Dataverse environment is linked. You can't map table maps to a specific legal entity.

If dual-write solutions are installed in Dataverse, can I uninstall them?

Dual-write solutions are managed solutions that can be uninstalled. However, when a managed solution is uninstalled, all components in the solution are deleted. Any data that is stored in the components is also deleted. For more information, see Maintain managed solutions.

I have data in both a customer engagement app and a Finance and Operations app, and I bootstrap my existing data in the customer engagement app. If my data isn't currently aligned, can I specify a master source for the initialization run, so that all differences are applied to the target?

After the bootstrapping is done, you can configure the initial synchronization to apply differences and select a master. For more information about bootstrapping, see Bootstrap with company data FAQ. For more information about the initial synchronization, see Enable table maps for dual-write.

Dual-write administration and management

What is the purpose of the integration key, and is it mandatory?

The integration key is the natural key that uniquely identifies rows. Integration keys are required only for Dataverse tables. You can manually create an integration key in dual-write. An integration key can also be automatically created from the table's alternate keys, if an alternate key is already provided for the table. Integration keys are used for the same purpose as alternate keys: they provide an efficient and accurate way to integrate data with external systems. Integration keys are essential in cases where an external system doesn't store the globally unique identifiers (GUIDs) that uniquely identify rows in Dataverse.

Dual-write uses integration keys to uniquely identify rows, by using one or more table column values that represent a unique combination. For example, to identify an account row by using an integration key, you can use the account number column. Alternatively, you can use the account number column together with other columns that have values that should not change. For more information, see Define alternate keys using Power Apps portal.

It's important that keys be matched between the Finance and Operations environment and the Dataverse environment. Otherwise, issues might occur during the initial synchronization phase.

How do I move table maps between environments? Is version control supported for table maps?

You can export maps and then import them into a different environment. You can automate the process by using Azure DevOps. You can have version control on your dual-write mappings, because the mappings are solution-aware components. For more information, see Update table maps and export them to other environments as a solution.

Where can I find examples and patterns for filtering dual-write maps?

For basic filtering examples, see Filter your data.

For more advanced examples for Dataverse, see Filter results. Nested lookup isn't supported in dual-write source filters. Only standard filter operators directly against table columns are supported.

For more advanced Finance and Operations filters, see Using Expressions in Query Ranges and Advanced filtering and query syntax.

Dual-write live synchronization introduces tight coupling across applications. What happens if one side fails? Will the other side fail too?

When the integration is in live sync mode, if the sync fails on one of the apps, then the other app will fail as well and users will receive an error. When the integration is paused, changes are staged. They are then written when the target system is up and running. For more information about how to automatically pause integrations, see Alert notifications

When live synchronization is paused and then resumed, does it follow the sequence of changes? For example, if the Name column in the Finance and Operations app is changed from NameA to NameB to NameC, is customer engagement data changed from NameA to NameB to NameC, or is it changed directly from NameA to NameC?

The integration follows the complete sequence of changes. In the example, the customer engagement app data is changed from **NameA** to **NameB** to **NameC**.

How do I handle a Finance and Operations database transfer from PROD to STAGE? What is the effect on dual-write? After the transfer, the systems are no longer in sync. Is the synchronization done automatically?

Each linked environment-pair (Finance and Operations apps environment and Dataverse environment) should be treated as a single unit and refreshed accordingly. For example, if you are refreshing a sandbox from production, then both Finance and Operations app sandbox environment and the Dataverse sandbox environment should be refreshed from their production counterparts. If dual-write is already used in target environments, those environments need to be unlinked. After the data refresh on target environments, these tables should be cleaned up:

- Finance and Operations apps tables: **DualWriteProjectConfiguration**, **DualWriteProjectFieldConfiguration**, and **BusinessEventsDefinition**.
- Dataverse tables: **DualwriteRuntimeConfiguration**.

The environments need to be relinked and maps reactivated manually.

I need real-time integration, and I want to move some tables or scenarios from Data integrator to dual-write. How do I migrate, and what are the implications of changing my integration pattern?

For information about how to migrate Prospect to cash to dual-write, see Migrating data from Data Integrator to Dual Write. In general, three things might change during migration:

- Manual migration of the maps from Data integrator to dual-write
- Table changes, because of the absence of advanced query capabilities
- Data migration, because of adaptation to new concepts such as company striping

On Finance and Operations data tables, can I develop unbounded columns that flow to Dataverse by using dual-write?

Yes. You can use both computed columns and virtual columns. However, you should monitor the performance overhead from the additional X++ logic that is required for reads and writes. Round-tripping within the same transaction isn't allowed. Therefore, you should avoid using virtual columns to transform or calculate additional values through X++ and expect that to go back to Dataverse within the same transaction.

When I use the Dataverse offline app, what happens if I can't sync the data after reconnection? Does this situation cause an inconsistent state between the Dataverse environment and the Finance and Operations environment?

You can interact with Dataverse data offline when using the Dynamics 365 for phones app or the Field Service Mobile app in offline mode. In both apps, data is stored offline and can be synced with the server at your discretion. If there are errors when the offline data is synced with the server, and updates can't be done because the other environment is failing, data sync will fail, and Dataverse will not be updated. When the integration is paused, you can re-run the sync and save your updates on the server. These changes will be staged and then synced with the Finance and Operations environment when the mapping is up and running again. For more information, see Run model-driven apps and canvas apps on Power Apps mobile.

Mapping concepts between apps

How are number sequences handled? For example, the customer account number is automatically generated in Finance and Operations apps, but it's added manually in customer engagement apps.

Number sequences for Finance and Operations apps and customer engagement apps aren't connected. In a scenario that involves a multi-mastered table, you must either plan for separate number sequence formats or create a range for each app. Here are some examples:

- In the Finance and Operations app, use **F0001, F0002, F0003**. In the customer engagement app, use **C0001, C0002, C0003**.

- In the Finance and Operations app, use **US0001 to US4999**. In the customer engagement app, use **US5000 to US9999**.

If a table is created in only one system, set up the number sequence in the source app only. For more information, see Autonumber columns.

Can I map a company-specific table in a customer engagement app with a global table in a Finance and Operations app, or a global table in a customer engagement app with a company-specific table in a Finance and Operations app?

Dual-write supports mappings only between cross-company tables or company-specific tables from both sides.

How do I make a company-specific table in Dataverse?

You can make Dataverse custom tables company-specific by adding a many-to-one (N:1) relationship between your custom tables and the out-of-box company table. You should also include the company foreign key as part of the table key. For more information, see Company concept in Dataverse.

To enable table maps for dual-write, you must define an alternate key in Dataverse. The value of the alternative key in Dataverse must match the key that is defined in the Finance and Operations app. For more information, see Criteria for linking tables.

Is there a document about best practices for table usage? Should I use Customers V2, Customers V3, or Customer Details? What is the difference between these tables, and what is the use case for each?

You should use the out-of-box scenarios if you can, because they cover common scenarios such as customer/vendor integration.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Prospect to cash

2/18/2021 • 2 minutes to read • Edit Online

The Prospect to cash solution provides direct synchronization across Dynamics 365 Supply Chain Management and Dynamics 365 Sales. The Prospect to cash templates that are available with the Data Integration feature enable the flow of data for accounts, contacts, products, sales quotations, sales orders, and sales invoices. While data is flowing, you can perform sales and marketing activities in Sales, and you can handle order fulfillment by using inventory management in Supply Chain Management.

For more information about the Prospect to cash integration, watch the short YouTube video Prospect to cash integration.

In the current version, the Prospect to cash solution provides the following types of direct synchronization:

- Synchronize accounts directly from Sales to customers in Supply Chain Management
- Synchronize products directly from Supply Chain Management to products in Sales
- Synchronize contacts directly from Sales to contacts or customers in Supply Chain Management
- Synchronize sales quotation headers and lines directly from Sales to Supply Chain Management
- Synchronization of sales orders directly between Sales and Supply Chain Management
- Synchronize sales invoice headers and lines directly from Supply Chain Management to Sales

System requirements for Supply Chain Management

Prospect to cash integration is supported on the following versions:

Microsoft Dynamics 365 for Finance and Operations, Enterprise edition 7.3 (December 2017)

- Dynamics 365 for Finance and Operations, Enterprise edition (December 2017) - Application build 7.3.11971.56116 with Platform Update 12 (7.0.4709.41129)

Dynamics 365 for Finance and Operations, Enterprise edition (July 2017)

- Dynamics 365 for Finance and Operations, Enterprise edition (July 2017) - with platform update 8 (application build 7.2.11792.56024 with platform build 7.0.4565.16212).
- The following hotfixes are required:
  - **KB4045570** – This hotfix enables sales order synchronization from Sales to Supply Chain Management via the Data Integration feature. It also provides several other enhancements.
  - **KB4036524** – This hotfix enables sales order line synchronization from Supply Chain Management to Sales via the Data Integration feature.
  - **KB4036461** – This hotfix enables sales order synchronization from Supply Chain Management to Sales via the Data Integration feature.

    > **NOTE**
    > You only have to install KB4045570 because the installation includes the changes from other hotfixes.

Dynamics 365 for Finance and Operations version 1611 (November 2016)

- Dynamics 365 for Finance and Operations version 1611 (November 2016) with platform update 8 or higher

- The following hotfixes are required:
  - KB4051266 - Enable sales order synchronization with Data integrator from Supply Chain Management to Sales.
  - KB4037542 - Enable sales order header and line synchronization with Data integrator from Supply Chain Management to Sales.
  - KB4033093 - Support for prospect to cash integration through data entities is required.

    NOTE

    After you install the hotfixes, you must trigger the following batch job from the **SalesPopulateProspectToCash** form. This form is hidden because you only need it once. To access the form, log in to the environment and add the following to the URL in your browser address: *&mi=action:SalesPopulateProspectToCash*, for example, `https://ax123456.cloud.test.dynamics.com/?cmp=USMF&mi=action:SalesPopulateProspectToCash`. When the form opens, click OK. This will populate a new **LineCreationSequnceNumber** field in the **SalesLine**, **SalesQuotationLine**, and **CustInvoiceTrans** tables with unique values, and the product list will be refreshed. This is required for the Prospect to cash integration to work.

System requirements for Sales

To use the Prospect to cash solution, you must install the following components:

- Dynamics 365 Sales version 1612 (8.2.1.207) (DB 8.2.1.207) online or a later version
- Prospect to cash solution for Dynamics 365 Sales, version 1.15.0.0 or a later version. The solution is available for download from AppSource. Download Dynamics 365, Prospect to Cash.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Synchronize accounts directly from Sales to customers in Supply Chain Management

2/18/2021 • 4 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

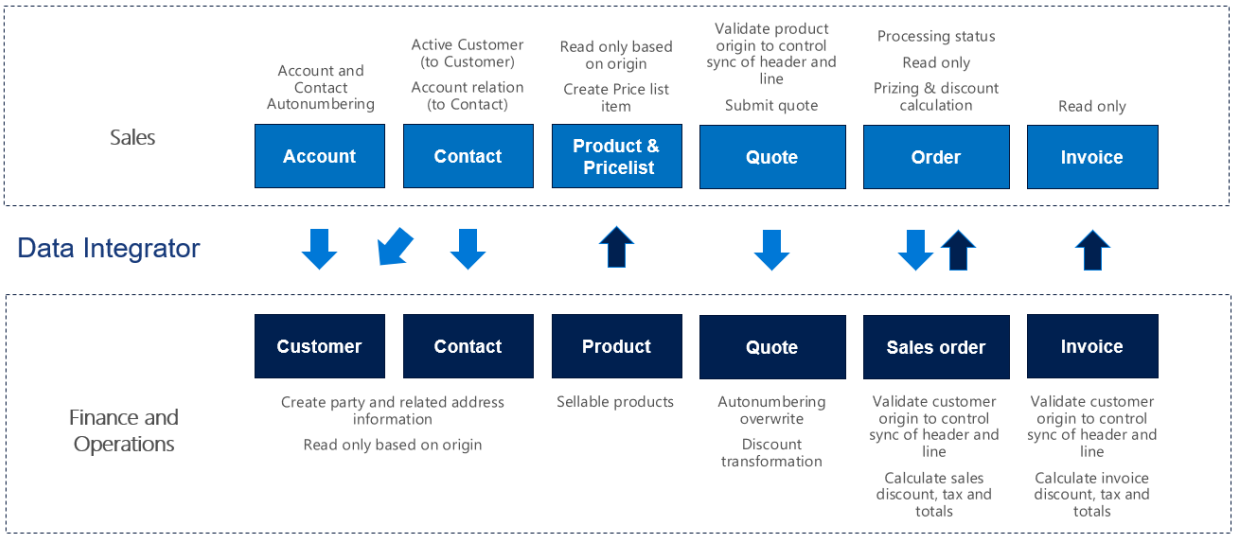
NOTE

Before you can use the Prospect to cash solution, you should be familiar with Integrate data into Microsoft Dataverse for Apps.

This topic discusses the templates and underlying tasks that are used to synchronize accounts directly from Dynamics 365 Sales to Dynamics 365 Supply Chain Management.

Data flow in Prospect to cash

The Prospect to cash solution uses the Data integration feature to synchronize data across instances of Supply Chain Management and Sales. The Prospect to cash templates that are available with the Data integration feature enable the flow of data about accounts, contacts, products, sales quotations, sales orders, and sales invoices between Supply Chain Management and Sales. The following illustration shows how the data is synchronized between Supply Chain Management and Sales.



Templates and tasks

To access the available templates, open Power Apps Admin Center. Select **Projects**, and then, in the upper-right corner, select **New project** to select public templates.

The following template and underlying task are used to synchronize accounts from Sales to Supply Chain Management:

- **Name of the template in Data integration:** Accounts (Sales to Fin and Ops) - Direct
- **Name of the task in the project:** Accounts - Customers

No synchronization tasks are required before Account/Customer synchronization can occur.

Entity set

SALES	SUPPLY CHAIN MANAGEMENT
Accounts	Customers V2

Entity flow

Accounts are managed in Sales and synchronized to Supply Chain Management as customers. The **Is Externally Maintained** property on these customers is set to **Yes** to track customers that originate from Sales. During invoicing, this information is used to filter invoices that are synchronized to Sales.

Prospect to cash solution for Sales

The **Account Number** column is available on the **Account** page. It has been made a natural and unique key in order to support the integration. The natural key feature of the Customer Relationship Management (CRM) solution might affect customers who already use the **Account Number** column, but who don't use unique **Account Number** values per account. Currently, the integration solution doesn't support this case.

When a new account is created, if an **Account Number** value doesn't already exist, it's automatically generated by using a number sequence. The value consists of **ACC**, followed by an increasing number sequence and then a suffix of six characters. Here is an example: **ACC-01000-BVRCPS**

When the integration solution for Sales is applied, an upgrade script sets the **Account Number** column for existing accounts in Sales. If there are no **Account Number** values, the number sequence that was mentioned earlier is used.

Preconditions and mapping setup

- The **CustomerGroupId** mapping must be updated to a valid value in Supply Chain Management. You can specify a default value, or you can set the value by using a value map.

  The default template value is **10**.

- By adding the following mappings, you can help reduce the number of manual updates that are required in Supply Chain Management. You can use a default value or a value map from, for example, **Country/Region** or **City**.

  - **SiteId** – A site is required in order to generate quotations and sales order lines in Supply Chain Management. A default site can be taken either from the product, or from the customer from the order header.

    The default template value is **1**.

  - **WarehouseId** – A warehouse is required in order to process quotations and sales order lines in Supply Chain Management. A default warehouse can be taken either from the product, or from the customer from the order header in Supply Chain Management.

The default template value is **13**.

- **LanguageId** – A language is required in order to generate quotations and sales orders in Supply Chain Management. By default, the language from the order header from the customer is used.

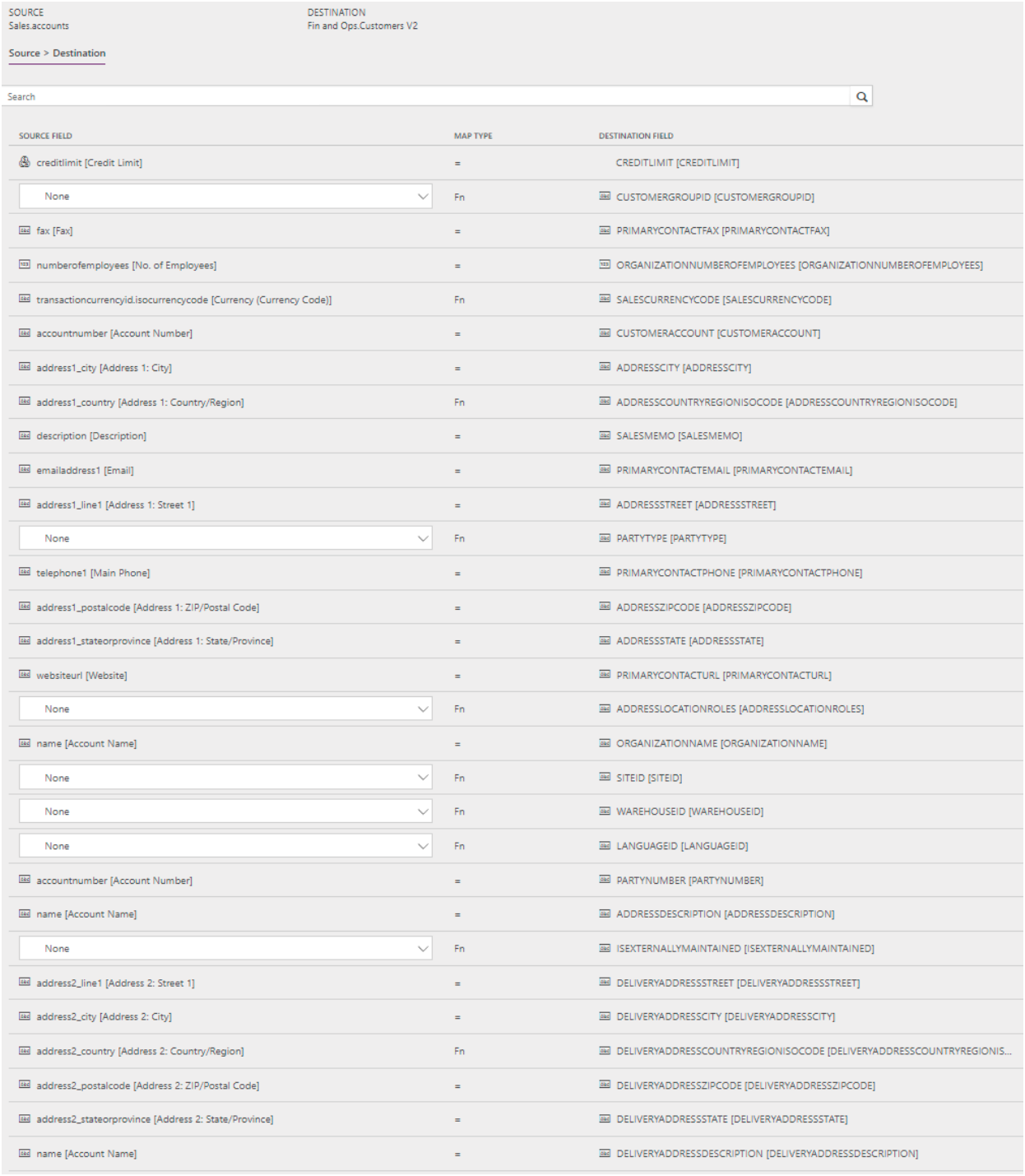
  The default template value is **en-us**.

Template mapping in Data integration

> **NOTE**
>
> The **Payment terms**, **Freight terms**, **Delivery terms**, **Shipping method**, and **Delivery mode** columns aren't included in the default mappings. To map these columns, you must set up a value mapping that is specific to the data in the organizations that the table is synchronized between.

The following illustrations show an example of a template mapping in Data integration.

> **NOTE**
>
> The mapping shows which column information will be synchronized from Sales to Supply Chain Management.



Related topics

Prospect to cash

Synchronize accounts directly from Sales to customers in Supply Chain Management

Synchronize contacts directly from Sales to contacts or customers in Supply Chain Management

Synchronization of sales orders directly between Sales and Supply Chain Management

Synchronize sales invoice headers and lines directly from Supply Chain Management to Sales

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Synchronize products directly from Supply Chain Management to products in Sales

2/18/2021 • 4 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

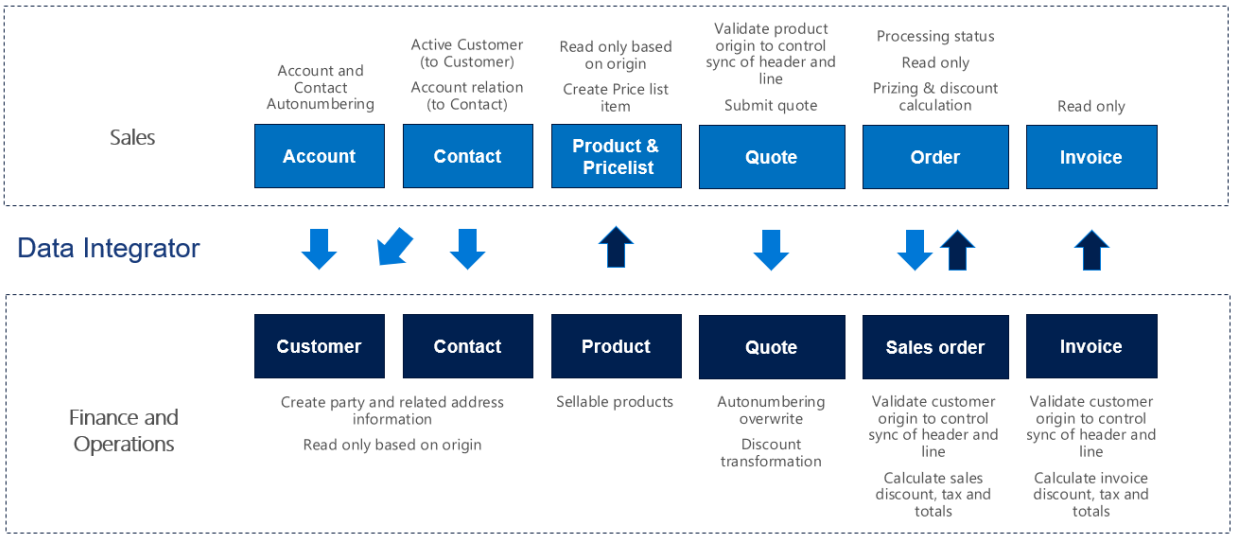
NOTE

Before you can use the Prospect to cash solution, you should be familiar with Integrate data into Microsoft Dataverse for Apps.

This topic discusses the templates and underlying tasks that are used to synchronize products directly from Dynamics 365 Supply Chain Management to Dynamics 365 Sales.

Data flow in Prospect to cash

The Prospect to cash solution uses the Data integration feature to synchronize data across instances of Supply Chain Management and Sales. The Prospect to cash templates that are available with the Data integration feature enable the flow of data about accounts, contacts, products, sales quotations, sales orders, and sales invoices between Supply Chain Management and Sales. The following illustration shows how the data is synchronized between Supply Chain Management and Sales.



Templates and tasks

To access the available templates, open Power Apps Admin Center. Select **Projects**, and then, in the upper-right corner, select **New project** to select public templates.

The following template and underlying tasks are used to synchronize products from Supply Chain Management to Sales.

- **Name of the template in Data integration:** Products (Supply Chain Management to Sales) - Direct
- **Name of the task in the Data integration project:** Products

No synchronization tasks are required before product synchronization can occur.

Entity set

SUPPLY CHAIN MANAGEMENT	SALES
Sellable released products	Products

Entity flow

Products are managed in Supply Chain Management and synchronized to Sales. The **Sellable released products** data entity in Supply Chain Management exports only products that are *sellable*. Sellable products are products that have the information that they require in order to be used on a sales order. The same rules apply when a product is validated by using the **Validate** function on the **Released product** page.

The product number is used as a key. Therefore, when product variants are synchronized to Sales, each product variant has an individual product ID.

Prospect to cash solution for Sales

In Sales, a new **Is Externally Maintained** field has been added on products to indicate that a given product is maintained externally. By default, the value is set to **Yes** during an import to Sales. The following values are available:

- **Yes** – The product originated from Supply Chain Management and won't be editable in Sales.
- **No** – The product was entered directly in Sales.
- **(Blank)** – The product existed in Sales before the Prospect to cash solution was enabled.

The **Is Externally Maintained** field helps ensure that only quotations and sales orders that have externally maintained products will be synchronized to Supply Chain Management.

Externally maintained products are automatically added to the first valid price list that has the same currency. Price lists are organized alphabetically by name. The product sales price from Supply Chain Management is used as the price on the price list. Therefore, there must be a price list in Sales for every product sales currency in Supply Chain Management. The currency on the released sellable products is set to the accounting currency in the legal entity that the product is exported from.

> **NOTE**
>
> - Product synchronization will not succeed unless there is a price list that has a matching currency.
> - You can control the used price list with the integration by mapping the pricelevelid.name [Default Price List (Name)] in the Data Integration project. The input has to be in all lowercase letters. For example, the default for a price list in Sales named 'Standard' would be: Destination field: pricelevelid.name [Default Price List (Name)] and Map type: [{ "transformType": "Default", "defaultValue": "standard" }].

Preconditions and mapping setup

- Before you run the synchronization for the first time, you must fill the Distinct product table for existing

products in Supply Chain Management. Existing products won't be synchronized until this job is completed.

1. In Supply Chain Management, use Search to search for **Populate distinct product table**.
2. Select **Populate distinct product table** to run the job. This job must be run only one time.

- Make sure that the required value map for the selling unit of measure (UOM) between Supply Chain Management and Sales exists in the mapping of **SalesUnitSymbol** to **DefaultUnit (Name)**.
- Update the value map for **Unit group** (**defaultuomscheduleid.name**) so that it matches **Unit groups** in Sales.

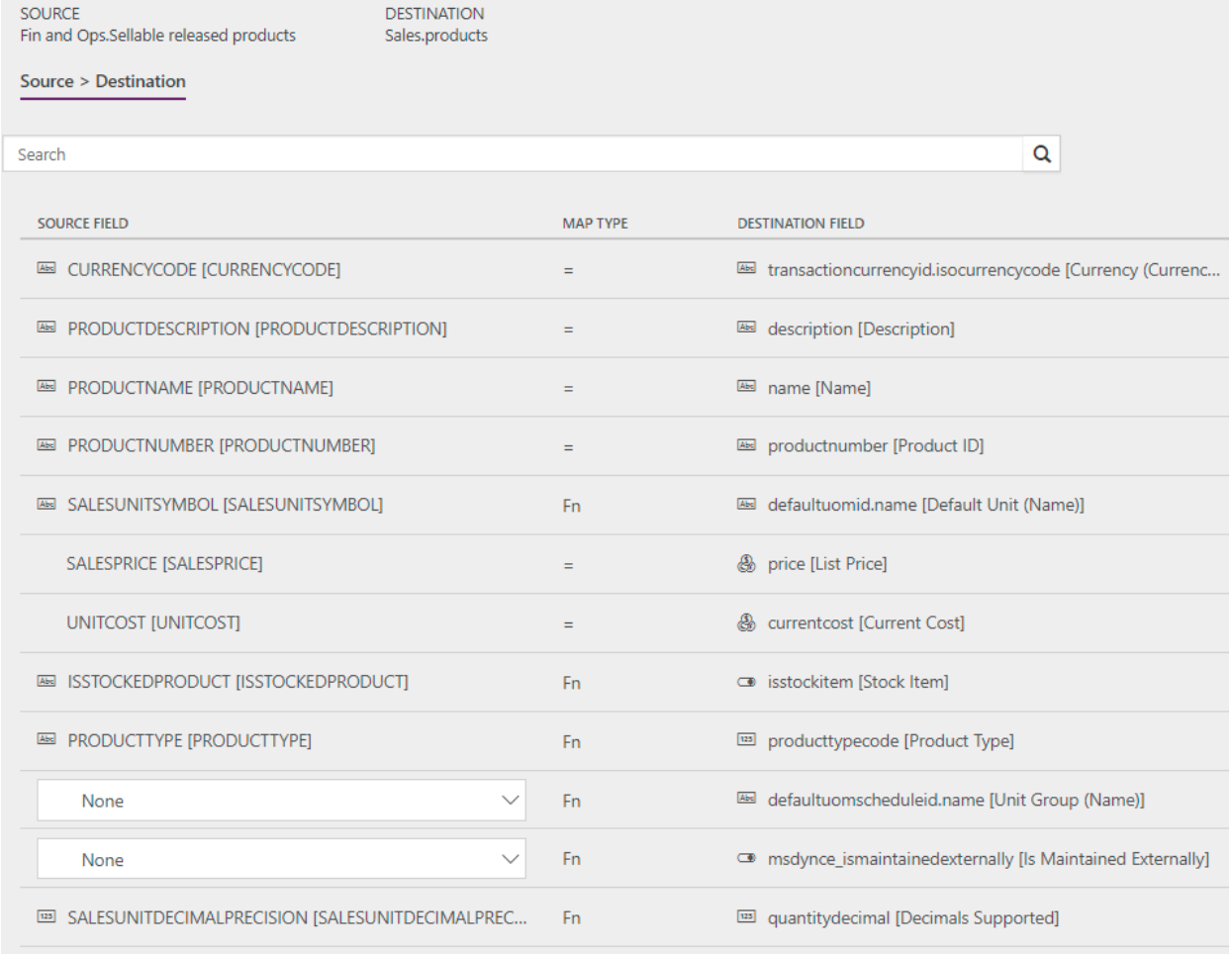
  The default template value is **Default unit**.
- Make sure that the selling UOMs for all products from Supply Chain Management exist in Sales.
- Make sure that price lists exist in Sales for every product sales currency in Supply Chain Management.
- When products are created in Sales, they can have a status of **Draft** or **Active**. The behavior is controlled at **Settings** > **Administration** > **System settings** > **Sales** in Sales.

  Products that have **Draft** status when they are created must be activated before they can be added to quotations or sales orders.

Template mapping in Data integration

The following illustration shows an example of a template mapping in Data integration.

> **NOTE**
> The mapping shows which field information will be synchronized from Sales to Supply Chain Management.



Related topics

Prospect to cash

Synchronize accounts directly from Sales to customers in Supply Chain Management

Synchronize contacts directly from Sales to contacts or customers in Supply Chain Management

Synchronization of sales orders directly between Sales and Supply Chain Management

Synchronize sales invoice headers and lines directly from Supply Chain Management to Sales

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Synchronize contacts directly from Sales to contacts or customers in Supply Chain Management

2/18/2021 • 4 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

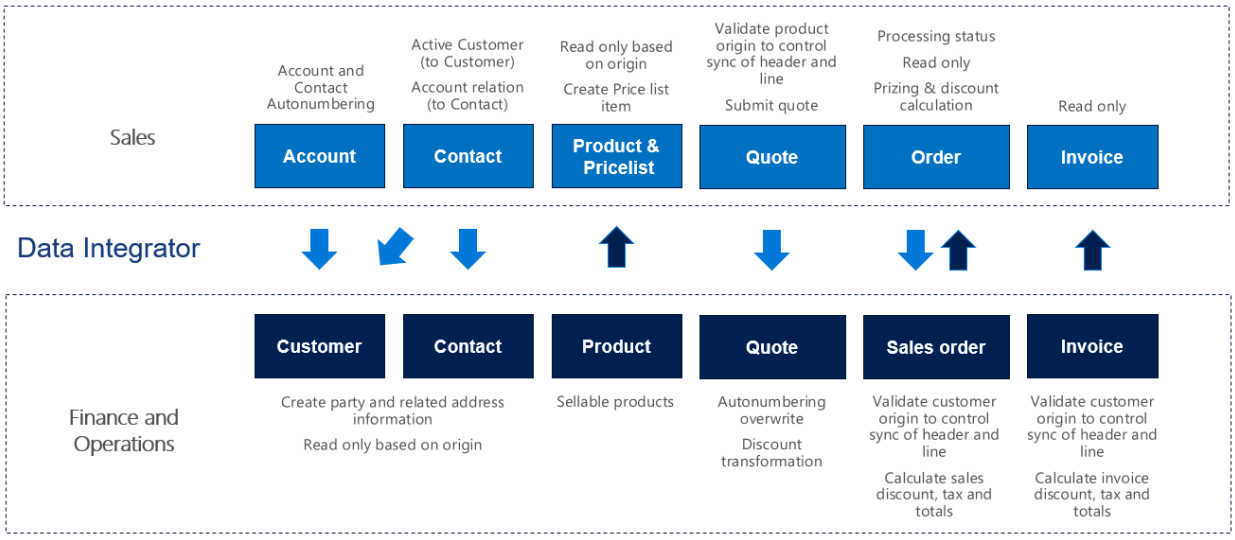
NOTE

Before you can use the Prospect to cash solution, you should be familiar with Integrate data into Microsoft Dataverse for Apps.

This topic discusses the templates and underlying tasks that are used to synchronize Contact (Contacts) and Contact (Customers) tables directly from Dynamics 365 Sales to Dynamics 365 Supply Chain Management.

Data flow in Prospect to cash

The Prospect to cash solution uses the Data integration feature to synchronize data across instances of Supply Chain Management and Sales. The Prospect to cash templates that are available with the Data integration feature enable the flow of data about accounts, contacts, products, sales quotations, sales orders, and sales invoices between Supply Chain Management and Sales. The following illustration shows how the data is synchronized between Supply Chain Management and Sales.



Templates and tasks

To access the available templates, open PowerApps Admin Center. Select **Projects**, and then, in the upper-right corner, select **New project** to select public templates.

The following templates and underlying tasks are used to synchronize Contact (Contacts) tables in Sales to Contact (Customers) tables in Supply Chain Management.

- **Names of the templates in Data integration**
  - Contacts (Sales to Supply Chain Management) - Direct
  - Contacts to Customer (Sales to Supply Chain Management) - Direct
- **Names of the tasks in the Data integration project**
  - Contacts
  - ContactToCustomer

The following synchronization task is required before contact synchronization can occur: Accounts (Sales to Supply Chain Management)

Entity sets

SALES	SUPPLY CHAIN MANAGEMENT
Contacts	Dataverse Contacts
Contacts	Customers V2

Entity flow

Contacts are managed in Sales and synchronized to Supply Chain Management.

A contact in Sales can become either a contact or a customer in Supply Chain Management. To determine whether a contact in Sales should be synchronized to Supply Chain Management as a contact or a customer, the system looks at the following properties on the contact in Sales:

- **Synchronization to a customer in Supply Chain Management**: Contacts where **Is Active Customer** is set to **Yes**
- **Synchronization to a contact in Supply Chain Management**: Contacts where **Is Active Customer** is set to **No** and **Company** (parent account/contact) points to an account (not a contact)

Prospect to cash solution for Sales

A new **Is Active Customer** column has been added to the contact. This column is used to differentiate contacts that have sales activity and contacts that don't have sales activity. **Is Active Customer** is set to **Yes** only for contacts that have related quotations, orders, or invoices. Only those contacts are synchronized to Supply Chain Management as customers.

A new **IsCompanyAnAccount** column has been added to the contact. This column indicates whether a contact is linked to a company (parent account/contact) of the **Account** type. This information is used to identify contacts that should be synchronized to Supply Chain Management as contacts.

A new **Contact Number** column has been added to the contact to help guarantee a natural and unique key for the integration. When a new contact is created, a **Contact Number** value is automatically generated by using a number sequence. The value consists of **CON**, followed by an increasing number sequence and then a suffix of six characters. Here is an example: **CON-01000-BVRCPS**

When the integration solution for Sales is applied, an upgrade script sets the **Contact Number** column for existing contacts by using the number sequence that was mentioned earlier. The upgrade script also sets the **Is Active Customer** column to **Yes** for any contacts that have sales activity.

In Supply Chain Management

Contacts are tagged by using the **IsContactPersonExternallyMaintained** property. This property indicates that a given contact is maintained externally. In this case, externally maintained contacts are maintained in Sales.

Preconditions and mapping setup

Contact to customer

- **CustomerGroup** is required in Supply Chain Management. To help prevent synchronization errors, you can specify a default value in the mapping. That default value is then used if the column is left blank in Sales.

  The default template value is **10**.

- By adding the following mappings, you can help reduce the number of manual updates that are required in Supply Chain Management. You can use a default value or a value map from, for example, **Country/Region** or **City**.

  - **SiteId** – A default site can also be defined on products in Supply Chain Management. A site is required in order to generate quotations and sales orders in Supply Chain Management.

    A template value for **SiteId** isn't defined.

  - **WarehouseId** – A default warehouse can also be defined on products in Supply Chain Management. A warehouse is required in order to generate quotations and sales orders in Supply Chain Management.

    A template value for **WarehouseId** isn't defined.

  - **LanguageId** – A language is required in order to generate quotations and sales orders in Supply Chain Management.

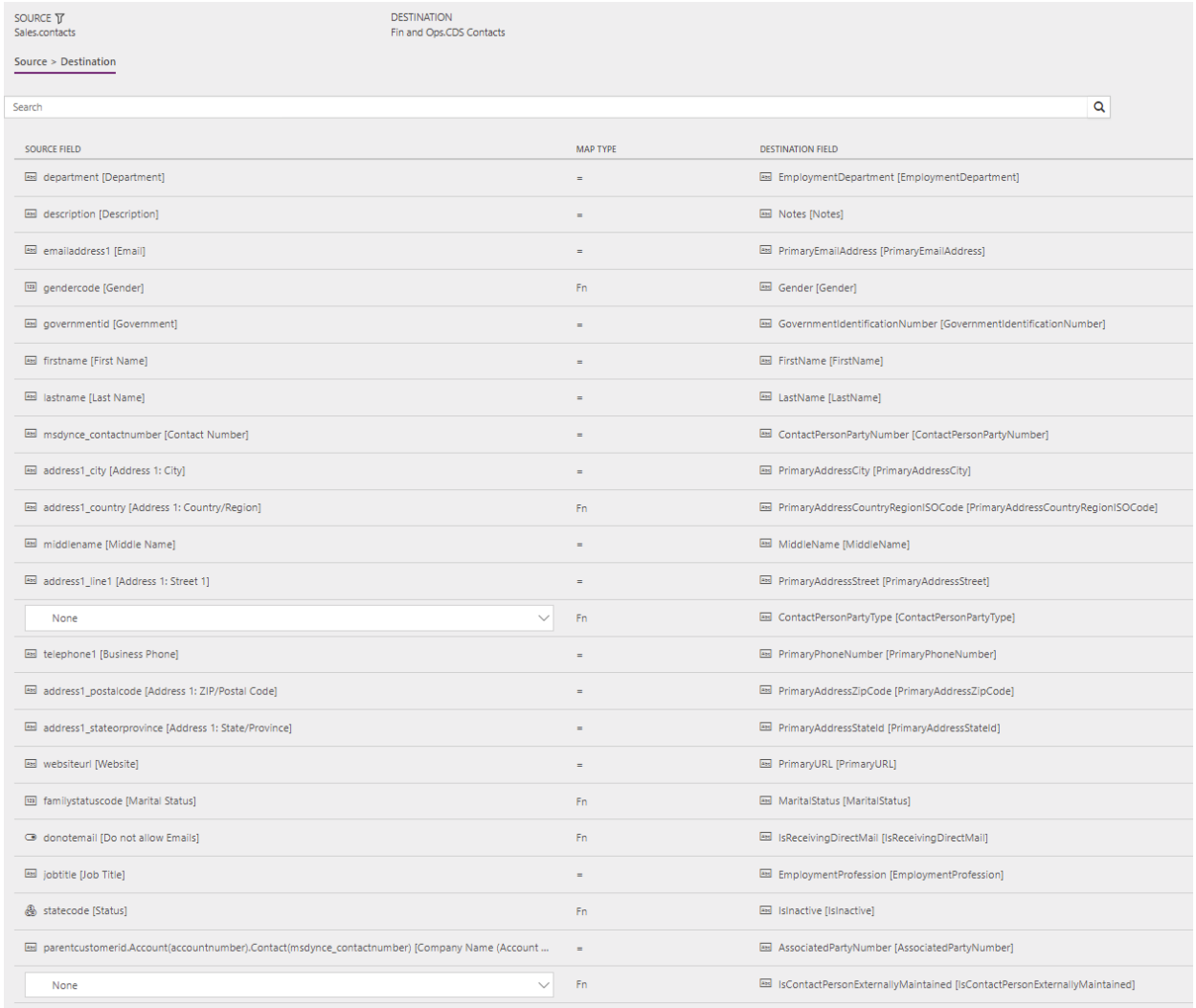
    The default template value for is **en-us**.

Template mapping in Data integration

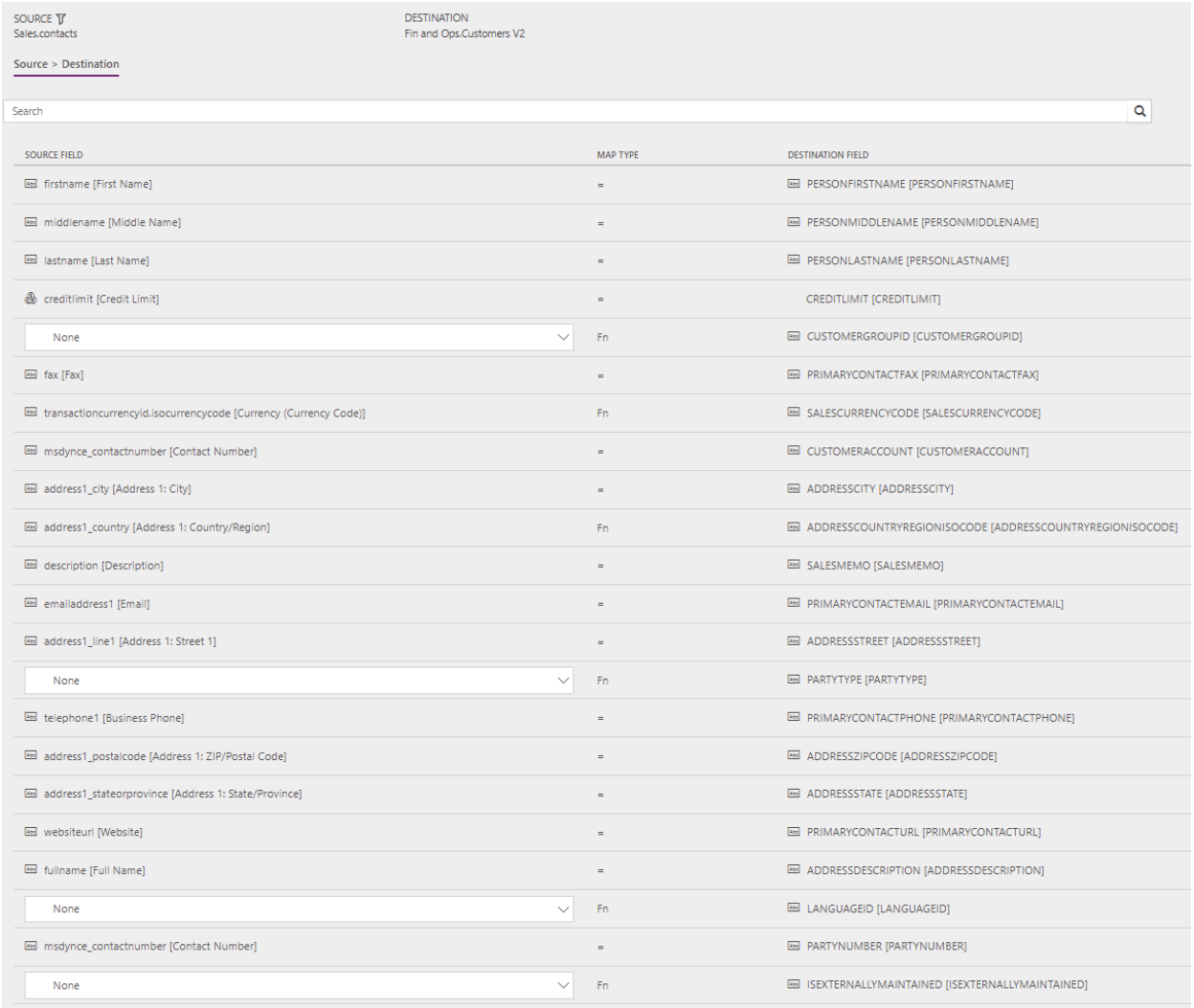
The following illustrations show an example of a template mapping in Data integration.

> **NOTE**
> The mapping shows which column information will be synchronized from Sales to Supply Chain Management.

Contact to contact



Contact to customer



Related topics

Prospect to cash

Synchronize accounts directly from Sales to customers in Supply Chain Management

Synchronize products directly from Supply Chain Management to products in Sales

Synchronization of sales orders directly between Sales and Supply Chain Management

Synchronize sales invoice headers and lines directly from Supply Chain Management to Sales

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Synchronize sales quotation headers and lines directly from Sales to Supply Chain Management

2/18/2021 • 5 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

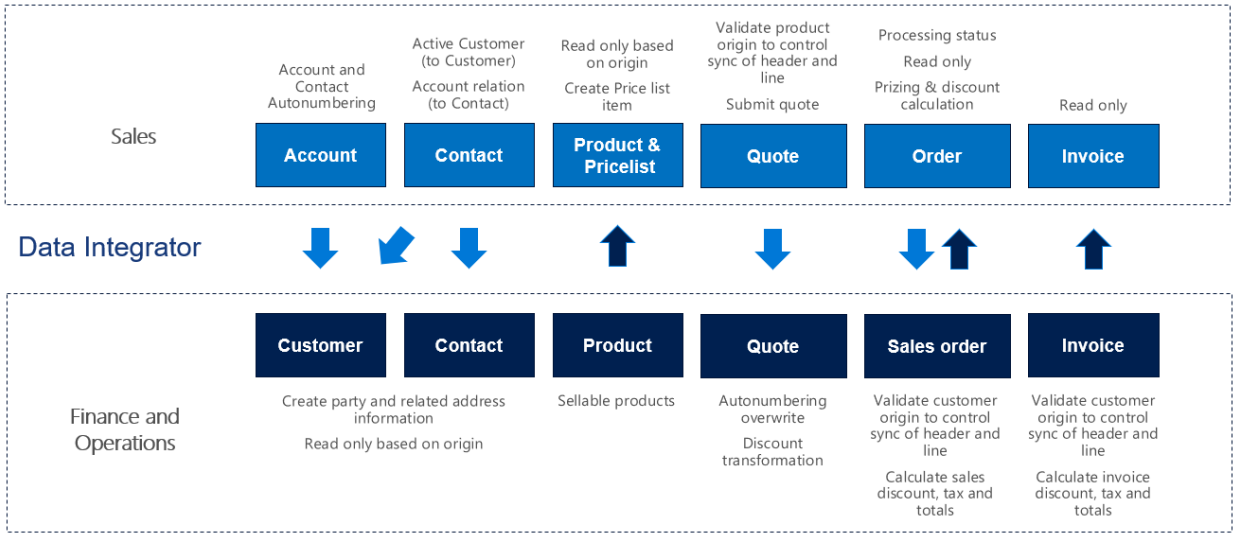
The topic discusses the templates and underlying tasks that are used to synchronize sales quotation headers and lines directly from Dynamics 365 Sales to Dynamics 365 Supply Chain Management.

NOTE

Before you can use the Prospect to cash solution, you should be familiar with Integrate data into Microsoft Dataverse for Apps.

Data flow in Prospect to cash

The Prospect to cash solution uses the Data integration feature to synchronize data across instances of Supply Chain Management and Sales. The Prospect to cash templates that are available with the Data integration feature enable the flow of data for accounts, contacts, products, sales quotations, sales orders, and sales invoices between Supply Chain Management and Sales. The following illustration shows how the data is synchronized between Supply Chain Management and Sales.



Template and tasks

The following template and underlying tasks are used to synchronize sales quotation headers and lines directly from Sales to Supply Chain Management:

- **Name of the template in Data integration:** Sales Quotes (Sales to Supply Chain Management) - Direct
- **Names of the tasks in the Data integration project:**
  - QuoteHeader
  - QuoteLine

The following synchronization tasks are required before synchronization of sales quotation headers and lines can occur:

- Products (Supply Chain Management to Sales) - Direct
- Accounts (Sales to Supply Chain Management) - Direct (if used)
- Contacts to Customers (Sales to Supply Chain Management) - Direct (if used)

Entity set

SALES	SUPPLY CHAIN MANAGEMENT
Quotes	Dataverse sales quotation header
QuoteDetails	Dataverse sales quotation lines

Entity flow

Sales quotations are created in Sales and synchronized to Supply Chain Management.

Sales quotations from Sales are synchronized only if the following conditions are met:

- All quote products on the sales quotation are externally maintained.
- After you click **Activate quote**, the sales quotation is active.

Prospect to cash solution for Sales

The **Has Externally Maintained Products Only** field has been added to the **Quote** entity to consistently track whether the sales quotation consists entirely of externally maintained products. If a sales quotation has only externally maintained products, the products are maintained in Supply Chain Management. This behavior helps guarantee that you don't try to synchronize sales quotation lines that have products that are unknown to Supply Chain Management.

All quote products on the sales quotation are updated with the **Has Externally Maintained Products Only** information from the sales quotation header. This information is found in the **Quote Has Externally Maintained Products Only** field on the **QuoteDetails** entity.

A discount can be added to the quote product and will be synchronized to Supply Chain Management. The **Discount**, **Charges**, and **Tax** fields on the header are controlled by a setup in Supply Chain Management. Currently, this setup doesn't support integration mapping. In the current design, the **Price**, **Discount**, **Charge**, and **Tax** fields are maintained and handled in FSupply Chain Management.

In Sales, the solution makes the following fields read-only, because the values aren't synchronized to Supply Chain Management:

- Read-only fields on the sales quotation header: **Discount %**, **Discount**, and **Freight Amount**
- Read-only fields on quote products: **Tax**

Preconditions and mapping setup

Before sales quotations are synchronized, it's important that you update the following settings.

Setup in Sales

- Make sure that permissions are set up for the team that the user from your connection set in Sales is assigned to. If you're using demo data, the user usually has admin access, but the team doesn't have admin access. If the team doesn't have admin access when you run the project from Data integration, you will receive an error message that states that the Principal team is missing.

  To set up permissions for the team, go to **Settings** > **Security** > **Teams**, and select the relevant team. Select **Manage Roles**, and then select a role that has the desired permissions, such as **System Administrator**.

- Go to **Settings** > **Administration** > **System settings** > **Sales**, and make sure that the following settings are used:

  - The **Use system prizing calculation system** option is set to **Yes**.
  - The **Discount calculation method** field is set to **Line item**.

Setup in the Data integration project

QuoteHeader

- Make sure that the required mapping exists for **Shipto_country** to **DeliveryAddressCountryRegionISOCode**. In the value map, you can define a default value that is used if the value is left blank. Just leave the left side blank, and set the right side to the desired country or region. In this way, you don't have to type the country or region for national orders.

  The template value is a value map where several countries or regions are mapped, and where a blank value equals a value of **US**.

QuoteLine

- Make sure that the required value map exists for **SalesUnitSymbol** in Supply Chain Management.
- Make sure that the required units are defined in Sales.

  A template value that has a value map is defined for **oumid.name** to **SalesUnitSymbol**.

- Optional: You can add the following mappings to help guarantee that sales quotation lines are imported into Supply Chain Management if there is no default information from either the customer or the product:

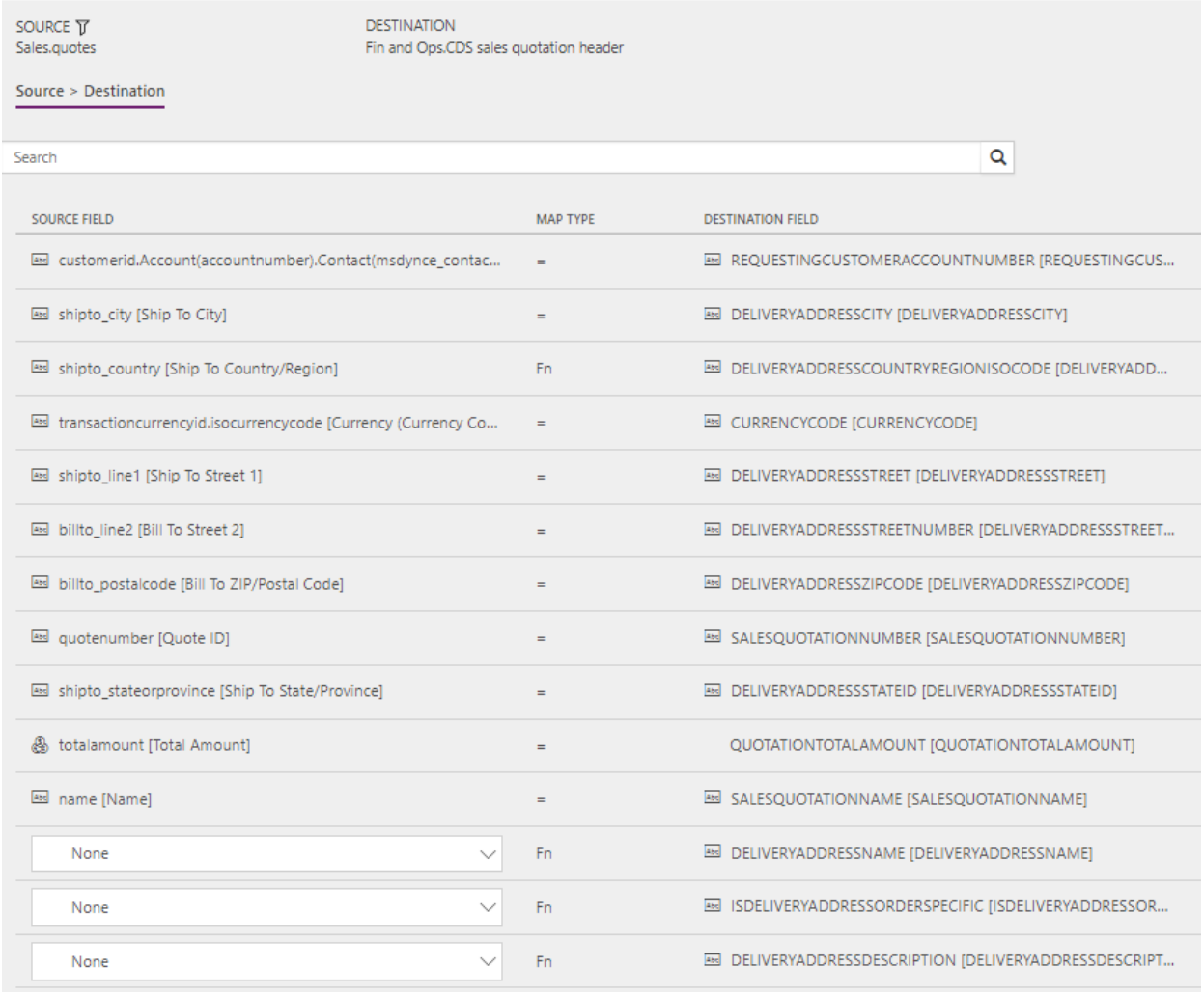
  - **SiteId** – A site is required in order to generate quotations and sales order lines in Supply Chain Management. There is no default template value for **SiteId**.
  - **WarehouseId** – A warehouse is required in order to process quotations and sales order lines in Supply Chain Management. There is no default template value for **WarehouseId**.

Template mapping in data integrator

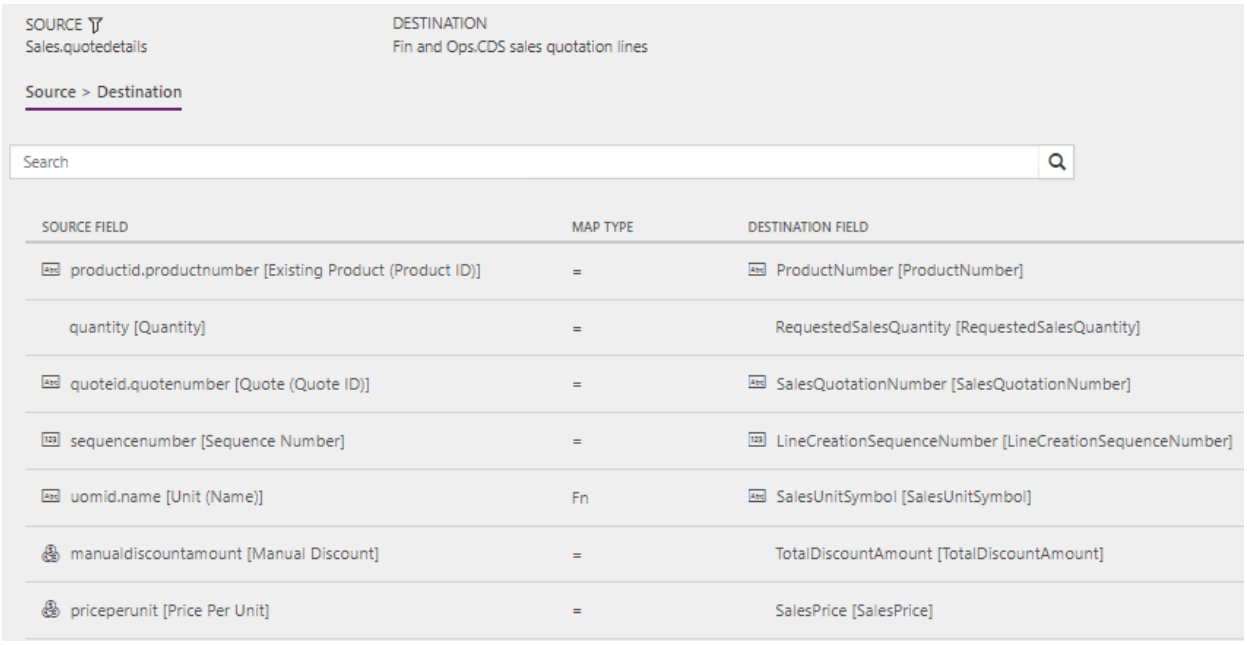
> **NOTE**
>
> - The **Discount**, **Charges**, and **Tax** fields are controlled by a complex setup in Supply Chain Management. Currently, this setup doesn't support integration mapping. In the current design, the **Price**, **Discount**, **Charge**, and **Tax** fields are handled by Supply Chain Management.
> - The **Payment terms**, **Freight terms**, **Delivery terms**, **Shipping method**, and **Delivery mode** fields aren't part of the default mappings. To map these fields, you must set up a value mapping that is specific to the data in the organizations that the entity is synchronized between.

The following illustrations show an example of a template mapping in data integrator.

QuoteHeader



QuoteLine



Related topics

Prospect to cash

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Synchronization of sales orders directly between Sales and Supply Chain Management

2/18/2021 • 10 minutes to read • Edit Online

NOTE

Effective November 2020:

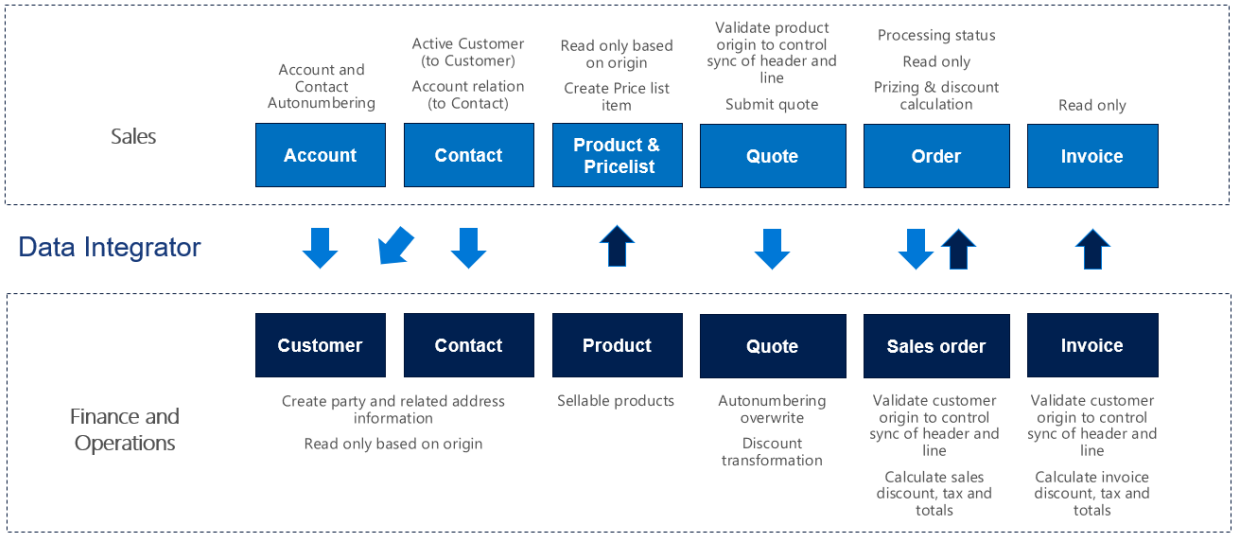
- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

The topic discusses the templates and underlying tasks that are used to run synchronization of sales orders directly between Dynamics 365 Sales and Dynamics 365 Supply Chain Management.

Data flow in Prospect to cash

The Prospect to cash solution uses the Data integration feature to synchronize data across instances of Supply Chain Management and Sales. The Prospect to cash templates that are available with the Data integration feature enable the flow of data for accounts, contacts, products, sales quotations, sales orders, and sales invoices between Supply Chain Management and Sales. The following illustration shows how the data is synchronized between Supply Chain Management and Sales.



Templates and tasks

To access the available templates, open Power Apps Admin Center. Select **Projects**, and then, in the upper-right corner, select **New project** to select public templates.

The following templates and underlying tasks are used to run synchronization of sales orders directly between Sales and Supply Chain Management.

- **Names of the templates in Data integration:**
  - Sales Orders (Sales to Supply Chain Management) - Direct

    - Sales Orders (Supply Chain Management to Sales) - Direct
- **Names of the tasks in the Data integration project:**
    - OrderHeader
    - OrderLine

The following synchronization tasks are required before synchronization of sales invoice headers and lines can occur:

- Products (Supply Chain Management to Sales) - Direct
- Accounts (Sales to Supply Chain Management) - Direct (if used)
- Contacts to Customers (Sales to Supply Chain Management) - Direct (if used)

Entity set

SUPPLY CHAIN MANAGEMENT	SALES
Dataverse sales order headers	SalesOrders
Dataverse sales order lines	SalesOrderDetails

Entity flow

Sales orders are created in Sales and synchronized to Supply Chain Management when **Run project** is triggered for a project based on the **Sales Orders (Sales to Supply Chain Management) - Direct** template. You can only activate and synchronize orders from Sales if all **Order Products** consist of products that are externally maintained. Therefore, there can be no write-in products. After the order is activated, the sales order becomes read-only in the user interface (UI). At that point, the updates are made from Supply Chain Management. After the sales order has a status of **Confirmed**, the a project based on the **Sales Orders (Supply Chain Management to Sales) - Direct** template can be used to synchronize updates or fulfillment status from Supply Chain Management to Sales.

You don't have to create orders in Sales. You can create new sales orders in Supply Chain Management instead. After a sales order has a status of **Confirmed**, it's synchronized to Sales as described in the previous paragraph.

In Supply Chain Management, filters in the template help guarantee that only the relevant sales orders are included in the synchronization:

- On the sales order, both the ordering customer and the invoicing customer have to originate from Sales to be included in the synchronization. In Supply Chain Management, the **OrderingCustomerIsExternallyMaintained** and **InvoiceCustomerIsExternallyMaintained** columns are used to filter sales orders from the data tables.
- The sales order in Supply Chain Management must be confirmed. Only confirmed sales orders or sales orders that have a higher processing status, such as **Shipped** or **Invoiced**, are synchronized to Sales.
- After a sales order is created or modified, the **Calculate sales totals** batch job in Supply Chain Management must be run. Only sales orders where sales totals are calculated will be synchronized to Sales.

Freight tax

Sales doesn't support tax at the header level, because tax is stored at the line level. To support tax at the header level from Supply Chain Management, such as tax on freight, the system synchronizes tax to Sales as a write-in product that is named **Freight Tax**, and that has the tax amount from Supply Chain Management. In this way, the standard price calculation in Sales can be used for totals, even when there is tax at the header level from

Supply Chain Management.

Discount calculation and rounding

The discount calculation model in Sales differs from the discount calculation model in Supply Chain Management. In Supply Chain Management, the final discount amount on a sales line can be the result of a combination of discount amounts and discount percentages. If this final discount amount is divided by the quantity on the line, rounding can occur. However, this rounding isn't considered if a rounded per-unit discount amount is synchronized to Sales. To help guarantee that the full discount amount from a sales line in Supply Chain Management is correctly synchronized to Sales, the full amount must be synchronized without being divided by the line quantity. Therefore, you must define the **Discount calculation method** as **Line item** in Sales.

When a sales order line is synchronized from Sales to Supply Chain Management, the full line discount amount is used. Because Supply Chain Management has no field that can store the full discount amount for a line, the amount is divided by the quantity and stored in the **Line discount** field. Any rounding that occurs in this division is stored in the **Sales charges** field on the sales line.

Example

Synchronization from Sales to Supply Chain Management

- **Sales:** Quantity = 3, per-line discount = $10.00
- **Supply Chain Management:** Quantity = 3, line discount amount = $3.33, sales charge = -$0.01

Synchronization from Supply Chain Management to Sales

- **Supply Chain Management:** Quantity = 3, line discount amount = $3.33, sales charge = -$0.01
- **Sales:** Quantity = 3, per-line discount = (3 × $3.33) + $0.01 = $10.00

Prospect to cash solution for Sales

New columns have been added to the **Order** table and appear on the page:

- **Is Maintained Externally** – Set this option to **Yes** when the order is coming from Supply Chain Management.
- **Processing status** – This column shows the processing status of the order in Supply Chain Management. The following values are available:
  - **Draft** – The initial status when an order is created in Sales. In Sales, only orders with this processing status can be edited.
  - **Active** – The status after the order is activated in Sales by using the **Activate** button.
  - **Confirmed**
  - **Packing Slip**
  - **Invoiced**
  - **Picked**
  - **Partially Picked**
  - **Partially Packed**
  - **Shipped**
  - **Partially Shipped**
  - **Partially Invoiced**
  - **Cancelled**

The **Has Externally Maintained Products Only** setting is used during order activation to consistently track

whether a sales order consists entirely of externally maintained products. If a sales order consists entirely of externally maintained products, the products are maintained in Supply Chain Management. This setting helps guarantee that you don't activate and try to synchronize sales order lines that have products that are unknown to Supply Chain Management.

The **Create Invoice**, **Cancel Order**, **Recalculate**, **Get Products**, and **Lookup Address** buttons on the **Sales order** page are hidden for externally maintained orders, because invoices will be created in Supply Chain Management and synchronized to Sales. These orders can't be edited, because sales order information will be synchronized from Supply Chain Management after activation.

The sales order status will remain **Active** to help guarantee that changes from Supply Chain Management can flow to the sales order in Sales. To control this behavior, set the default **Statecode [Status]** value to **Active** in the Data integration project.

Preconditions and mapping setup

Before you synchronize sales orders, it's important that you update the following settings in the systems.

Setup in Sales

- Make sure that permissions are set up for the team that the user from your Sales connection set is assigned to. If you're using demo data, the user usually has admin access, but the team doesn't have admin access. If the team doesn't have admin access, when you run the project from Data integration, you will receive an error message that states that the Principal team is missing.

  Go to **Settings** > **Security** > **Teams**, select the relevant team, select **Manage Roles**, and select a role that has the desired permissions, such as **System Administrator**.

- To ensure correct calculation of discounts in both Sales and Supply Chain Management **Discount calculation method** must be set to **Line item**.

- Go to **Settings** > **Administration** > **System settings** > **Sales**, and make sure that the following settings are used:

  - The **Use system prizing calculation system** option is set to **Yes**.
  - The **Discount calculation method** column is set to **Line item**.

Setup in Supply Chain Management

- Go to **Sales and marketing** > **Periodic tasks** > **Calculate sales totals**, and set the job to run as a batch job. Set the **Calculate totals for sales orders** option to **Yes**. This step is important, because only sales orders where sales totals are calculated will be synchronized to Sales. The frequency of the batch job should be aligned with the frequency of sales order synchronization.

If you also use work order integration, you need to set up the sales origin. The sales origin is used to distinguish sales orders in Supply Chain Management that were created from work orders in Field Service. When a sales order has a sales origin of the **Work order integration** type, the **External work order status** field appears on the sales order header. Additionally, the sales origin ensures that sales orders that were created from work orders in Field Service are filtered out during sales order synchronization from Supply Chain Management to Field Service.

1. Go to **Sales and marketing** > **Setup** > **Sales orders** > **Sales origin**.
2. Select **New** to create a new sales origin.
3. In the **Sales origin** column, enter a name for the sales origin, such as **SalesOrder**.
4. In the **Description** column, enter a description, such as **Sales Order from Sales**.
5. Select the **Origin type assignment** check box.
6. Set the **Sales origin type** column to **Sales order integration**.

7. Select **Save**.

Setup in the Sales Orders (Sales to Supply Chain Management) - Direct Data integration project

- Make sure that the required mapping exists for **Shipto_country** to **DeliveryAddressCountryRegionISOCode**. You can make blank a default value in the value map to avoid having to type country for national orders. Set the left side to 'Blank', and set the right side to the desired country or region.

  The template value is a value map where several countries or regions are mapped, and where 'Blank' = US.

Setup in the Sales Orders (Supply Chain Management to Sales) - Direct Data integration project

SalesHeader task

- A price list is required in order to create orders in Sales. Update the value map for **pricelevelid.name [Price List Name]** to the price list that is used in Sales per currency. You can use the default price list for a single currency. Alternatively, if you have price lists in multiple currencies, you can use a value map.

  The default template value for **pricelevelid.name [Price List Name]** is **CRM Service USA (sample)**.

SalesLine task

- Make sure that the required value map for **SalesUnitSymbol** in Supply Chain Management exists.
- Make sure that the required units are defined in Sales.

  A template value that has a value map is defined for **SalesUnitSymbol** to **oumid.name**.

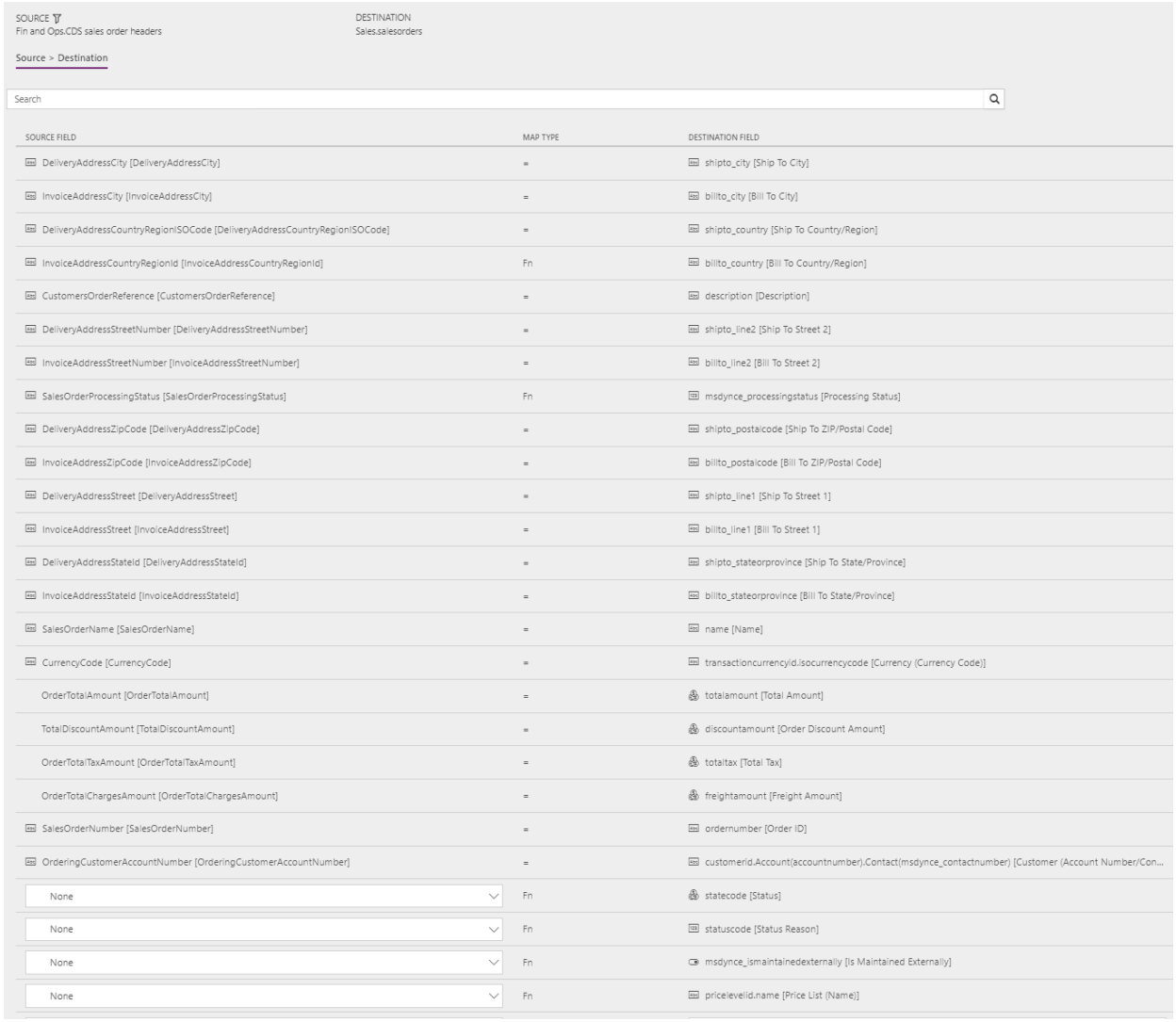
Template mapping in Data integration

> **NOTE**
> The **Payment terms**, **Freight terms**, **Delivery terms**, **Shipping method**, and **Delivery mode** columns aren't part of the default mappings. To map these columns, you must set up a value mapping that is specific to the data in the organizations that the table is synchronized between.

The following illustrations show an example of a template mapping in Data integration.

> **NOTE**
> The mapping shows which column information will be synchronized from Sales to Supply Chain Management, or from Supply Chain Management to Sales.

Sales Orders (Supply Chain Management to Sales) - Direct: OrderHeader



Sales Orders (Supply Chain Management to Sales) - Direct: OrderLine

SOURCE
Fin and Ops.CDS sales order lines

DESTINATION
Sales.salesorderdetails

Source > Destination

Search

SOURCE FIELD	MAP TYPE	DESTINATION FIELD
CurrencyCode [CurrencyCode]	=	transactioncurrencyid.isocurrencycode [Currency (Currency Code)]
DeliveryAddressCity [DeliveryAddressCity]	=	shipto_city [Ship To City]
DeliveryAddressCountryRegionISOCode [DeliveryAddressCountryRegionISOCode]	=	shipto_country [Ship To Country/Region]
DeliveryAddressStateId [DeliveryAddressStateId]	=	shipto_stateorprovince [Ship To State/Province]
DeliveryAddressStreet [DeliveryAddressStreet]	=	shipto_line1 [Ship To Street 1]
DeliveryAddressStreetNumber [DeliveryAddressStreetNumber]	=	shipto_line2 [Ship To Street 2]
DeliveryAddressZipCode [DeliveryAddressZipCode]	=	shipto_postalcode [Ship To ZIP/Postal Code]
LineAmount [LineAmount]	Fn	extendedamount [Extended Amount]
LineCreationSequenceNumber [LineCreationSequenceNumber]	=	sequencenumber [Sequence Number]
OrderedSalesQuantity [OrderedSalesQuantity]	=	quantity [Quantity]
ProductName [ProductName]	=	description [Description]
ProductNumber [ProductNumber]	=	productid.productnumber [Existing Product (Product ID)]
SalesOrderNumber [SalesOrderNumber]	=	salesorderid.ordernumber [Order (Order ID)]
SalesOrderNumber [SalesOrderNumber]	=	msdynce_ordernumber [Order Number]
SalesUnitSymbol [SalesUnitSymbol]	Fn	uomid.name [Unit (Name)]
TotalDiscountAmount [TotalDiscountAmount]	=	manualdiscountamount [Manual Discount]
TotalTaxAmount [TotalTaxAmount]	=	tax [Tax]
SalesPrice [SalesPrice]	=	priceperunit [Price Per Unit]
None	Fn	ispriceoverridden [Pricing]
None	Fn	msdynce_ismaintainedexternally [Is Maintained Externally]

Sales Orders (Sales to Supply Chain Management) - Direct: OrderHeader

SOURCE
Sales.salesorders

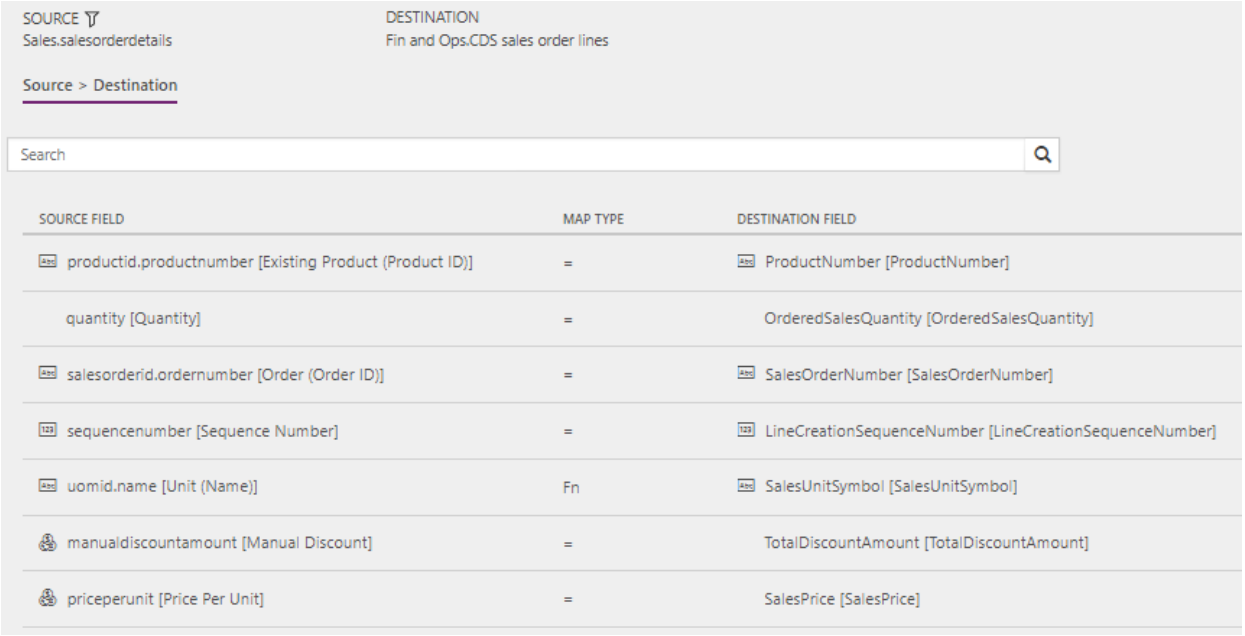
DESTINATION
Fin and Ops.CDS sales order headers

Source > Destination

Search

SOURCE FIELD	MAP TYPE	DESTINATION FIELD
customerid.Account(accountnumber).Contact(msdynce_contac...	=	OrderingCustomerAccountNumber [OrderingCustomerAccoun...
shipto_city [Ship To City]	=	DeliveryAddressCity [DeliveryAddressCity]
shipto_country [Ship To Country/Region]	Fn	DeliveryAddressCountryRegionISOCode [DeliveryAddressCoun...
transactioncurrencyid.isocurrencycode [Currency (Currency Co...	=	CurrencyCode [CurrencyCode]
shipto_line1 [Ship To Street 1]	=	DeliveryAddressStreet [DeliveryAddressStreet]
shipto_postalcode [Ship To ZIP/Postal Code]	=	DeliveryAddressZipCode [DeliveryAddressZipCode]
ordernumber [Order ID]	=	SalesOrderNumber [SalesOrderNumber]
shipto_line2 [Ship To Street 2]	=	DeliveryAddressStreetNumber [DeliveryAddressStreetNumber]
shipto_stateorprovince [Ship To State/Province]	=	DeliveryAddressStateId [DeliveryAddressStateId]
totalamount [Total Amount]	=	OrderTotalAmount [OrderTotalAmount]
name [Name]	=	CustomersOrderReference [CustomersOrderReference]
None	Fn	DeliveryAddressName [DeliveryAddressName]
None	Fn	IsDeliveryAddressOrderSpecific [IsDeliveryAddressOrderSpecific]
None	Fn	DeliveryAddressDescription [DeliveryAddressDescription]

Sales Orders (Sales to Supply Chain Management) - Direct: OrderLine



Related topics

Prospect to cash

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

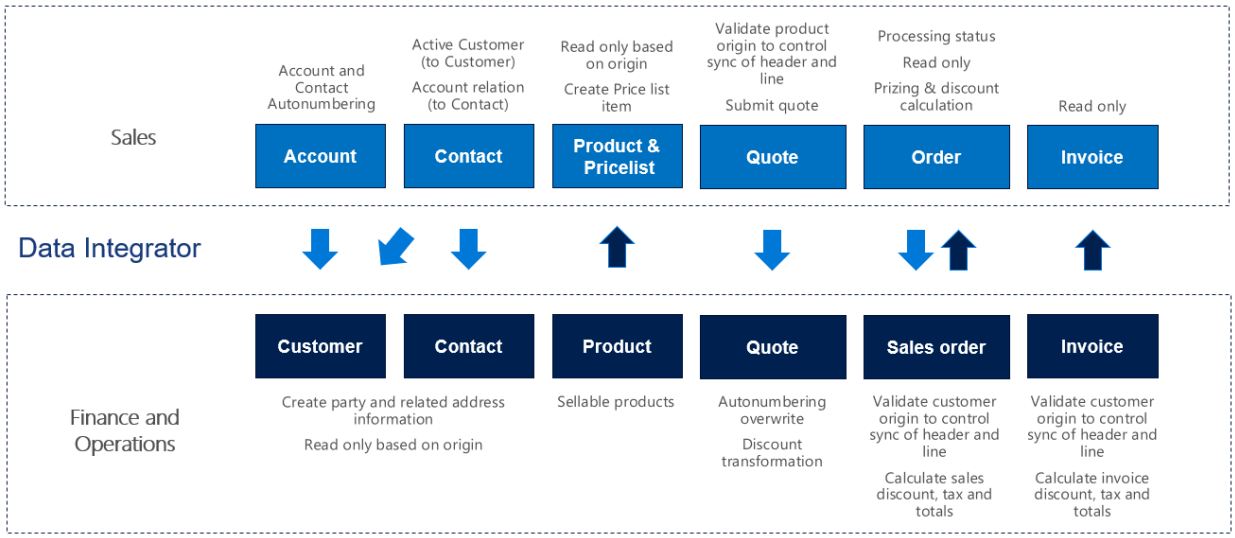
Synchronize sales invoice headers and lines directly from Finance and Operations to Sales

2/18/2021 • 4 minutes to read • Edit Online

This topic discusses the templates and underlying tasks that are used to synchronize sales invoice headers and lines directly from Dynamics 365 Supply Chain Management to Dynamics 365 Sales.

Data flow in Prospect to cash

The Prospect to cash solution uses the Data integration feature to synchronize data across instances of Supply Chain Management and Sales. The Prospect to cash templates that are available with the Data integration feature enable the flow of data about accounts, contacts, products, sales quotations, sales orders, and sales invoices between Supply Chain Management and Sales. The following illustration shows how the data is synchronized between Supply Chain Management and Sales.



Templates and tasks

To access the available templates, open Power Apps Admin Center. Select **Projects**, and then, in the upper-right corner, select **New project** to select public templates.

The following template and underlying tasks are used to synchronize sales invoice headers and lines from Supply Chain Management to Sales:

- **Name of the template in Data integration:** Sales Invoices (Fin and Ops to Sales) - Direct
- **Names of the tasks in the Data integration project:**
  - SalesInvoiceHeader
  - SalesInvoiceLine

The following synchronization tasks are required before synchronization of sales invoice headers and lines can occur:

- Products (Supply Chain Management to Sales) - Direct
- Accounts (Sales to Supply Chain Management) - Direct (if used)
- Contacts (Sales to Supply Chain Management) - Direct (if used)

- Sales order header and lines (Supply Chain Management to Sales) - Direct

Entity set

SUPPLY CHAIN MANAGEMENT	SALES
Externally maintained customer sales invoice headers	Invoices
Externally maintained customer sales invoice lines	InvoiceDetails

Entity flow

Sales invoices are created in Supply Chain Management and synchronized to Sales.

> **NOTE**
> Currently, tax that is related to charges on the sales invoice header isn't included in the synchronization from Supply Chain Managements to Sales. Sales doesn't support tax information at the header level. However, tax that is related to charges at the line level is included in the synchronization.

Prospect to cash solution for Sales

- An **Invoice number** field has been added to the **Invoice** entity and appears on the page.
- The **Create invoice** button on the **Sales order** page is hidden, because invoices will be created in Supply Chain Management and synchronized to Sales. The **Invoice** page can't be edited, because invoices will be synchronized from Supply Chain Management.
- The **Sales order status** value is automatically changed to **Invoiced** when the related invoice from Supply Chain Management has been synchronized to Sales. Additionally, the owner of the sales order that the invoice was created from is assigned as the owner of the invoice. Therefore, the owner of the sales order can view the invoice.

Preconditions and mapping setup

Before you synchronize sales invoices, it's important that you update the following settings in the systems.

Setup in Sales

Go to **Settings** > **Administration** > **System settings** > **Sales**, and make sure that the following settings are used:

- The **Use system prizing calculation system** option is set to **Yes**.
- The **Discount calculation method** field is set to **Line item**.

Setup in the Data integration project

SalesInvoiceHeader task

- Make sure that the required mapping exists for **InvoiceCountryRegionId** to **BillingAddress_Country**.

  The template value is a value map where several countries or regions are mapped.

- A price list is required in order to create invoices in Sales. Update the value map for **pricelevelid.name [Price list name]** to the price list that is used in Sales per currency. You can use the default price list for a single currency. Alternatively, if you have price lists in multiple currencies, you can use a value map.

  The template value for **pricelevelid.name [Price list name]** is a value map that is based on currency with USD = CRM Service USA (sample).

SalesInvoiceLine task

- Make sure that the required mapping exists for **Unit of measure**.
- Make sure that the required value map for **SalesUnitSymbol** in Supply Chain Management exists.

  A template value that has a value map is defined for **SalesUnitSymbol** to **Quantity_UOM**.

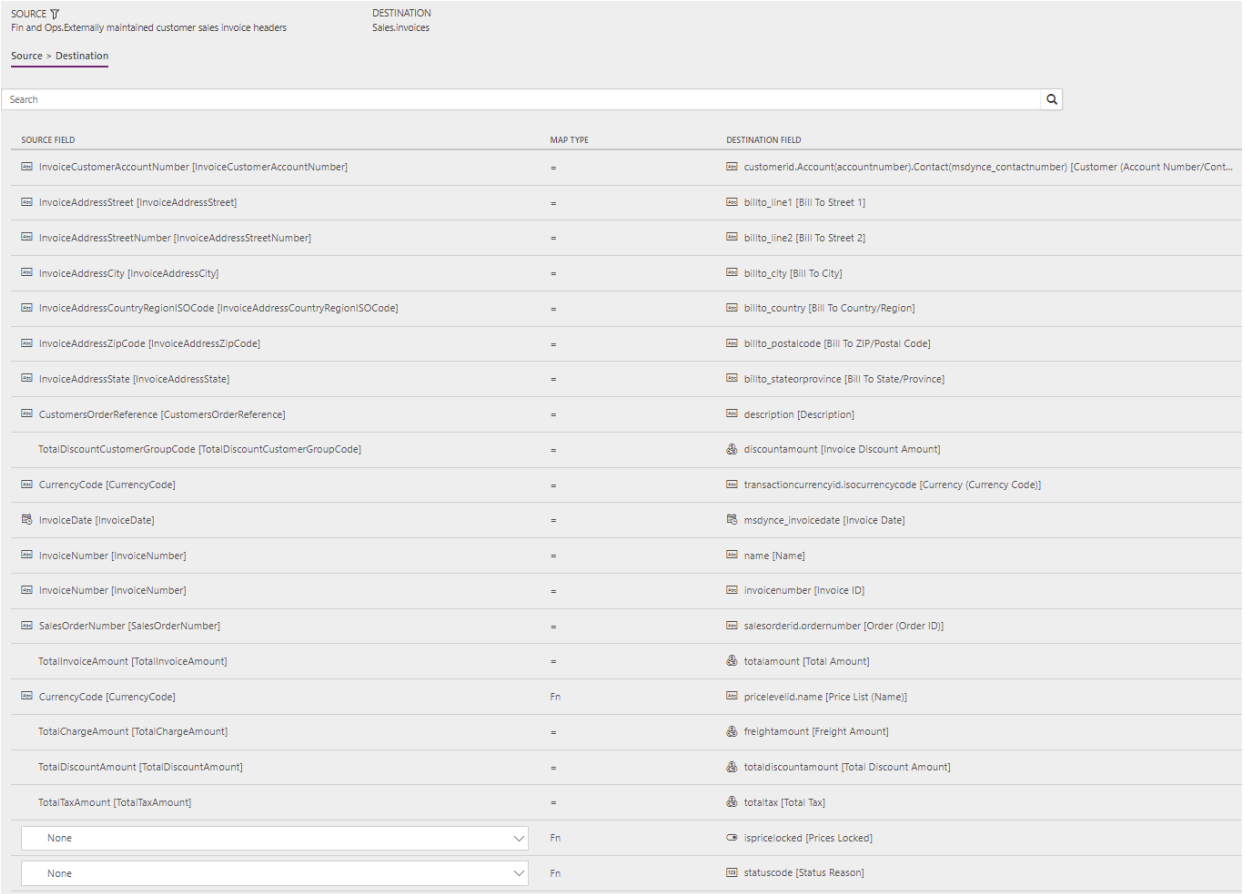
Template mapping in Data integration

> **NOTE**
>
> The **Payment terms**, **Freight terms**, **Delivery terms**, **Shipping method**, and **Delivery mode** fields aren't included in the default mappings. To map these fields, you must set up a value mapping that is specific to the data in the organizations that the entity is synchronized between.

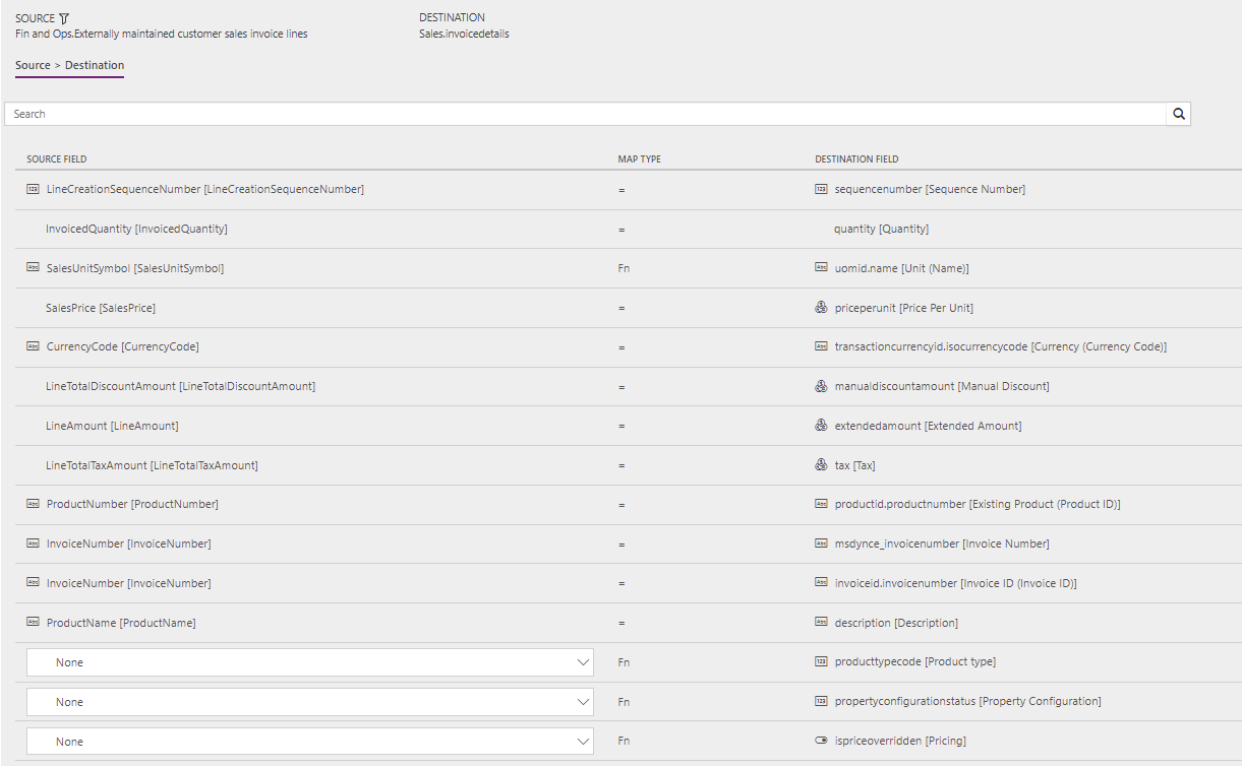
The following illustrations show an example of a template mapping in Data integration.

> **NOTE**
>
> The mapping shows which field information will be synchronized from Sales to Supply Chain Management.

SalesInvoiceHeader



SalesInvoiceLine



Related topics

Prospect to cash

Synchronize accounts directly from Sales to customers in Supply Chain Management

Synchronize products directly from Supply Chain Management to products in Sales

Synchronize contacts directly from Sales to contacts or customers in Supply Chain Management

Synchronization of sales orders directly between Sales and Supply Chain Management

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Integration with Microsoft Dynamics 365 Field Service overview

2/18/2021 • 3 minutes to read • Edit Online

NOTE

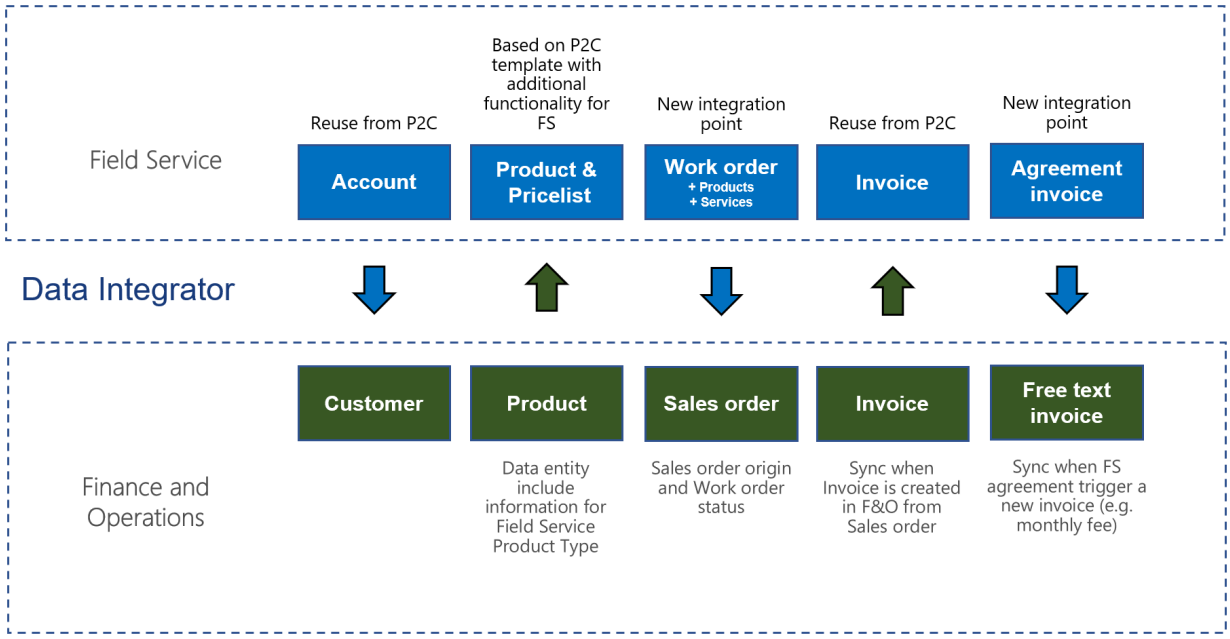
Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

Supply Chain Management enables synchronization of business processes between Dynamics 365 Supply Chain Management and Dynamics 365 Field Service. The integration scenarios are configured by using extensible Data integrator templates and Microsoft Dataverse to enable the synchronization of business processes. Standard templates can be used to create custom integration projects, where additional standard and custom columns and tables can be mapped to adjust the integration and meet specific business needs.

The field service integration builds on top of the existing prospect-to-cash functionality.



The first phase of the integration between Field Service and Supply Chain Management is focused on enabling work orders and agreements in Field Service to be invoiced in Supply Chain Management. The supported flow starts in Field Service, where information from work orders is synchronized to Supply Chain Management as sales orders. In Supply Chain Management, the sales orders are invoiced to generate invoice documents. In addition, the information from Field Service agreement invoices is synchronized to Supply Chain Management. The Microsoft Dynamics 365 Data integrator synchronizes data by using customizable projects. Standard templates can be used to create custom integration projects where additional standard and custom columns, and also tables, can be mapped to adjust the integration and meet specific requirements.

The first phase of the integration between Field Service and Supply Chain Management enables synchronization of the following items:

- Synchronize products in Supply Chain Management to products in Field Service
- Synchronize work orders in Field Service to sales orders in Supply Chain Management
- Synchronize agreement invoices in Field Service to free text invoices in Supply Chain Management

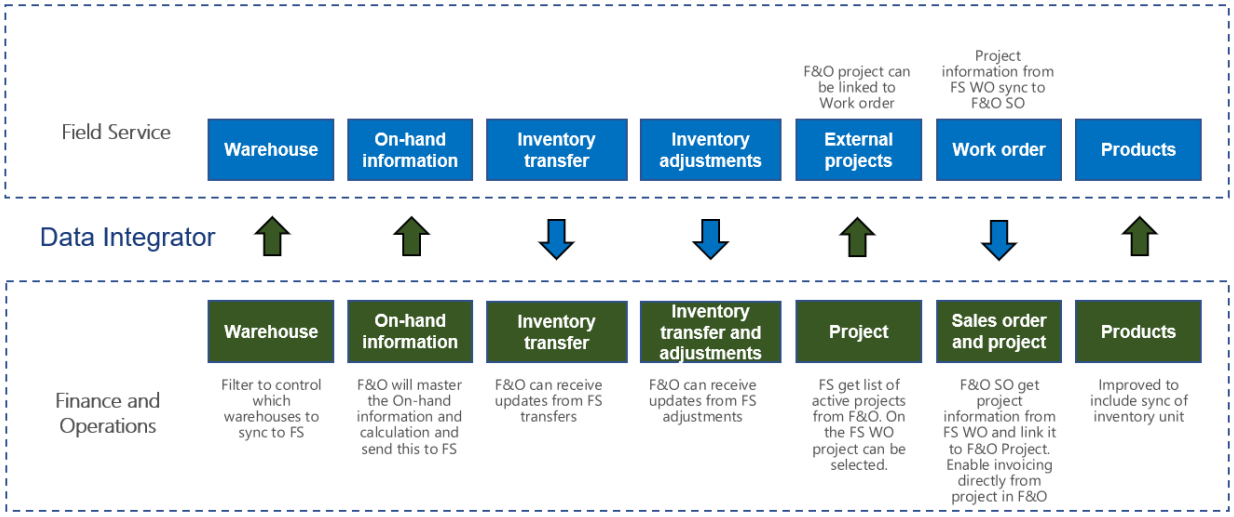
To see an example of how you can synchronize a work order between Field Service and Supply Chain Management, watch the short YouTube video How to synchronize a work order with Microsoft Dynamics 365 Integration.

Integration with Field Service, including inventory and project information

The additional functionality in this second phase focused on giving field technicians insight about the inventory information from Supply Chain Management, allowing them to update inventory levels and do material transfers. In addition, companies installing or servicing sold goods will benefit from better control and visibility to the full sales and service process with integration from projects.

Functionality includes integration of:

- Warehouse information
- On-hand inventory information
- Inventory transfers
- Inventory adjustments
- Supply Chain Management projects connected with Dynamics 365 Field Service work orders
- Dynamics 365 Field Service work orders with link to Supply Chain Management projects, apply this project number to the sales order to allow invoicing from the project.



The second phase of the integration between Field Service and Supply Chain Management enables synchronization with the following templates:

- Warehouses (Supply Chain Management to Field Service) - Warehouses from Supply Chain Management to Field Service [Advanced Query]
- Product Inventory (Supply Chain Management to Field Service) - Inventory level information from Supply Chain Management to Field Service [Advanced Query]
- Inventory Adjustment (Field Service to Supply Chain Management) - Inventory adjustments from Field Service to Supply Chain Management [Advanced Query]
- Inventory Transfers (Field Service to Supply Chain Management) - Inventory transfers from Field Service to Supply Chain Management [Advanced Query]
- Projects (Supply Chain Management to Field Service) - Project list from Supply Chain Management to Field Service

- Work Orders with Project (Field Service to Supply Chain Management) - Work orders in Field Service to Sales orders in Supply Chain Management, with support for Project [Advanced Query]
- Field Service Products with Inventory unit (Supply Chain Management to Sales) - Supply Chain Management 'Sellable released products' to Sales 'Products' for Field Service, including Inventory unit

System requirements

System requirements for Supply Chain Management

Field Service integration supports the following versions:

- Dynamics 365 for Finance and Operations version 8.1.2 (December 2018) was released in December 2018 and has an application build number 8.1.195 with Platform update 22 (7.0.5095).

System requirements for Field Service

To use the Field Service integration solution, you must install the following components:

- Field Service (version 8.2.0.286) or a later version on Dynamics 365 9.1.x - Released November 2018
- Prospect to Cash (P2C) solution for Dynamics 365, version 1.15.0.1 or a later version. The solution is available for download from AppSource.
- 'Field Service Integration, Project and Inventory' solution for Dynamics 365, version 2.0.0.0 or a later version. The solution is available for download from AppSource.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Synchronize products in Supply Chain Management to products in Field Service

2/18/2021 • 2 minutes to read • Edit Online

This topic discusses the templates and underlying task that are used to synchronize products from Dynamics 365 Supply Chain Management to Dynamics 365 Field Service.

The used **Field Service Products (Supply Chain Management to Field Service)** template is based on the **Products (Supply Chain Management to Sales) – Direct** template from Prospect to Cash. For more information, see Products (Supply Chain Management to Sales) – Direct.

This topic only describes the differences between the **Field Service Products (Supply Chain Management to Field Service)** and **Products (Supply Chain Management to Sales) – Direct** templates.

Templates and tasks

Name of the template in Data integration

- Field Service Products (Supply Chain Management to Field Service)

Name of the task in the Data integration project

- Products - Products

The **Field Service Products (Supply Chain Management to Field Service)** template includes one mapping that isn't included in the **Products (Supply Chain Management to Sales) – Direct** template. This mapping ensures that the required Field Service-specific field **Service Product Type** is set correctly.

```
FIELDSERVICEPRODUCTTYPE        Fn        msdyn_fieldserciveproducttype
```

The following value mapping is used.

```
inventory     :  690970000
nonInventory  :  690970001
service       :  690970002
```

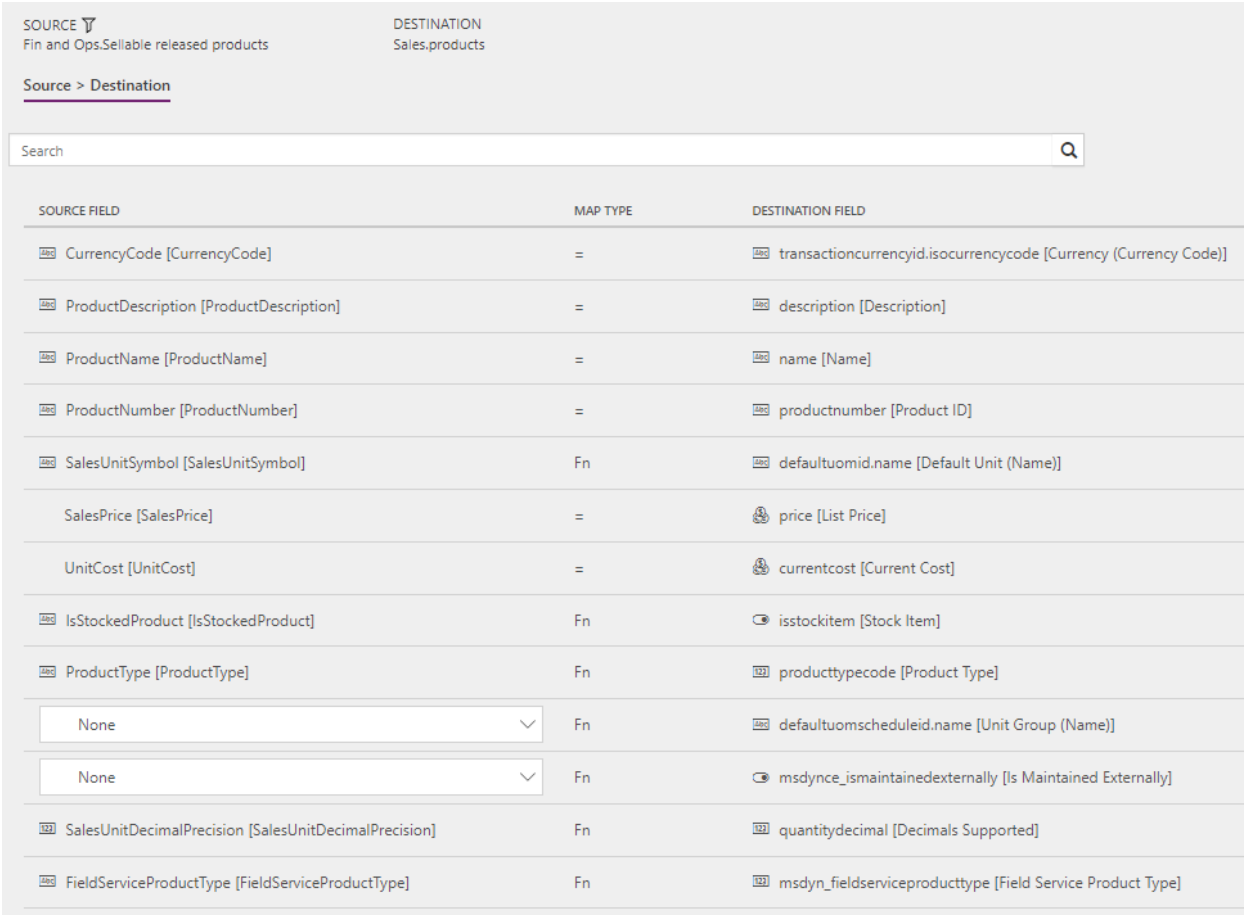
In Supply Chain Management, the **Field Service product type** value on the **Sellable released products** data entity is calculated as follows:

- **Inventory:** Product type = Product and Item model group, Stocked product = True
- **NonInventory:** Product type = Product and Item model group, Stocked product = False
- **Service:** Product type = Service

Template mapping in Data integration

The following illustrations show the template mapping in Data integration.

Field Service Products (Supply Chain Management to Field Service): Products - Products



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Synchronize work orders in Field Service to sales orders in Supply Chain Management

2/18/2021 • 13 minutes to read • Edit Online

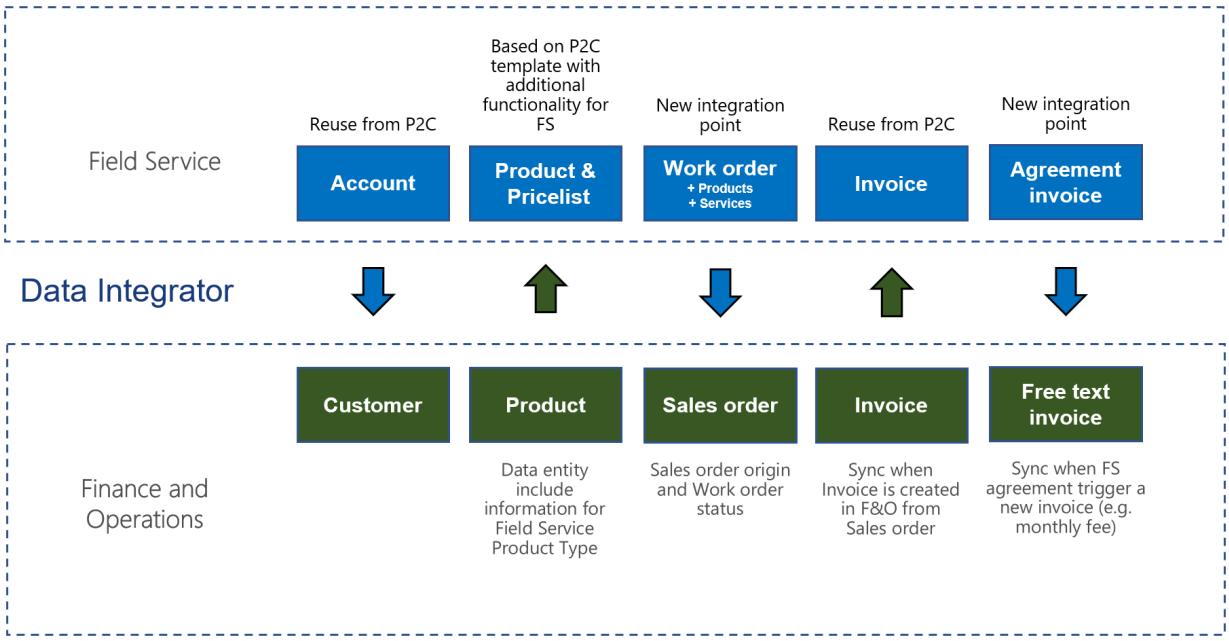
NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

This topic discusses the templates and underlying tasks that are used to synchronize work orders in Dynamics 365 Field Service to sales order in Dynamics 365 Supply Chain Management.



Templates and tasks

The following templates and underlying tasks are used to run the synchronization of work orders in Field Service to sales orders in Supply Chain Management.

Names of the templates in Data integration

The **Work orders to Sales orders (Field Service to Supply Chain Management)** template is used to run synchronization.

Names of the tasks in the Data integration project

- WorkOrderHeader
- WorkOrderServiceLineEstimate
- WorkOrderServiceLineUsed
- WorkOrderProductLineEstimate

- WorkOrderProductLineUsed

The following synchronization tasks are required before synchronization of sales order headers and lines can occur:

- Field Service Products (Supply Chain Management to Field Service)
- Accounts (Sales to Supply Chain Management) – Direct

Entity set

FIELD SERVICE	SUPPLY CHAIN MANAGEMENT
msdyn_workorders	Dataverse sales order headers
msdyn_workorderservices	Dataverse sales order lines
msdyn_workorderproducts	Dataverse sales order lines

Entity flow

Work orders are created in Field Service. If the work orders include only externally maintained products, and if the **Work order status** value differs from **Open-Unscheduled** and **Closed – Cancelled**, the work orders can be synchronized to Supply Chain Management via a Microsoft Dataverse Data integration project. Updates on the work orders will be synchronized as sales orders in Supply Chain Management. These updates include the information about the origin type and status.

Estimated versus Used

In Field Service, products and services on work orders have both **Estimated** values and **Used** values for quantities and amounts. However, in Supply Chain Management, sales orders don't have the same concept of **Estimated** and **Used** values. To support product allocation that uses the expected quantity on the sales order in Supply Chain Management, but to keep the used quantity that should be consumed and invoiced, two sets of tasks synchronize the products and services on the work order. One set of tasks is for **Estimated** values, and the other set of tasks is for **Used** values.

This behavior enables scenarios where estimated values are used for allocation or reservation in Supply Chain Management, whereas used values are used for consumption and invoicing.

Estimated

For synchronization of product lines, the **Estimated** values are used when the **Line Status** value is **Estimated**, the **Allocated** option is set to **Yes**, and **System Status** value isn't **Closed – Posted**.

For synchronization of service lines, the **Estimated** values are used when the **Line Status** value is **Estimated** and the **System Status** value isn't **Closed – Posted**.

Used

The **Used** values are used for consumption and invoicing. In these cases, the **Used** values are synchronized.

The following table provides an overview of the various combinations for product lines.

SYSTEM STATUS (FIELD SERVICE)	LINE STATUS (FIELD SERVICE)	ALLOCATED (FIELD SERVICE)	SYNCHRONIZED VALUE (SUPPLY CHAIN MANAGEMENT)
Open - Scheduled	Estimated	Yes	Estimated

SYSTEM STATUS (FIELD SERVICE)	LINE STATUS (FIELD SERVICE)	ALLOCATED (FIELD SERVICE)	SYNCHRONIZED VALUE (SUPPLY CHAIN MANAGEMENT)
Open - Scheduled	Estimated	No	Used
Open - Scheduled	Used	Yes	Used
Open - Scheduled	Used	No	Used
Open - In Progress	Estimated	Yes	Estimated
Open - In Progress	Estimated	No	Used
Open - In Progress	Used	Yes	Used
Open - In Progress	Used	No	Used
Open - Completed	Estimated	Yes	Estimated
Open - Completed	Estimated	No	Used
Open - Completed	Used	Yes	Used
Open - Completed	Used	No	Used
Closed - Posted	Estimated	Yes	Used
Closed - Posted	Estimated	No	Used
Closed - Posted	Used	Yes	Used
Closed - Posted	Used	No	Used

The following table provides an overview of the various combinations for service lines.

SYSTEM STATUS (FIELD SERVICE)	LINE STATUS (FIELD SERVICE)	SYNCHRONIZED VALUE (SUPPLY CHAIN MANAGEMENT)
Open - Scheduled	Estimated	Estimated
Open - Scheduled	Used	Used
Open - In Progress	Estimated	Estimated
Open - In Progress	Used	Used
Open - Completed	Estimated	Estimated
Open - Completed	Used	Used
Closed - Posted	Estimated	Used

SYSTEM STATUS (FIELD SERVICE)	LINE STATUS (FIELD SERVICE)	SYNCHRONIZED VALUE (SUPPLY CHAIN MANAGEMENT)
Closed - Posted	Used	Used

Synchronization of **Estimated** values versus **Used** values is managed through the two sets of tasks for product lines and service lines. Predefined filters trigger the correct task, and the underlying mapping helps guarantee that the correct values for **Expected** versus **Used** are synchronized.

Example

1. A work order is created and scheduled in Field Service.

   The **System Status** value is **Open – Scheduled**.

   - **Product line:** Estimated Qty = 5ea, Used Qty = 0ea, Line Status = Estimated, Allocated = No
   - **Service line:** Estimated Qty = 2h, Used Qty = 0h, Line Status = Estimated

   In this example, the product's **Used Qty** value of **0** (zero) and the service's **Estimated Qty** value of **2h** are synchronized to Supply Chain Management.

2. Products are allocated in Field Service.

   The **System Status** value is **Open – Scheduled**.

   - **Product line:** Estimated Qty = 5ea, Used Qty = 0ea, Line Status = Estimated, Allocated = Yes
   - **Service line:** Estimated Qty = 2h, Used Qty = 0h, Line Status = Estimated

   In this example, the product's **Estimated Qty** value of **5ea** and the service's **Estimated Qty** value of **2h** are synchronized to Supply Chain Management.

3. The service technician starts to work on the work order and registers material usage of 6.

   The **System Status** value is **Open – In Progress**.

   - **Product line:** Estimated Qty = 5ea, Used Qty = 6ea, Line Status = Used, Allocated = Yes
   - **Service line:** Estimated Qty = 2h, Used Qty = 0h, Line Status = Estimated

   In this example, the product's **Used Qty** value of **6** and the service's **Estimated Qty** value of **2h** are synchronized to Supply Chain Management.

4. The service technician completes the work order and registers used time of 1.5 hours.

   The **System Status** value is **Open – Completed**.

   - **Product line:** Estimated Qty = 5ea, Used Qty = 6ea, Line Status = Used, Allocated = Yes
   - **Service line:** Estimated Qty = 2h, Used Qty = 1.5h, Line Status = Used

   In this example, the product's **Used Qty** value of **6** and the service's **Used Qty** of **1.5h** are synchronized to Supply Chain Management.

Sales order origin and status

Sales origin

To keep track of sales orders that originate from work orders, you can create a sales origin where the **Origin type assignment** option is set to **Yes** and the **Sales origin type** field is set to **Work order integration**.

By default, the mapping selects the sales origin for the **Work order integration** sales origin type for all sales orders that are created from work orders. This behavior can be useful when you work with the sales order in Supply Chain Management. You must make sure that sales orders that originate from work orders aren't synchronized back to Field Service as work orders.

For details about how to create the correct sales origin setup in Supply Chain Management, see the "Preconditions and mapping setup" section of this topic.

Status

When the sales order originates from a work order, the **External work order status** field appears on the **Setup** tab on the sales order header. This field shows the system status from the work order in Field Service, to help track the synchronized work order status of sales orders in the Supply Chain Management. This field can also help the user determine when the sales order should be shipped or invoiced.

The **External work order status** field can have the following values:

- Open - Scheduled
- Open - In Progress
- Open - Completed
- Closed - Posted

Field Service CRM solution

To support the integration between Field Service and Supply Chain Management, additional functionality from the Field Service CRM solution is required. The solution includes the following changes.

Work Order entity

The **Has Externally Maintained Products Only** field has been added to the **Work Order** entity and appears on the page. It's used to consistently track whether a work order consists entirely of externally maintained products. A work order consists entirely of externally maintained products when all the related products are maintained in Supply Chain Management. This field helps guarantee that users don't synchronize work orders that have products that are unknown.

Work Order Product entity

- The **Order Has Externally Maintained Products Only** field has been added to the **Work Order Product** entity and appears on the page. It's used to consistently track whether the work order product is maintained in Supply Chain Management. This field helps guarantee that users don't synchronize work order products that are unknown to Supply Chain Management.
- The **Header System Status** field has been added to the **Work Order Product** entity and appears on the page. It's used to consistently track the system status of the work order and helps guarantee correct filtering when work order products are synchronized to Supply Chain Management. When filters are set on the integration tasks, **Header System Status** information is also used to determine whether the estimated or used values should be synchronized.
- The **Invoiced Unit Amount** field shows the amount that is invoiced per actual unit that is used. The value is calculated as the **Total Amount** value divided by the **Actual Quantity** value. The field is used for integration to systems that don't support different values for the used quantity and the billed quantity. This field doesn't appear in the user interface (UI).
- The **Invoiced Discount Amount** field is calculated as the **Discount Amount** value plus the rounding from the calculation of the **Invoiced Unit Amount** value. This field is used for integration and doesn't appear in the UI.
- The **Decimal Quantity** field stores the value from the **Quantity** field as a decimal number. This field is used for integration and doesn't appear in the UI.
- The value in **Used** fields is reset to **0** (zero) if the **Line Status** value of the work order product is changed from **Used** to **Estimated**. This change helps prevent situations where a used quantity that is mistakenly entered is used for synchronization when the **Line Status** value is **Estimated** and the **Allocated** option is set to **No**.

Work Order Service entity

- The **Order Has Externally Maintained Products Only** field has been added to the **Work Order Service** entity and appears on the page. It's used to consistently track whether the work order service is maintained in Supply Chain Management. This field helps guarantee that users don't synchronize work order services that are unknown to Supply Chain Management.
- The **Header System Status** field has been added to the **Work Order Service** entity and appears on the page. It's used to consistently track the system status of the work order and helps guarantee correct filtering when work order services are synchronized to Supply Chain Management. When filters are set on the integration tasks, **Header System Status** information is also used to determine whether the estimated or used values should be synchronized.
- The **Duration In Hours** field stores the value from the **Duration** field after that value is converted from minutes to hours. This field is used for integration and doesn't appear in the UI.
- The **Estimated Duration In Hours** field stores the value from the **Estimated Duration** field after that value is converted from minutes to hours. This field is used for integration and doesn't appear in the UI.
- The **Invoiced Unit Amount** field stores the amount that is invoiced per actual unit that is used. The value is calculated as the **Total Amount** value divided by the **Actual Quantity** value. This field is used for integration to systems that don't support different values for the used quantity and the billed quantity. The field doesn't appear in the UI.
- The **Invoiced Discount Amount** field is calculated as the **Discount Amount** value plus the rounding from the calculation of the **Invoiced Unit Amount** value. This field is used for integration and doesn't appear in the UI.
- The **External Line Order** field is a calculated negative line order number that can be used in external systems where work order product and service lines are combined. This field enables work order products that are inserted to have positive line numbers and work order services to have negative line numbers. The value of this this field is calculated as the **Line Order** value multiplied by -1. This field doesn't appear in the UI.
- The value in **Used** fields is reset to **0** (zero) if the **Line Status** value of the work order service is changed from **Used** to **Estimated** for some reason. This change helps prevent situations where a used quantity that is mistakenly entered is used for synchronization when the **Line Status** value is **Estimated** and the **Header System Status** value is **Closed – Posted**.

Preconditions and mapping setup

Before you synchronize work orders, it's important that you update the following settings in the systems.

Setup in Field Service

- Make sure that the number series that is used for work orders in Field Service doesn't overlap the number sequence that is used for sales orders in Supply Chain Management. Otherwise, existing sales orders can be incorrectly updated in Field Service or Supply Chain Management.
- The **Work Order Invoice Creation** field must be set to **Never**, because the invoicing will be done from Supply Chain Management. Go to **Field Service** > **Settings** > **Administration** > **Field Service Settings**, and make sure that the **Work Order Invoice Creation** field is set to **Never**.

Setup in Supply Chain Management

Work order integration requires that you set up the sales origin. The sales origin is used to distinguish sales orders in Supply Chain Management that were created from work orders in Field Service. When a sales order has a sales origin of the **Work order integration** type, the **External work order status** field appears on the sales order header. Additionally, the sales origin helps guarantee that sales orders that were created from work orders in Field Service are filtered out during sales order synchronization from Supply Chain Management to Field Service.

1. Go to **Sales and marketing** > **Setup** > **Sales orders** > **Sales origin**.
2. Select **New** to create a new sales origin.

3. In the **Sales origin** field, enter a name for the sales origin, such as **WorkOrder**.
4. In the **Description** field, enter a description, such as **Field Service Work Order**.
5. Select the **Origin type assignment** check box.
6. Set the **Sales origin type** field to **Work order integration**.
7. Select **Save**.

Setup in Data integration

Ensure the **Integration key** exist for **msdyn_workorders**

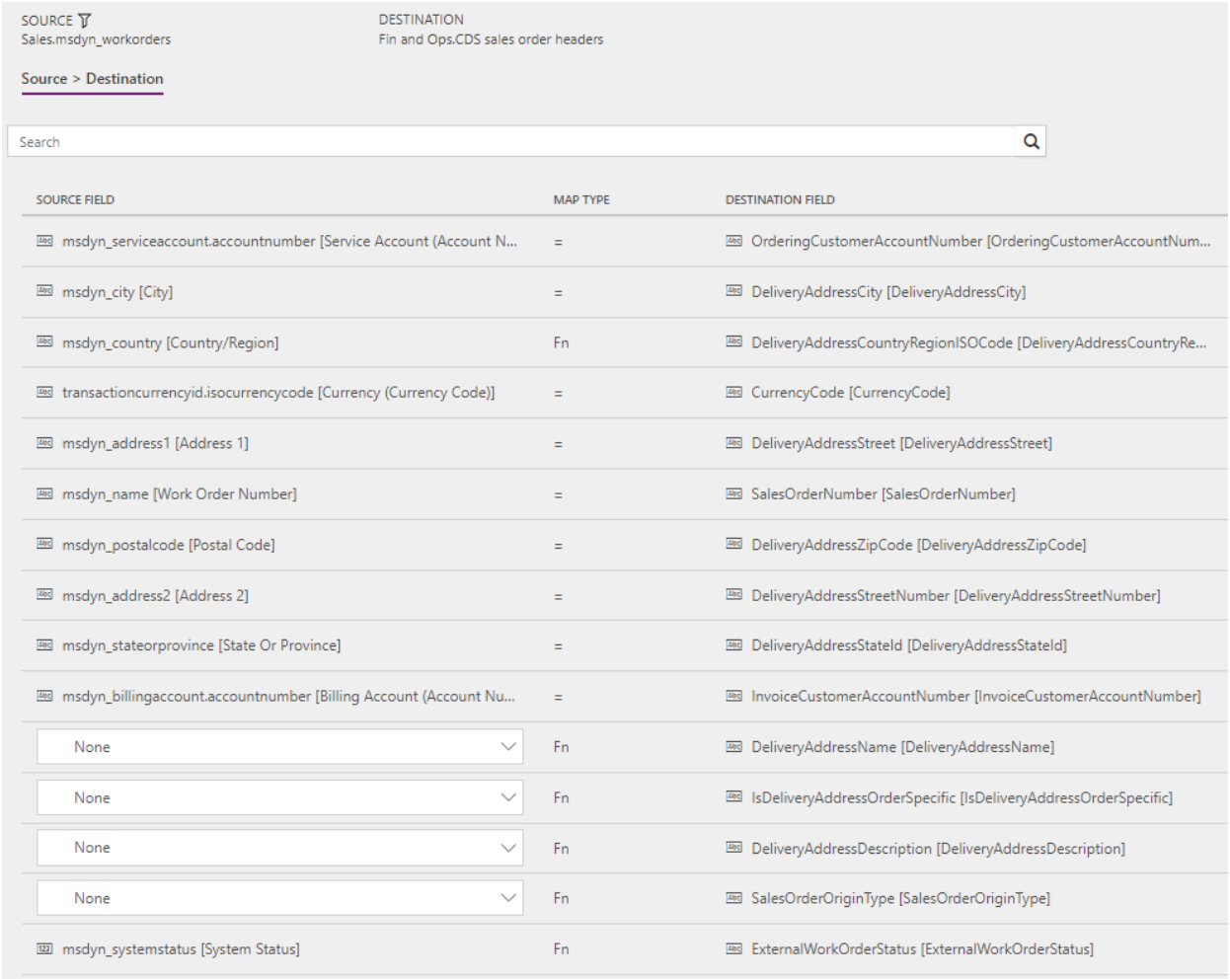
1. Go to Data Integration
2. Select **Connection Set** tab
3. Select the Connection set used for Work order synchronization
4. Select **Integration key** tab
5. Find msdyn_workorders and check that the key **msdyn_name (Work Order Number)** is added. If it is not shown, add it by click **Add key** and click **Save** in the top of the page

Template mapping in Data integration

The following illustrations show the template mapping in Data integration.

Work orders to Sales orders (Field Service to Supply Chain Management): WorkOrderHeader

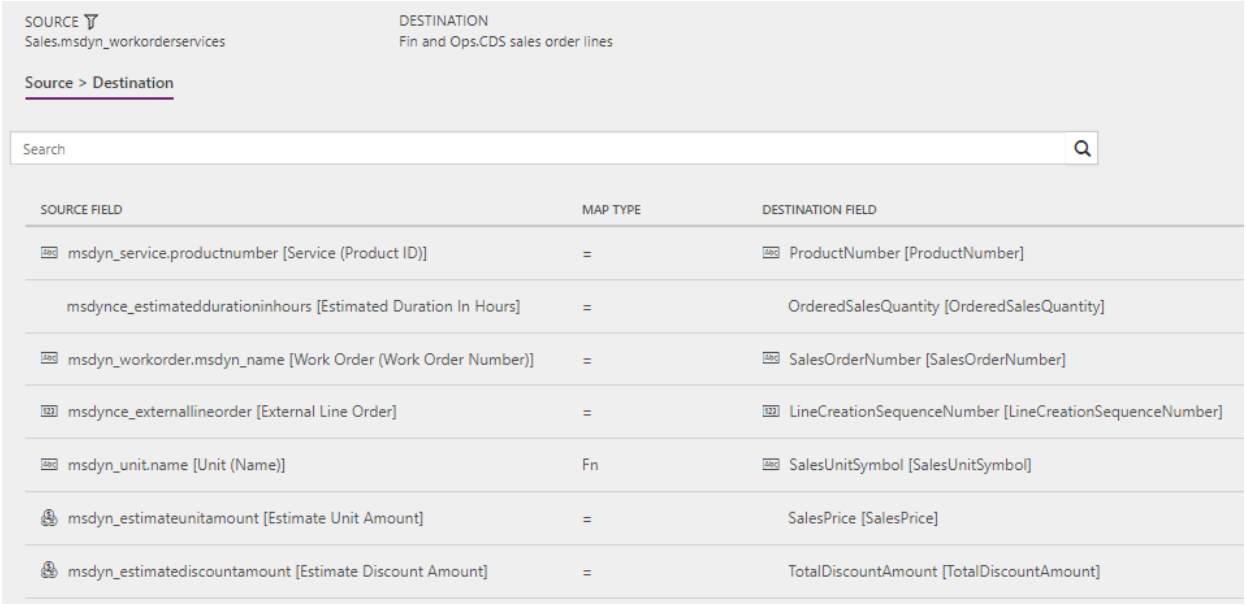
Filter: (msdyn_systemstatus ne 690970005) and (msdyn_systemstatus ne 690970000) and (msdynce_hasexternallymaintainedproductsonly eq true)



Work orders to Sales orders (Field Service to Supply Chain Management): WorkOrderServiceLineEstimate

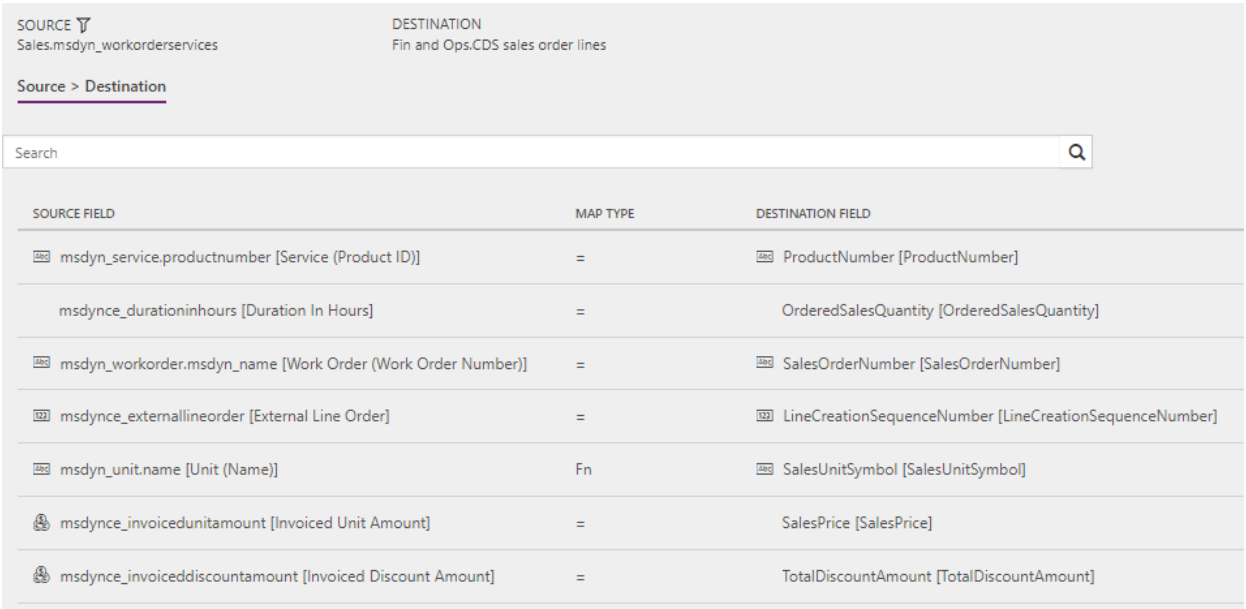
Filter: (msdynce_headersystemstatus ne 690970005) and (msdynce_headersystemstatus ne 690970000) and (msdynce_orderhasexternalmaintainedproductsonly eq true) and (msdyn_linestatus eq 690970000) and

(msdynce_headersystemstatus ne 690970004)



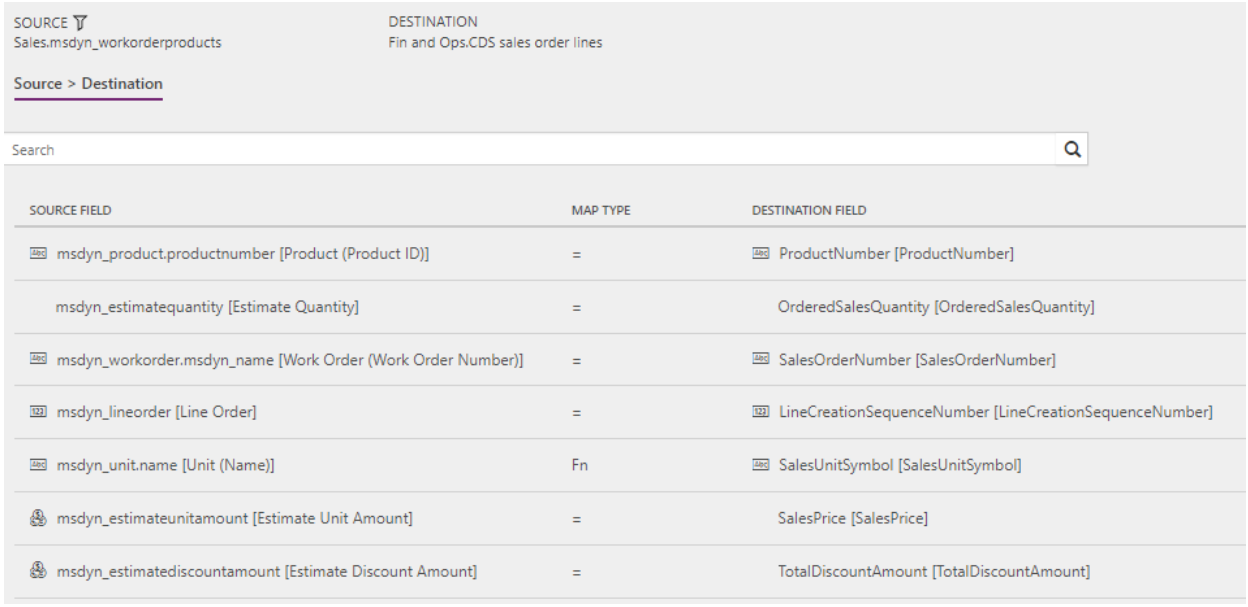
Work orders to Sales orders (Field Service to Supply Chain Management): WorkOrderServiceLineUsed

Filter: (msdynce_headersystemstatus ne 690970005) and (msdynce_headersystemstatus ne 690970000) and (msdynce_orderhasexternalmaintainedproductsonly eq true) and ((msdyn_linestatus eq 690970001) or (msdynce_headersystemstatus eq 690970004))



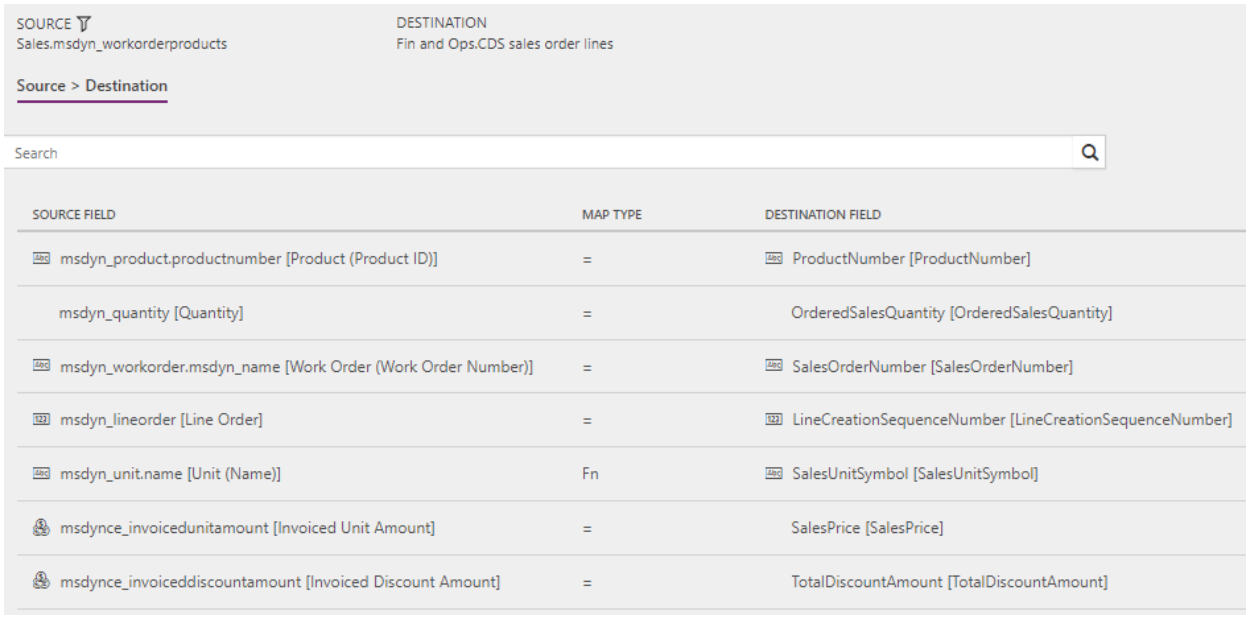
Work orders to Sales orders (Field Service to Supply Chain Management): WorkOrderProductLineEstimate

Filter: (msdynce_headersystemstatus ne 690970005) and (msdynce_headersystemstatus ne 690970000) and (msdynce_orderhasexternalmaintainedproductsonly eq true) and (msdyn_linestatus eq 690970000) and (msdynce_headersystemstatus ne 690970004) and (msdyn_allocated eq true)



Work orders to Sales orders (Field Service to Supply Chain Management): WorkOrderProductLineUsed

Filter: (msdynce_headersystemstatus ne 690970005) and (msdynce_headersystemstatus ne 690970000) and (msdynce_orderhasexternalmaintainedproductsonly eq true) and ((msdyn_linestatus eq 690970001) or (msdynce_headersystemstatus eq 690970004) or (msdyn_allocated ne true))



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Synchronize agreement invoices in Field Service to free text invoices in Supply Chain Management

2/18/2021 • 3 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

This topic discusses the templates and underlying tasks that are used to synchronize agreement invoices in Dynamics 365 Field Service to free text invoices in Dynamics 365 Supply Chain Management.

Templates and tasks

The following template and underlying tasks are used to run synchronization of agreement invoices from Field Service to free text invoices in Supply Chain Management.

Name of the template in Data integration

- Agreement invoices (Field Service to Supply Chain Management)

Names of the tasks in the Data integration project

- Invoice headers
- Invoice lines

The following synchronization is required before synchronization of agreement invoices can occur:

- Accounts (Sales to Supply Chain Management) – Direct

Entity set

FIELD SERVICE	SUPPLY CHAIN MANAGEMENT
invoices	Dataverse customer free text invoice headers
invoicedetails	Dataverse customer free text invoice lines

Entity flow

Invoices that are created from an agreement in Field Service can be synchronized to Supply Chain Management via a Microsoft Dataverse Data integration project. Updates to these invoices will be synchronized to the free text invoices in Supply Chain Management if the accounting status of the free text invoices is **In process**. After the free text invoices are posted in Supply Chain Management, and the accounting status is updated to **Completed**, you can no longer synchronize updates from Field Service.

Field Service CRM solution

The **Has Lines With Agreement Origin** column has been added to the **Invoice** table. This column helps guarantee that only invoices that are created from an agreement are synchronized. The value is **true** if the invoice contains at least one invoice line that originates from an agreement.

The **Has Agreement Origin** column has been added to the **Invoice Line** table. This column helps guarantee that only invoice lines that are created from an agreement are synchronized. The value is **true** if the invoice line originates from an agreement.

Invoice date is a mandatory field in Supply Chain Management. Therefore, the column must have a value in Field Service before synchronization occurs. To meet this requirement, the following logic is added:

- If the **Invoice Date** column is blank on the **Invoice** table (that is, if it has no value), it's set to the current date when an invoice line that originates from an agreement is added.
- The user can change the **Invoice Date** column. However, when the user tries to save an invoice that originates from an agreement, he or she receives a business process error if the **Invoice Date** column is blank on the invoice.

Prerequisites and mapping setup

In Supply Chain Management

An invoice origin must be set up for the integration, to distinguish the free text invoices in Supply Chain Management that are created from agreement invoices in Field Service. When an invoice has an invoice origin of the **Agreement invoice integration** type, the **External invoice number** field is shown on the **Sales invoice** header.

Besides appearing on the invoice header, the **External invoice number** information can be used to help guarantee that invoices that are created from Field Service agreement invoices are filtered out during invoice synchronization from Supply Chain Management to Field Service.

1. Go to **Accounts receivable** > **Setup** > **Invoice origin codes**.
2. Select **New** to create a new invoice origin.
3. In the **Invoice origin** field, enter a name for the invoice origin, such as **FS**.
4. In the **Description** field, enter a description, such as **Field Service Agreement Invoice**.
5. Select the **Origin type assignment** check box.
6. Set the **Invoice origin type** field to **Agreement invoice integration**.
7. Select **Save**.

In the Data Integration project

Task: **Invoice lines**

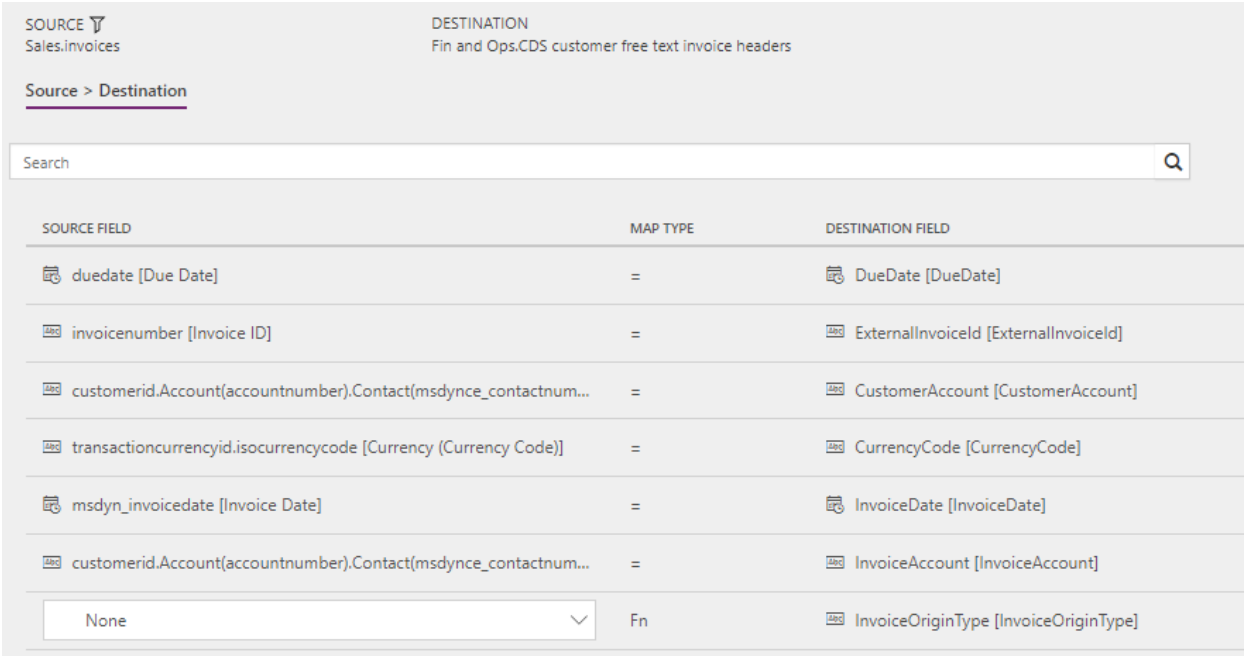
Make sure that the default value for the Supply Chain Management field **Main Account Display Value** is updated to match the desired value.

The default template value is **401100**.

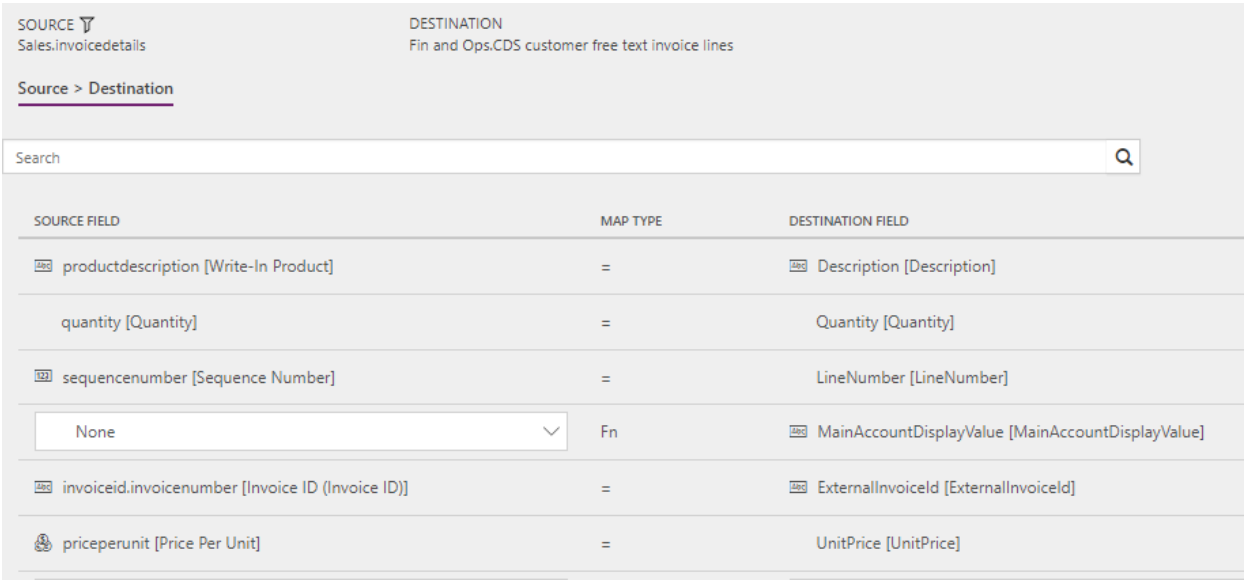
Template mapping in Data integration

The following illustrations show the template mapping in Data integration.

Agreement invoices (Field Service to Supply Chain Management): Invoice headers



Agreement invoices (Field Service to Supply Chain Management): Invoice lines



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Synchronize inventory transfers and adjustments from Field Service to Supply Chain Management

2/18/2021 • 3 minutes to read • Edit Online

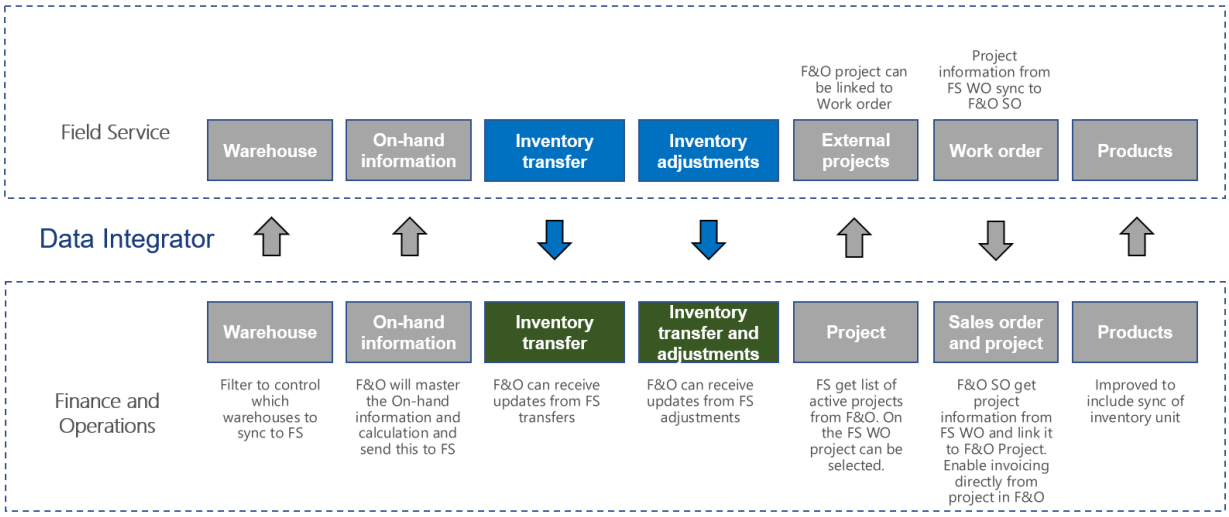
NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

This topic discusses the templates and underlying tasks that are used to synchronize inventory adjustments and transfers from Dynamics 365 Supply Chain Management to Dynamics 365 Field Service.



Templates and tasks

The following template and underlying tasks are used to synchronize inventory adjustments and transfers from Field Service to Supply Chain Management.

Templates in Data integration

- Inventory Adjustment (Field Service to Supply Chain Management)
- Inventory Transfers (Field Service to Supply Chain Management)

Tasks in the Data integration projects

- Inventory Adjustments
- Inventory Transfers

Table set

FIELD SERVICE	SUPPLY CHAIN MANAGEMENT
msdyn_inventoryadjustmentproducts	Dataverse Inventory adjustment journal headers and lines
msdyn_inventoryadjustmentproducts	Dataverse inventory transfer journal headers and lines

Table flow

Inventory adjustments and transfers made in Field Service will synchronize to Supply Chain Management after the **Post status** changes from **Created** to **Posted**. When this occurs, the adjustment or the transfer order will be locked and become read only. This means that adjustments and transfers can be posted in Supply Chain Management, but cannot be modified. In Supply Chain Management, you can set up a batch job to automatically post the adjustments and transfer inventory journals that have been generated during the integration. See the following prerequisites for details on how to enable the batch job.

Field Service CRM solution

The **Inventory unit** column has been added to the **Product** table. This column is needed because the Sales and Inventory unit is not always the same in Supply Chain Management, and the Inventory Unit is needed for the Warehouse Inventory in Supply Chain Management. When you set the product on an Inventory adjustment product for both Inventory adjustments and Inventory transfers, the unit will be fetched from the inventory product value. If a value is found, the **Unit** column will be locked on the Inventory adjustment product.

The **Post status** column has been added to both the **Inventory adjustment** table and the **Inventory transfer** table. This column is used as a filter when an adjustment or transfer is sent to Supply Chain Management. The default for this column is Created (1), however it is not sent to Supply Chain Management. When you update the value to Posted (2), it is sent to Supply Chain Management, but after that you will no longer be able to change the adjustment or transfer or add new lines.

The **Number sequence** column has been added to the **Inventory adjustment product** table. This column ensures that the integration has a unique number, so the integration can create and update the adjustment. When you create your first inventory adjustment product, it will create a new record in the **P2C AutoNumber** table to maintain the number series and the prefix that is used.

Prerequisites and mapping setup

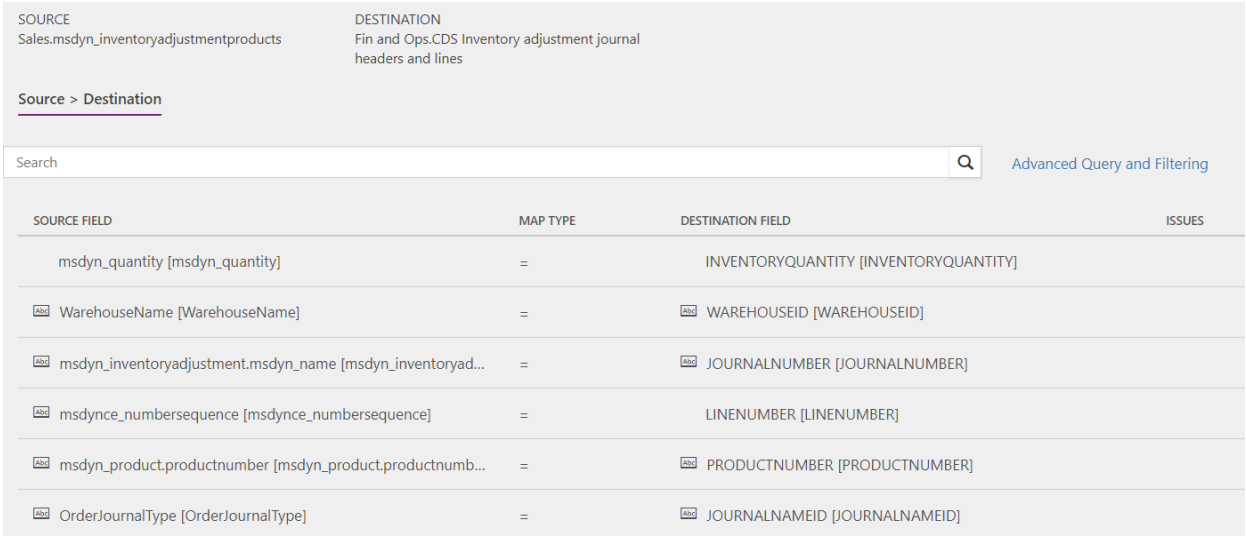
Supply Chain Management

The integration inventory journals generated by the integration can automatically be posted using a batch job. This is enabled from **Inventory management > Periodic tasks > Dataverse integration > Post integration inventory journals**.

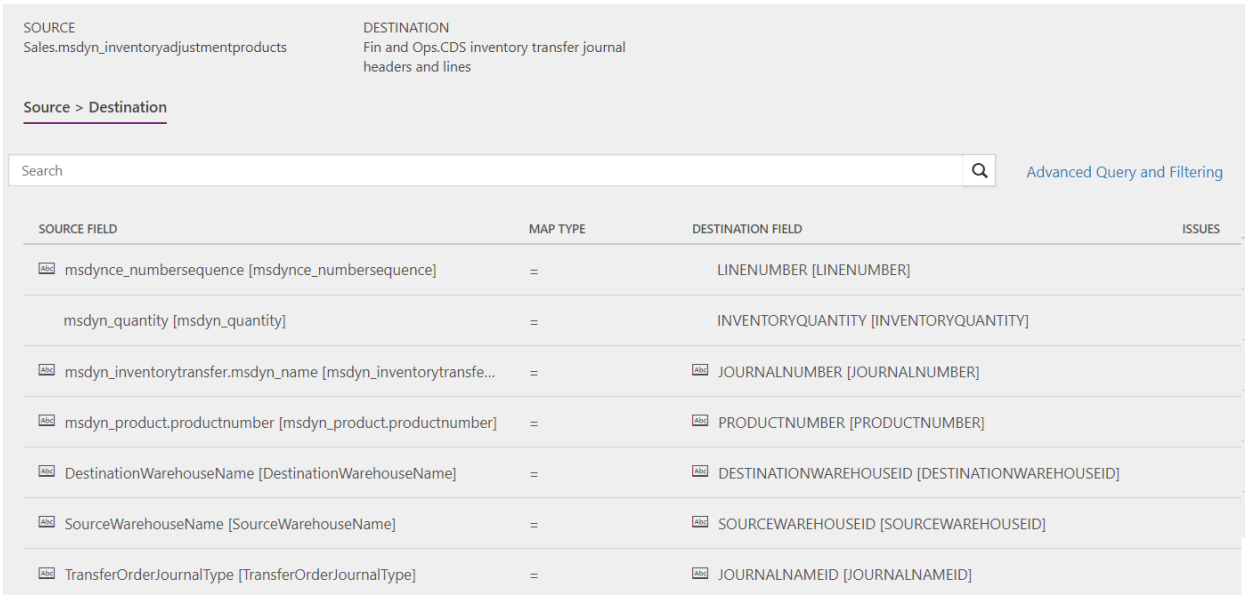
Template mapping in Data integration

The following illustrations show the template mapping in Data integration.

Inventory adjustment (Field Service to Supply Chain Management): Inventory adjustment



Inventory transfer (Field Service to Supply Chain Management): Inventory transfer



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Synchronize inventory level information from Supply Chain Management to Field Service

2/18/2021 • 3 minutes to read • Edit Online

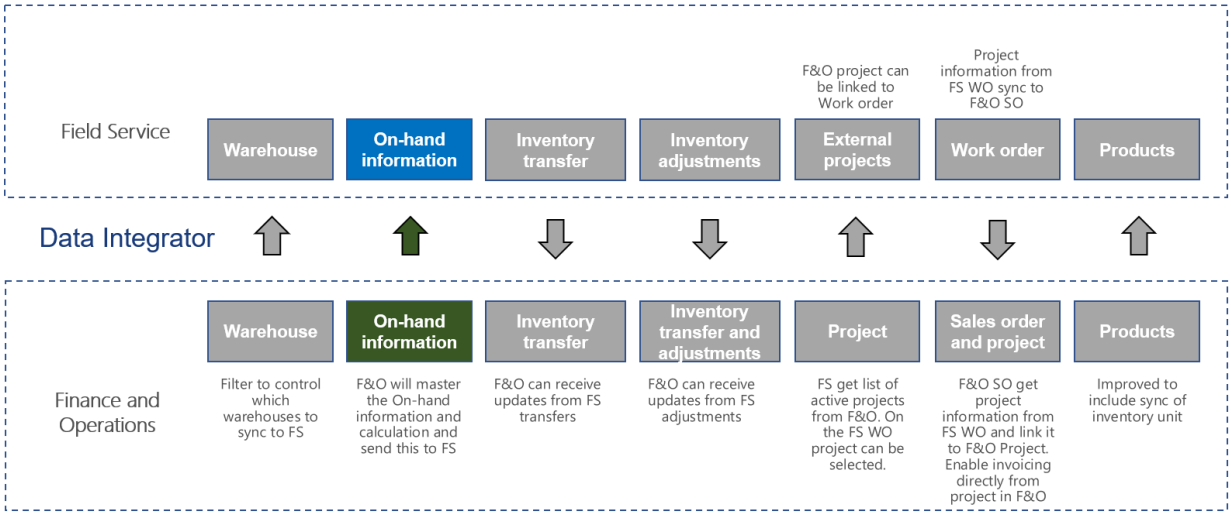
NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

This topic discusses the templates and underlying tasks that are used to synchronize inventory-level information from Dynamics 365 Supply Chain Management to Dynamics 365 Field Service.



Templates and tasks

The following template and underlying tasks are used to synchronize inventory on-hand levels from Supply Chain Management to Field Service.

Template in Data integration

- Product Inventory (Supply Chain Management to Field Service)

Task in the Data integration project

- Product inventory

The following synchronization tasks are required before synchronization of inventory levels can occur:

- Warehouses (Supply Chain Management to Field Service)
- Field Service products with Inventory unit (Supply Chain Management to Sales)

Entity set

FIELD SERVICE	SUPPLY CHAIN MANAGEMENT
msdynce_externalproductinventories	Dataverse inventory on-hand by warehouse

Entity flow

Inventory-level information from Finance and Operation is sent to Field Service for selected products. The inventory-level information includes:

- On hand quantity (current recorded physical quantity located in the warehouse)
- On order quantity (total recorded quantity on order, such as sales orders)
- Ordered quantity (total recorded ordered quantity, such as purchase orders)

This information is captured per released product for each warehouse and synchronized based on change tracking, when the inventory level changes.

In Field Service, the integration solution creates inventory journals for the delta, so that the levels in Field Service match the levels in Supply Chain Management.

Supply Chain Management will act as the master for inventory levels. Therefore it is important to set up integration for work orders, transfers, and adjustments from Field Service to Supply Chain Management if this functionality is used in Field Service, together with synchronization of inventory levels from Supply Chain Management.

The products and warehouses where inventory levels are mastered from Supply Chain Management can be controlled with the Advanced Query and Filtering (Power Query).

> **NOTE**
>
> It is possible to create multiple warehouses in Field Services (with **Is Externally Maintained = No**) and then map them to a single warehouse in Supply Chain Management, with the Advanced query and filtering functionality. This is used in situations where you want Field Service to master the detailed inventory level and only send updates to Supply Chain Management. In this case, Field Service will not receive inventory-level updates from Supply Chain Management. For additional information, see Synchronize inventory adjustments from Field Service to Supply Chain Management and Synchronize work orders in Field Service to sales orders linked to project in Supply Chain Management.

Field Service CRM solution

The **External product inventory** entity is only used for back end in to the integration. This entity receives the inventory level values from Supply Chain Management in the integration and then transforms those values to Manual inventory journals, which then changes the Inventory products in the Warehouse.

Prerequisites and mapping setup

Data integration

For the project to work, you need to ensure that the Integration key is updated for msdynce_externalproductinventories.

1. Go to **Data integration > Connection sets**.
2. Select the used Connection set.
3. On the **Integration key** tab, ensure that the following keys are added to msdynce_externalproductinventories:
   - msdynce_productnumber (Product Number)

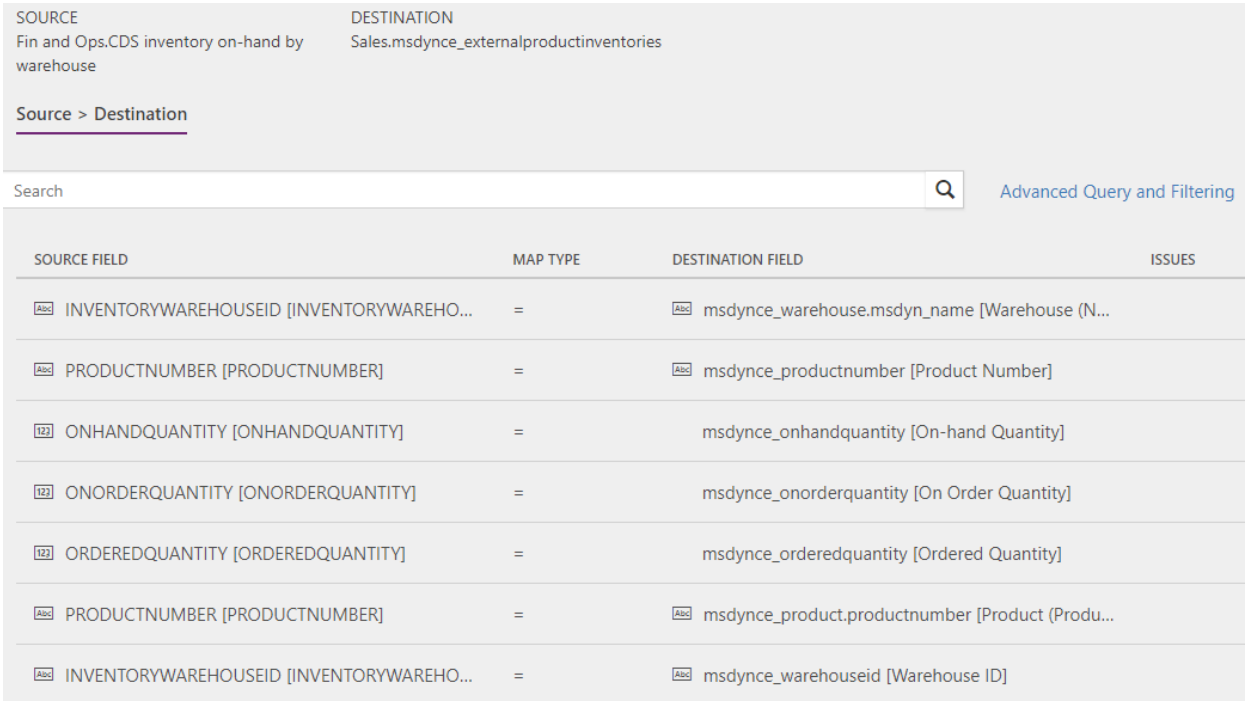
- msdynce_warehouseid (Warehouse ID)

Data integration project

You can apply filters with Advanced Query and Filtering so that only certain products and warehouses send inventory-level information from Supply Chain Management to Field Service.

Template mapping in Data integration

Product inventory (Supply Chain Management to Field Service): Product inventory



NOTE

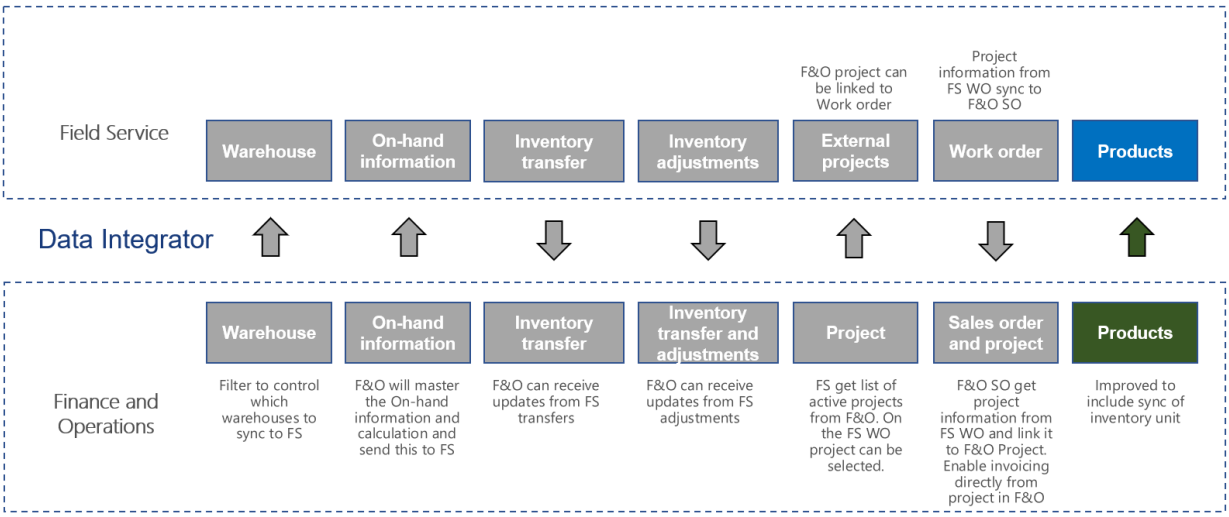
Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Synchronize products with inventory unit from Supply Chain Management to Field Service

2/18/2021 • 2 minutes to read • Edit Online

This topic discusses the templates and underlying task that are used to synchronize products with inventory unit from Dynamics 365 Supply Chain Management to Dynamics 365 Field Service.



The used **Field Service Products with Inventory unit (Supply Chain Management to Field Service)** template is based on the **Field Service Products (Supply Chain Management to Field Service)** template. For more information, see Synchronize products in Supply Chain Management to products in Field Service.

This topic only describes the differences between the two templates:

- **Field Service Products with Inventory unit (Supply Chain Management to Sales)**
- **Field Service Products (Supply Chain Management to Field Service)**

Templates and tasks

Name of the template in Data integration:

- Field Service Products with Inventory unit (Supply Chain Management to Sales)

Name of the task in the Data integration project:

- Products

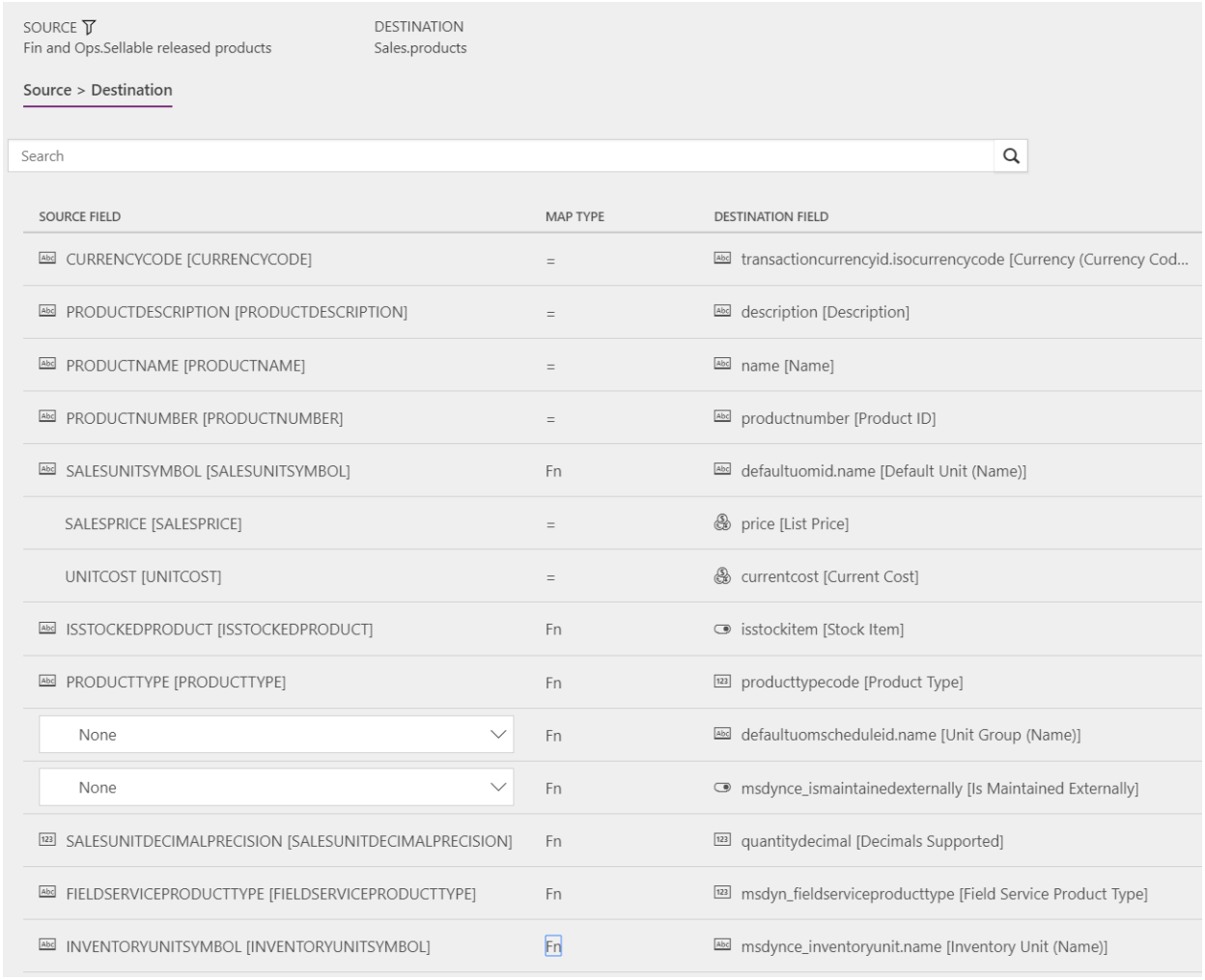
The **Field Service Products with Inventory unit (Supply Chain Management to Field Service)** template includes one mapping that isn't included in the **Field Service Products (Supply Chain Management to Field Service)** template. This mapping ensures that the Inventory unit needed for inventory level synchronization is included.

```
INVENTORYUNITSYMBOL [INVENTORYUNITSYMBOL]        Fn        msdynce_inventoryunit.name [Inventory
Unit(Name)]
```

Template mapping in Data integration

The following illustrations show the template mapping in Data integration.

Field Service Products with Inventory unit (Supply Chain Management to Field Service): Products



NOTE

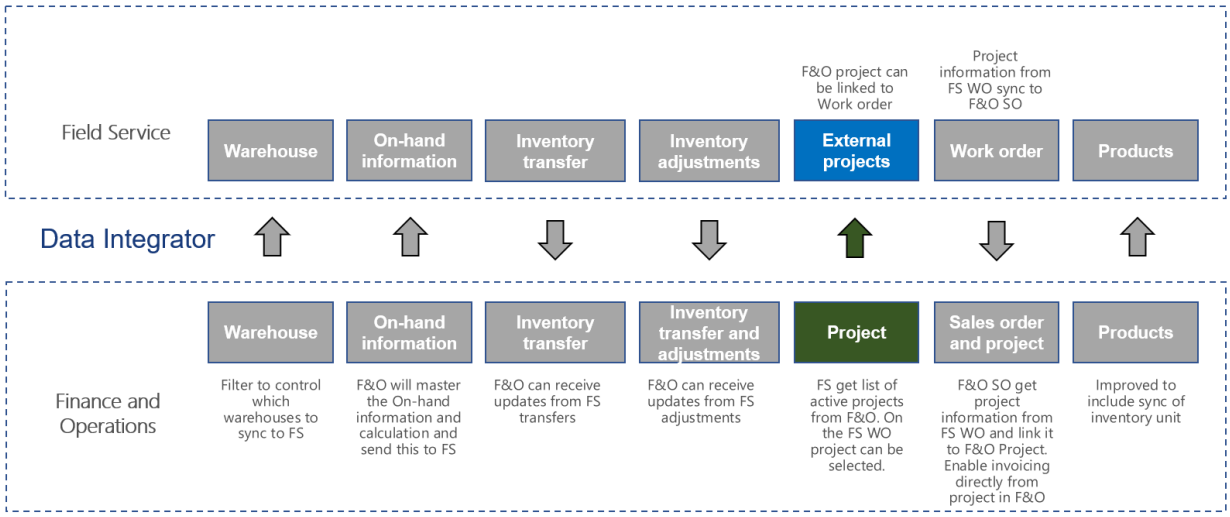
Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Synchronize project list from Supply Chain Management to Field Service

2/18/2021 • 2 minutes to read • Edit Online

This topic discusses the templates and underlying tasks that are used to synchronize projects from Dynamics 365 Supply Chain Management to Dynamics 365 Field Service.



Templates and tasks

The following template and underlying tasks are used to run synchronization of projects from Supply Chain Management to Field Service.

Template in Data integration

- Projects (Supply Chain Management to Field Service)

Task in the Data integration project

- Projects

The following synchronization tasks are required before synchronization of project list can occur:

- Accounts (Sales to Supply Chain Management)

Entity set

FIELD SERVICE	SUPPLY CHAIN MANAGEMENT
msdynce_externalprojects	Projects

Entity flow

Projects are created in Supply Chain Management. Projects with **Project type** set to **Time and material** and **Project stage** set to **In process** will synchronize to the **External Project** entity in Field Service, including Project number, Project name, Project stage, and Customer account information. The **External Project** list is used to pair Field service work orders with Supply Chain Management projects.

Field Service CRM solution

The **External Project** entity gets all the projects from Supply Chain Management. The **External Project** field has been added to the **Work Order** entity. This is a lookup field, so by tagging your work order with a project, the sales order will be connected to a project within Supply Chain Management. After the **System Status** changes **Open – In Progress(690,970,000)** to a higher status, the **External Project** field will be locked and you can no longer add, remove, or change the value.

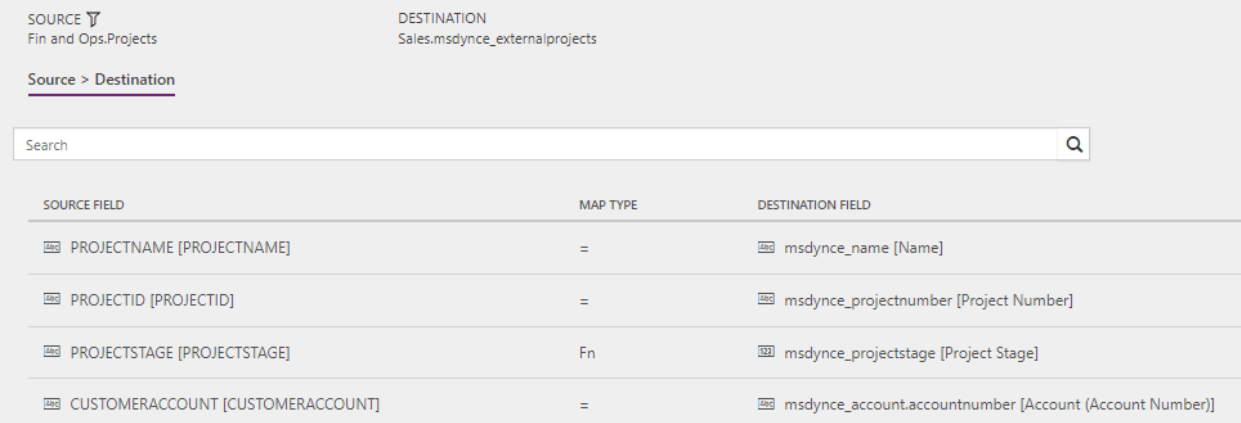
Prerequisites and mapping setup

Supply Chain Management

Enable change tracking for Data entity projects.

Template mapping in Data integration

Projects (Supply Chain Management to Field Service): Projects



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Synchronize warehouses from Supply Chain Management to Field Service

2/18/2021 • 3 minutes to read • Edit Online

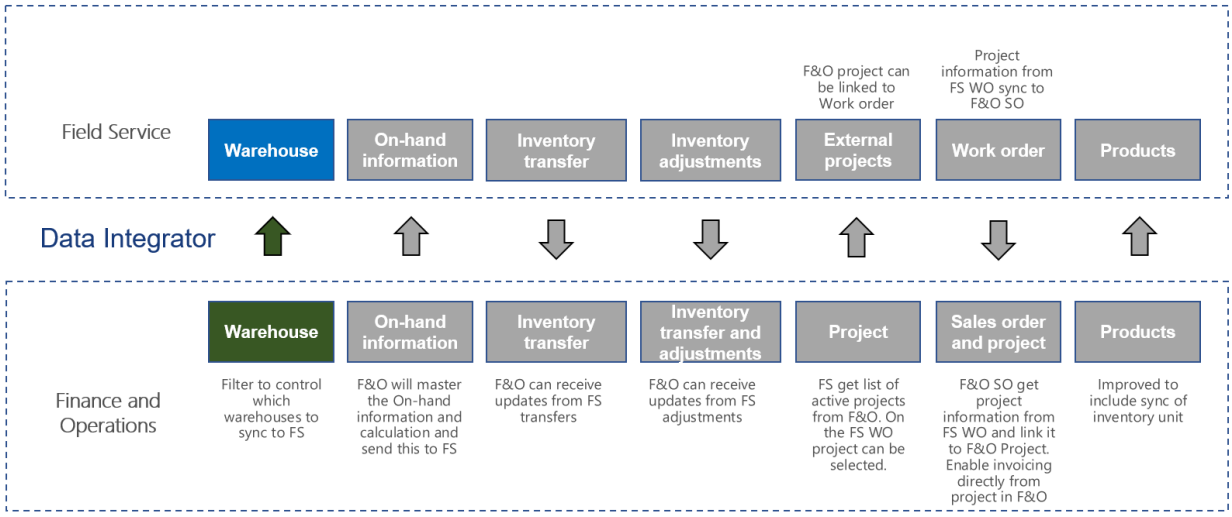
NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

This topic discusses the templates and underlying tasks that are used to synchronize warehouses from Dynamics 365 Supply Chain Management to Dynamics 365 Field Service.



Templates and tasks

The following template and underlying tasks are used to run synchronization of warehouses from Supply Chain Management to Field Service.

Template in Data integration

- Warehouses (Supply Chain Management to Field Service)

Task in the Data integration project

- Warehouse

Table set

FIELD SERVICE	SUPPLY CHAIN MANAGEMENT
msdyn_warehouses	Warehouses

Table flow

Warehouses that are created and maintained in Supply Chain Management can be synchronized to Field Service via a Microsoft Dataverse Data integration project. The warehouses that you want to synchronize to Field Service can be controlled with the Advanced query and filtering on the project. Warehouses that synchronize from Supply Chain Management are created in Field Service with the **Is maintained externally** column set to **Yes** and the record is read only.

Field Service CRM solution

To support the integration between Field Service and Supply Chain Management, additional functionality from the Field Service CRM solution is required. In the solution, the **Is Maintained Externally** column has been added to the **Warehouse (msdyn_warehouses)** table. This column helps to identify if the warehouse is handled from Supply Chain Management or if it only exists in Field Service. The settings for this column include:

- **Yes** – The warehouse originated from Supply Chain Management and won't be editable in Sales.
- **No** – The warehouse was entered directly in Field Service and is maintained here.

The **Is Externally Maintained** column helps control the synchronization of inventory levels, adjustments, transfers, and usage on work orders. Only warehouses with **Is Externally Maintained** set to **Yes** can be used to synchronize directly to the same warehouse in the other system.

> **NOTE**
>
> It is possible to create multiple warehouses in Field Service (with **Is Externally Maintained** = No) and then map them to a single warehouse, with the Advanced query and filtering functionality. This is used in situations where you want Field Service to master the detailed inventory level and just send updates to Supply Chain Management. In this case, Field Service will not receive inventory-level updates from Supply Chain Management. For additional information, see Synchronize inventory adjustments from Field Service to Finance and Operations and Synchronize work orders in Field Service to sales orders linked to project in Finance and Operations.

Prerequisites and mapping setup

Data Integration project

Before synchronizing the warehouses, make sure to update the Advanced query and filtering on the project to only include the warehouses that you want to bring from Supply Chain Management to Field Service. Note that you will need the warehouse in Field Service to apply it on work orders, adjustments, and transfers.

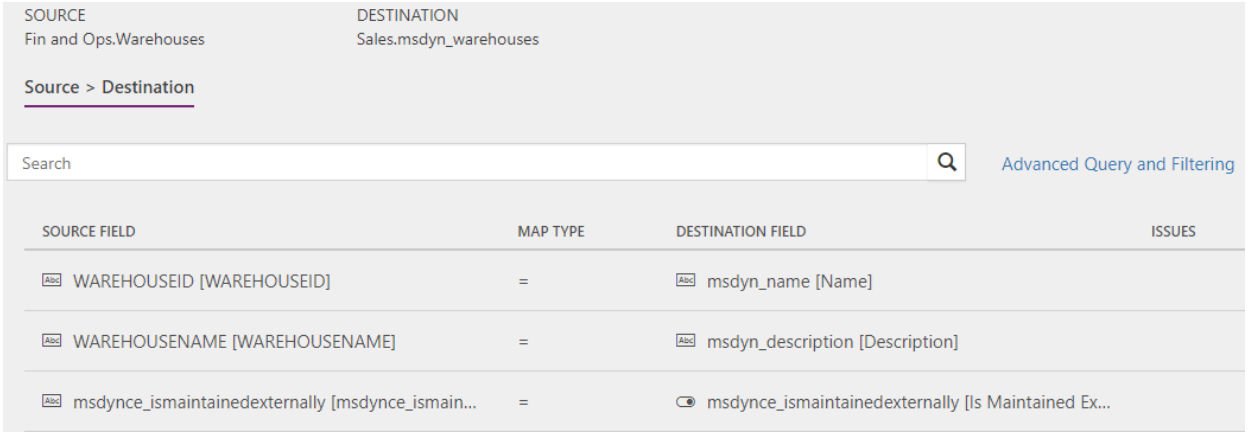
To ensure that the **Integration key** exists for **msdyn_warehouses**:

1. Go to Data Integration.
2. Select the **Connection Set** tab.
3. Select the connection set used for work order synchronization.
4. Select the **Integration key** tab.
5. Find msdyn_warehouses and confirm that the key **msdyn_name (name)** is added. If it is not shown, add it by clicking **Add key** and then click **Save** at the top of the page.

Template mapping in Data integration

The following illustration shows the template mapping in Data integration.

Warehouses (Supply Chain Management to Field Service): Warehouse



NOTE

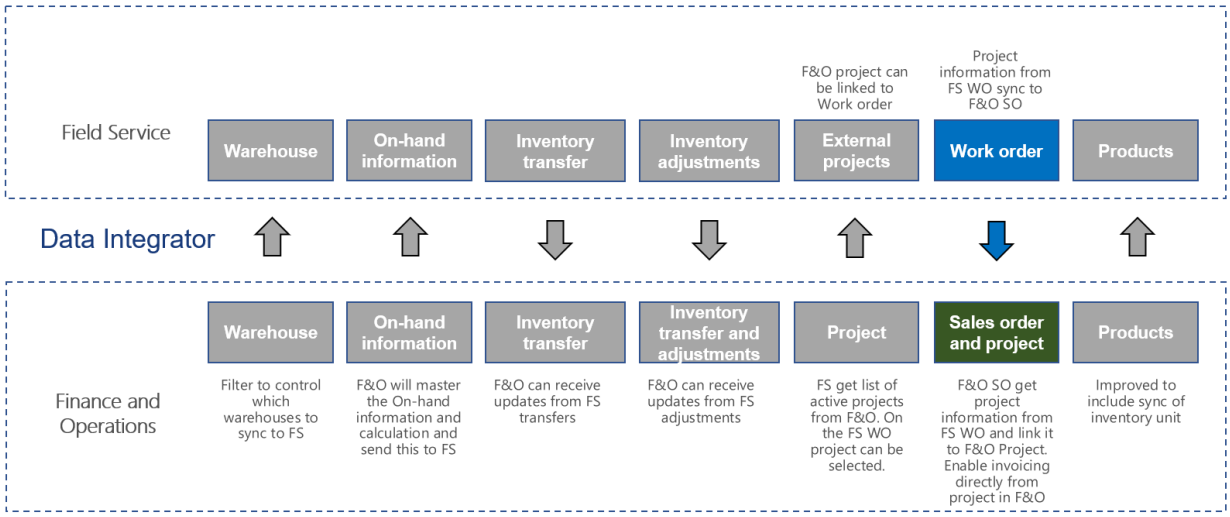
Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Synchronize work orders with project from Field Service to Supply Chain Management

2/18/2021 • 2 minutes to read • Edit Online

This topic discusses the templates and underlying task that are used to synchronize work orders with a project number from Dynamics 365 Field Service to Dynamics 365 Supply Chain Management.



The used **Work Orders with Project (Field Service to Supply Chain Management)** template is based on the **Work Orders (Field Service to Supply Chain Management)** template. For more information, see Synchronize work orders in Field Service to sales orders in Supply Chain Management.

This topic only describes the differences between the two templates:

- **Work Orders with Project (Field Service to Supply Chain Management)**
- **Work Orders (Field Service to Supply Chain Management)**

The main difference is that this template includes mapping of the project number assigned to the Work order in Field Service, ensuring that the Sales order created in Supply Chain Management include the project number and that invoicing can happen on the related project. Besides this the template use Advanced Query and Filtering.

Templates and tasks

Name of the template in Data integration:

- Work Orders with Project (Field Service to Supply Chain Management)

Name of the task in the Data integration project:

- WorkOrderHeader
- WorkOrderHeaderProject
- WorkOrderProduct
- WorkOrderService

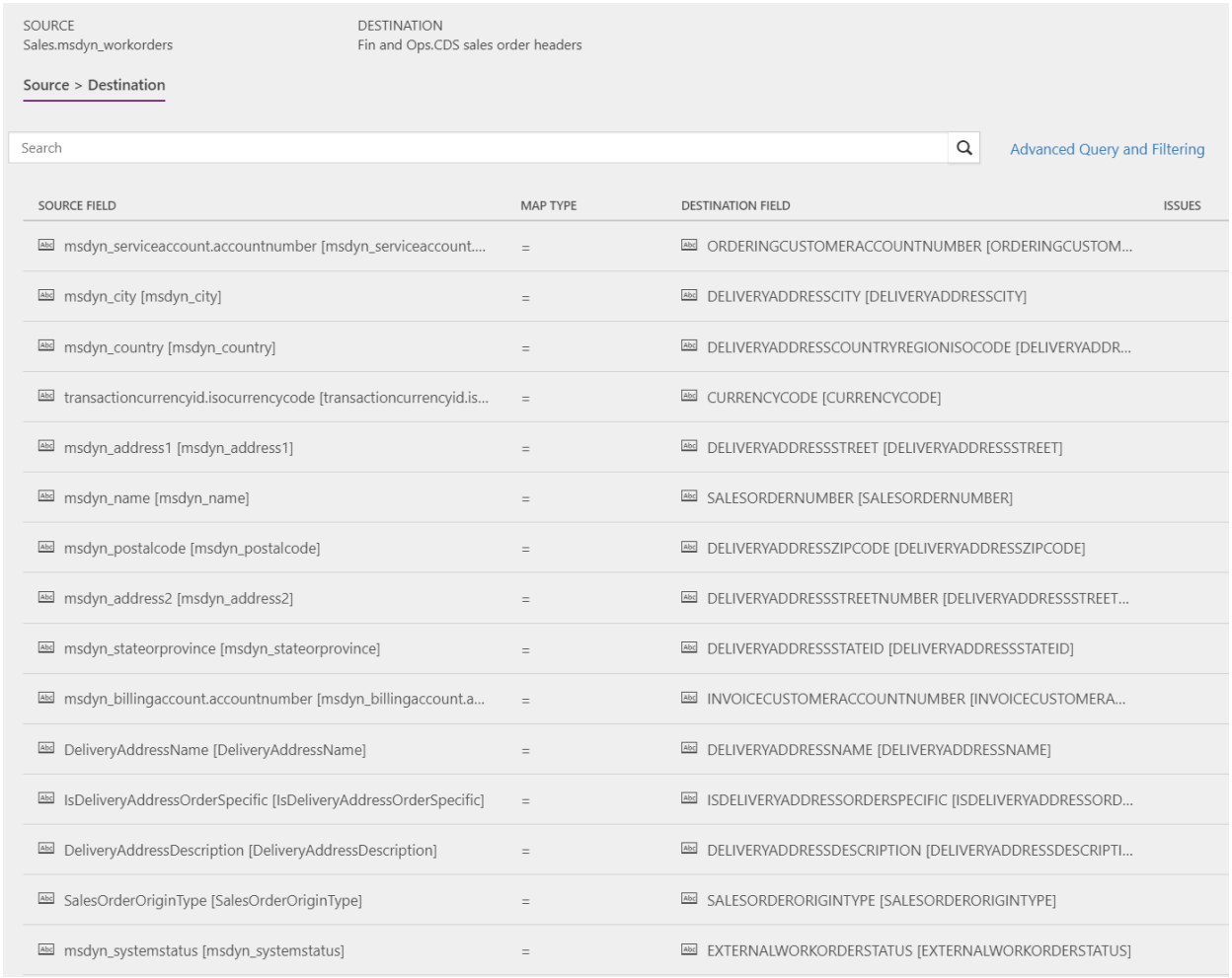
Field Service CRM solution

The **External Project** field has been added to the Work Order entity. This field is a lookup and buy tagging your Work Order with a project the Sales Order will then be connected to a Project within Supply Chain Management. When the **System Status** changes from Open – In Progress(690,970,000) to a higher status, the **External Project** field will be locked and you can't add, remove, or change the value.

Template mapping in Data integration

The following illustrations show the template mapping in Data integration.

Work Orders with Project (Field Service to Supply Chain Management): WorkOrderHeader



SOURCE FIELD	MAP TYPE	DESTINATION FIELD	ISSUES
msdyn_serviceaccount.accountnumber [msdyn_serviceaccount....	=	ORDERINGCUSTOMERACCOUNTNUMBER [ORDERINGCUSTOM...	
msdyn_city [msdyn_city]	=	DELIVERYADDRESSCITY [DELIVERYADDRESSCITY]	
msdyn_country [msdyn_country]	=	DELIVERYADDRESSCOUNTRYREGIONISOCODE [DELIVERYADDR...	
transactioncurrencyid.isocurrencycode [transactioncurrencyid.is...	=	CURRENCYCODE [CURRENCYCODE]	
msdyn_address1 [msdyn_address1]	=	DELIVERYADDRESSSTREET [DELIVERYADDRESSSTREET]	
msdyn_name [msdyn_name]	=	SALESORDERNUMBER [SALESORDERNUMBER]	
msdyn_postalcode [msdyn_postalcode]	=	DELIVERYADDRESSZIPCODE [DELIVERYADDRESSZIPCODE]	
msdyn_address2 [msdyn_address2]	=	DELIVERYADDRESSSTREETNUMBER [DELIVERYADDRESSSTREET...	
msdyn_stateorprovince [msdyn_stateorprovince]	=	DELIVERYADDRESSSTATEID [DELIVERYADDRESSSTATEID]	
msdyn_billingaccount.accountnumber [msdyn_billingaccount.a...	=	INVOICECUSTOMERACCOUNTNUMBER [INVOICECUSTOMERA...	
DeliveryAddressName [DeliveryAddressName]	=	DELIVERYADDRESSNAME [DELIVERYADDRESSNAME]	
IsDeliveryAddressOrderSpecific [IsDeliveryAddressOrderSpecific]	=	ISDELIVERYADDRESSORDERSPECIFIC [ISDELIVERYADDRESSORD...	
DeliveryAddressDescription [DeliveryAddressDescription]	=	DELIVERYADDRESSDESCRIPTION [DELIVERYADDRESSDESCRIPTI...	
SalesOrderOriginType [SalesOrderOriginType]	=	SALESORDERORIGINTYPE [SALESORDERORIGINTYPE]	
msdyn_systemstatus [msdyn_systemstatus]	=	EXTERNALWORKORDERSTATUS [EXTERNALWORKORDERSTATUS]	

Work Orders with Project (Field Service to Supply Chain Management): WorkOrderHeaderProject

SOURCE
Sales.msdyn_workorders

DESTINATION
Fin and Ops.CDS sales order headers

Source > Destination

Search

Advanced Query and Filtering

SOURCE FIELD	MAP TYPE	DESTINATION FIELD	ISSUES
msdyn_city [msdyn_city]	=	DELIVERYADDRESSCITY [DELIVERYADDRESSCITY]	
msdyn_country [msdyn_country]	=	DELIVERYADDRESSCOUNTRYREGIONISOCODE [DELIVERYADDR...	
msdyn_address1 [msdyn_address1]	=	DELIVERYADDRESSSTREET [DELIVERYADDRESSSTREET]	
msdyn_name [msdyn_name]	=	SALESORDERNUMBER [SALESORDERNUMBER]	
msdyn_postalcode [msdyn_postalcode]	=	DELIVERYADDRESSZIPCODE [DELIVERYADDRESSZIPCODE]	
msdyn_address2 [msdyn_address2]	=	DELIVERYADDRESSSTREETNUMBER [DELIVERYADDRESSSTREET...	
msdyn_stateorprovince [msdyn_stateorprovince]	=	DELIVERYADDRESSSTATEID [DELIVERYADDRESSSTATEID]	
DeliveryAddressName [DeliveryAddressName]	=	DELIVERYADDRESSNAME [DELIVERYADDRESSNAME]	
IsDeliveryAddressOrderSpecific [IsDeliveryAddressOrderSpecific]	=	ISDELIVERYADDRESSORDERSPECIFIC [ISDELIVERYADDRESSORD...	
DeliveryAddressDescription [DeliveryAddressDescription]	=	DELIVERYADDRESSDESCRIPTION [DELIVERYADDRESSDESCRIPTI...	
SalesOrderOriginType [SalesOrderOriginType]	=	SALESORDERORIGINTYPE [SALESORDERORIGINTYPE]	
msdyn_systemstatus [msdyn_systemstatus]	=	EXTERNALWORKORDERSTATUS [EXTERNALWORKORDERSTATUS]	
msdynce_externalproject.msdynce_projectnumber [msdynce_ex...	=	PROJECTID [PROJECTID]	

Work Orders with Project (Field Service to Supply Chain Management): WorkOrderProduct

SOURCE
Sales.msdyn_workorderproducts

DESTINATION
Fin and Ops.CDS sales order lines

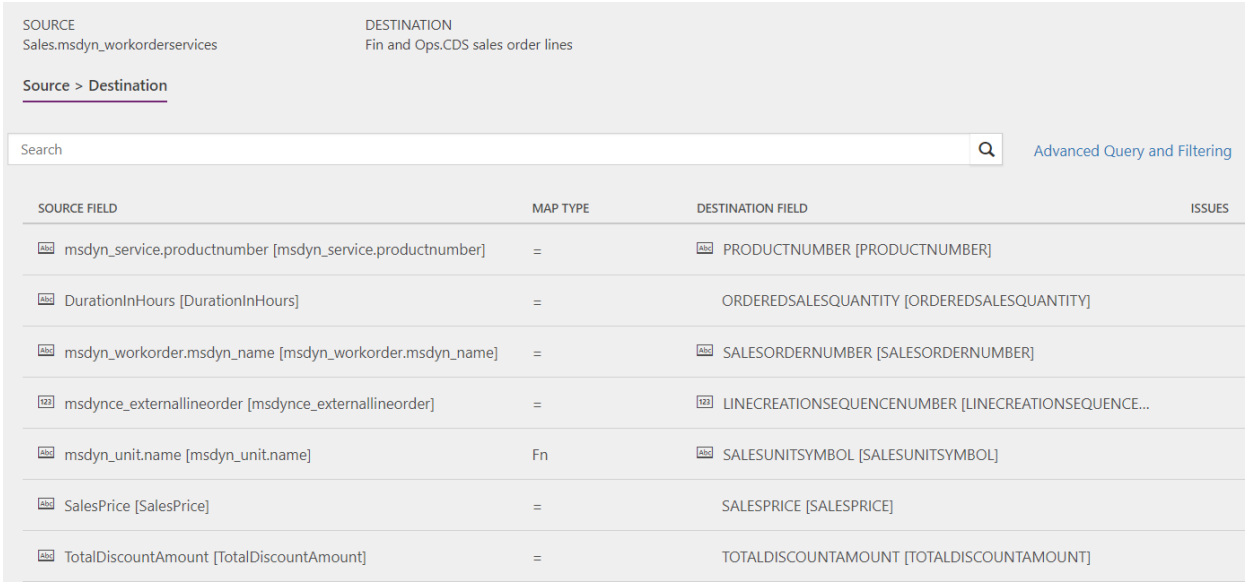
Source > Destination

Search

Advanced Query and Filtering

SOURCE FIELD	MAP TYPE	DESTINATION FIELD	ISSUES
msdyn_product.productnumber [msdyn_product.productnumber]	=	PRODUCTNUMBER [PRODUCTNUMBER]	
OrderedSalesQuantity [OrderedSalesQuantity]	=	ORDEREDSALESQUANTITY [ORDEREDSALESQUANTITY]	
msdyn_workorder.msdyn_name [msdyn_workorder.msdyn_name]	=	SALESORDERNUMBER [SALESORDERNUMBER]	
msdyn_lineorder [msdyn_lineorder]	=	LINECREATIONSEQUENCENUMBER [LINECREATIONSEQUENCE...	
msdyn_unit.name [msdyn_unit.name]	Fn	SALESUNITSYMBOL [SALESUNITSYMBOL]	
SalesPrice [SalesPrice]	=	SALESPRICE [SALESPRICE]	
TotalDiscountAmount [TotalDiscountAmount]	=	TOTALDISCOUNTAMOUNT [TOTALDISCOUNTAMOUNT]	
PickedQuantity [PickedQuantity]	=	PickedQuantity [PickedQuantity]	
WarehouseName [WarehouseName]	=	SHIPPINGWAREHOUSEID [SHIPPINGWAREHOUSEID]	

Work Orders with Project (Field Service to Supply Chain Management): WorkOrderService



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Consume external web services

2/18/2021 • 2 minutes to read • Edit Online

You can consume web services by adding new class libraries. In Microsoft Dynamics AX 2012, you could consume web services from X++ code by adding Microsoft Visual Studio projects as a reference and by using **Aif::CreateServiceClient**. This scenario is supported, but the steps have changed. Application Integration Framework (AIF) is no longer supported.

The following steps show how to consume an external StockQuote service from X++.

Note that the web service URL in this sample is fictional. There is no known web service at http://www.contoso.net/stockquote.asmx. To make this code work you will need to adapt it to your specific web service.

1. Create a new Class Library project in Visual Studio, and name it **ExternalServiceLibrary.csproj**.
2. In the Visual Studio project, add a service reference to the external web service: `http://www.contoso.net/stockquote.asmx`.
3. Create a new static class, and wrap the StockQuote service operation as shown in the following example.

```
public static string GetQuote(string s)
{
    var binding = new System.ServiceModel.BasicHttpBinding();
    var endpointAddress = new EndpointAddress("http://www.contoso.net/stockquote.asmx");
    ServiceLibrary.QuoteReference.StockQuoteSoapClient client = new
ServiceLibrary.QuoteReference.StockQuoteSoapClient(binding, endpointAddress);

    //GetQuote is the operation on the StockQuote service
    return client.GetQuote("MSFT");
}
```

4. Build the project. The binary ExternalServiceLibrary.dll is created.
5. Create a new Dynamics project in Visual Studio.
6. Add **ExternalServiceLibrary.dll** as a reference.
7. In the X++ class, you can use the external web services that were referenced in ExternalServiceLibrary.dll.

```
public static void main(Args _args)
{
    info(ServiceLibrary.StockQuoteClass::GetQuote("MSFT"));
}
```

Electronic messaging

2/18/2021 • 32 minutes to read • Edit Online

This topic provides overview and setup information for electronic messaging.

Recently, the governments and legislative authorities of various countries and regions around the world have implemented reporting requirements for companies that are registered in those countries or regions. The purpose of the requirements is to enable data to be obtained from those companies in electronic format, directly from the systems where it was accounted, stored, and processed.

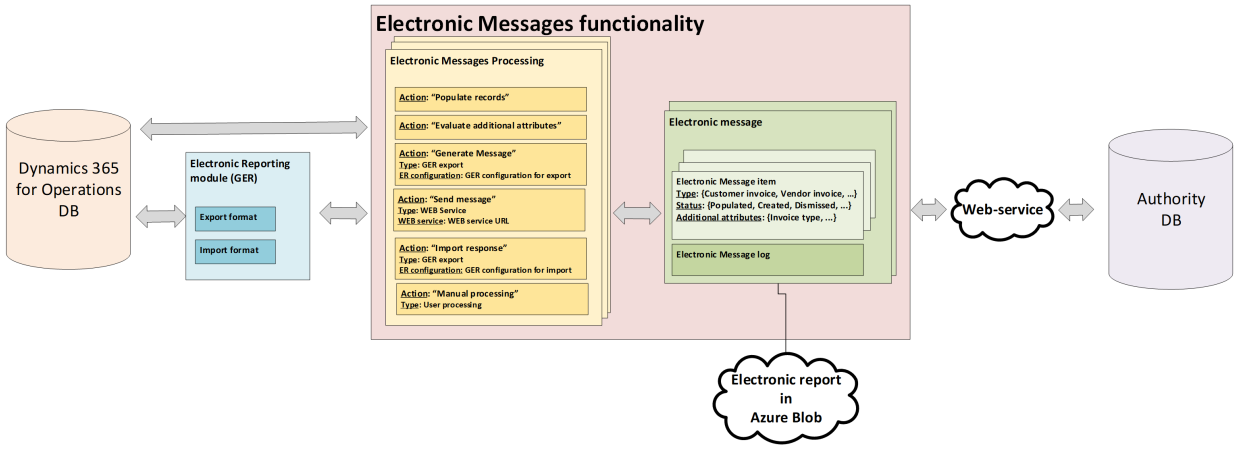
The Electronic messages functionality in Finance supports various processes for electronic interoperation between Finance and the systems that governments and legislative authorities offer for reporting, submitting, and receiving official information.

The Electronic messages functionality is integrated with the **Electronic Reporting** (ER) module. Therefore, you can set up ER formats for electronic messages. For more information, see Electronic reporting (ER).

Electronic messaging is based on the following entities:

- **Electronic message** – A report or declaration that should be reported and/or transmitted internally. An example is a report that is sent to a tax office.
- **Electronic message items** – Records that should be included in the message that is reported.
- **Electronic message processing** – A chain of actions that should be run to collect the required data, generate reports, store data in Microsoft Azure Blob storage, transmit reports outside the system, receive responses from outside the system, and, based on the information that is received, update the database. The actions in the chain can be either linked or unlinked

The following illustration shows the flow of data for electronic messaging.



The Electronic messages functionality supports the following scenarios:

- Manually create messages, and generate reports that are based on associated exporting ER formats of various types: Microsoft Excel, XML, JavaScript Object Notation (JSON), PDF, text, and Microsoft Word.
- Automatically create and process messages that are based on information that was requested and received from an authority via an associated importing ER format.
- Collect and process information from a data source as message items. The data source is a Finance table.
- Store additional information, and evaluate various values by calling specifically defined executable classes in relation to messages or message items.
- Aggregate information that is collected in message items, split that information by message, and generate

reports that are in associated exporting ER formats.

- Transmit the reports that are generated to a web service by using security information that is stored in Azure Key Vault.
- Receive a response from a web service, interpret the response, and update data in Finance as appropriate.
- Store and review all the reports that are generated.
- Store and review all the log information that is related to actions that are run for a message or message item.
- Control the processing through various message statuses and message item statuses.

Set up electronic messaging

Electronic messaging can help you maintain different electronic reporting processes for different document types. In some complex scenarios, electronic messaging is set up so that it has a combination of many message statuses, message items statuses, actions, additional fields, and executable classes. For these scenarios, packages of data entities are available for import. If you use these data entity packages, you should import them to a legal entity by using the Data management tool. For more information about how to use the Data management tool, see Data management.

If you don't import a data entity package, you can manually set up the Electronic messages functionality. In this case, you must set up the following elements:

- Number sequences
- Message item types and statuses
- Message statuses
- Additional fields
- Executable class settings
- Populate records actions
- Web applications
- Web service settings
- Message processing actions
- Electronic message processing

The following sections provide more information about each of these elements.

Number sequences

Set up number sequences for both messages and message items. The number sequences are used to automatically number the messages and the message items. The numbers that are assigned will be used as unique identifiers for the messages and message items in the system. You can set up number sequences for electronic messaging on the **General ledger parameters** page (**General ledger** > **Ledger setup** > **General ledger parameters**).

Message item types and statuses

Message item types identify the types of records that will be used in electronic messages. You can set up message item types on the **Message item types** page (**Tax** > **Setup** > **Electronic messages** > **Message item types**).

Message item statuses identify the statuses that will apply to message items in the processing that you're setting up. You can set up message item types on the **Message item statuses** page (**Tax** > **Setup** > **Electronic messages** > **Message item statuses**).

The **Allow delete** parameter of a message item status defines whether users can delete message items that have this status on the **Electronic messages** page or the **Electronic message items** page.

Message statuses

Set up the message statuses that should be available in message processing. You can set up message statuses on the **Message statuses** page (**Tax** > **Setup** > **Electronic messages** > **Message statuses**).

The following table describes the fields on the **Message statuses** page.

FIELD NAME	DESCRIPTION
Message status	Enter a unique name for the message status. Message statuses are used to characterize the state of an electronic message at every moment. The name that you enter is shown on the **Electronic messages** page and in a log that is related to electronic messages.
Description	Enter a description of the message status.
Response type	Select the type of response for the message status. Some actions in a processing can produce more than one response type. For example, actions of the **Web service** type can produce responses of either the **Successfully executed** type or the **Technical error** type, depending on the result of its execution. In this case, you must define message statuses for both response types. For more information about action types and the types of responses that are related to them, see Message processing action types.
Message item status	Sometimes, the status of an electronic message must influence the status of related message items. Select a message item status in this field to associate it with the message status.
Allow delete	Select this check box if users should be able to delete electronic messages that have this status on the **Electronic messages** page.

Additional fields

The Electronic messages functionality lets you fill in records from a transactional table. In this way, you can prepare the records for reporting and then report them. However, transactional tables sometimes don't have enough information to fill in records in a manner that meets the reporting requirements. To fill in all the information that must be reported for a record, you can set up additional fields. Additional fields can be associated with both messages and message items. You can set up additional fields on the **Additional fields** page (**Tax** > **Setup** > **Electronic messages** > **Additional fields**).

The following table describes the general fields on the **Additional fields** page.

FIELD	DESCRIPTION
Field name	Enter the name of an additional attribute of message items that are related to the process. This name is shown in the user interface (UI) while you work with the process. It can also be used in ER configurations that are related to the process.
Description	Enter a description of the additional field.
User edit	Set this option to **Yes** if users should be able to change the value of the additional field from the UI.

FIELD	DESCRIPTION
Counter	Set this option to **Yes** if the additional field should contain a number sequence in an electronic message. Value of the additional field will be filled in automatically when an action of the **Electronic reporting export** is run.
Hidden	Set this option to **Yes** if the additional field should be hidden in the UI.

Each additional field can have different values for the processing. You define these values on **Values** FastTab. The following table describes the fields.

FIELD	DESCRIPTION
Field value	Enter the field value to use for a message or message item during reporting.
Description	Enter a description of the field value.
Account type	Some field values might be limited to specific account types. Select one of the following values: **All**, **Customer**, or **Vendor**.
Account code	If you selected **Customer** or **Vendor** in the **Account type** field, you can further limit the use of the field value to a specific group or table.
Account/Group number	If you selected **Customer** or **Vendor** in the **Account type** field, and if you entered a group or table in the **Account code** field, you can enter a specific group or counteragent in this field.
Effective	Specify the date when the value should start to be considered.
Expiration	Specify the date when the value should stop being considered.

By default, combinations of criteria that are defined by the **Account/Group number**, **Account code**, **Effective**, and **Expiration** fields don't influence the selection of values for additional fields. However, these combinations can be used in an executable class to implement specific logic that calculates values for additional fields.

Executable class settings

An executable class is an X++ method or class that the electronic message processing can call in relation to an action if some evaluation is required for the process.

You can manually set up an executable class on the **Executable class settings** page (**Tax** > **Setup** > **Electronic messages** > **Executable class settings**). Create a line, and set the following fields.

FIELD	DESCRIPTION
Executable class	Enter the name that will be used during the setup of a message processing action that this class is called in relation to.

FIELD	DESCRIPTION
Description	Enter a description of the executable class.
Executable class name	Select an X++ executable class.
Execution level	This field is set automatically, because the value should be predefined for the selected executable class. This field limits the level that related evaluation is run on.
Class description	This field is set automatically, because the value should be predefined for the selected executable class.

Some executable classes might have mandatory parameters that must be defined before the executable class is run for the first time. To define these parameters, select **Parameters** on the Action Pane, set the fields in the dialog box that appears, and then select **OK**. It's important that you select **OK**. Otherwise, the parameters won't be saved to the database, and the executable class won't be called correctly.

Populate records actions

You use populate records actions to set up actions that add records to the Message items table so that they can be added to an electronic message. For example, if your electronic message must report customer invoices, you must set up a populate records action that is linked to the **Data source** field in the Customer invoice journal table. You can set up populate records actions on the **Populate records action** page (**Tax** > **Setup** > **Electronic messages** > **Populate records actions**). Create a new record for every action that should add records to the table, and set the following fields.

FIELD	DESCRIPTION
Name	Enter a name for the action that fills in records in your process.
Description	Enter a description of the populate records action.

On the **Datasources setup** FastTab, add a line for every data source that is used for the process, and set the following fields.

FIELD	DESCRIPTION
Name	Enter a name for the data source.
Message item type	Select the type of message item that should be used when records are created for the data source.
Account type	Select the type of account that should be associated with records from the data source.
Master table name	Select the table that should be a data source.
Document number field	Select the field that the document number should be taken from in the selected table.
Document date field	Select the field that the document date should be taken from in the selected table.

FIELD	DESCRIPTION
Document account field	Select the field that the document account should be taken from in the selected table.
User query	If this check box is selected, you can set up a query by selecting **Edit query** above the grid. Otherwise, all the records will be filled in from the selected data source.

Web applications

You use web application settings to set up a web application so that it supports Open Authorization (OAuth) 2.0. OAuth is the open standard that lets users grant "secure delegated access" to the application on their behalf, without sharing their access credentials. You can also go through the authorization process by getting an authorization code and access token. You can set up web application settings on the **Web applications** page (**Tax** > **Setup** > **Electronic messages** > **Web applications**).

The following table describes the fields on the **Web applications** page.

FIELD	DESCRIPTION
Application name	Enter a name for the web application.
Description	Enter a description of the web application.
Base URL	Enter the base internet address of the web application.
Authorization URL path	Specify the path that is used to compose the URL for authorization.
Token URL path	Specify the path that is used to compose the URL for the token.
Redirect URL	Enter the redirect URL.
Client ID	Enter the client ID of the web application.
Client secret	Enter the client secret of the web application.
Server token	Enter the server token of the web application.
Authorization format mapping	Select the ER format that is used to generate the request for authorization.
Import token model mapping	Select the ER importing model mapping that is used to store the access token.
Granted scope	The scope that is granted for requests to the application. This field is automatically updated.
Access token will expire in	The remaining time before the access token expires.
Accept	Specify the **Accept** property of the web request. For example, enter **application/vnd.hmrc.1.0+json**.

FIELD	DESCRIPTION
Content type	Specify the content type. For example, enter **application/json**.

In addition, the following buttons are available on the Action Pane of the **Web applications** page to support the authorization process:

- **Get authorization code** – Initialize authorization of the web application.
- **Obtain access token** – Initialize the process of getting an access token.
- **Refresh access token** – Refresh an access token.

When an access token to a web application is stored in the system's database in encrypted format, it can be used for requests to a web service. For security purposes, access to the access token must be limited to security roles that must be allowed to address those requests. If users outside the security group try to address a request, they receive an error that states that they aren't allowed to interoperate via the selected web application. To set up the security roles that must have access to the access token, use the **Security roles** FastTab on the **Web applications** page. If security roles aren't defined for a web application, only a system admin can interoperate via this web application.

Web service settings

You use web service settings to set up direct data transmission to a web service. You can set up web service settings on the **Web service settings** page (**Tax** > **Setup** > **Electronic messages** > **Web service settings**).

The following table describes the fields on the **Web service settings** page.

FIELD	DESCRIPTION
Web service	Enter a name for the web service.
Description	Enter a description of the web service.
Internet address	Enter the internet address of the web service. If a web application is specified for the web service, and if the internet address of the web service should be the same as the internet address that is defined for that web application, select **Copy base URL** to copy the base URL of the web application to this field.
Certificate	Select a Key Vault certificate that has previously been set up.
Web application	Select a Key Vault certificate that has previously been set up.
The response type – XML	Set this option to **Yes** if the response type is XML.
Request method	Specify the method of the request. HTTP defines a set of request methods that indicate the action that should be performed for a given resource. The request method can be **GET**, **POST**, or another HTTP method.
Request headers	Specify request headers. A request header is an HTTP header that can be used in an HTTP request, and that isn't related to the content of the message.
Accept	Specify the **Accept** property of the web request.

FIELD	DESCRIPTION
Accept encoding	Specify the **Accept-Encoding** value. The Accept-Encoding request HTTP header advertises the content encoding that the client can understand. This content encoding is usually a compression algorithm.
Content type	Specify the content type. The Content-Type entity HTTP header indicates the media type of the resource.
Successful response code	Specify the HTTP status code that indicates that the request was successful.
Request headers format mapping	Select the ER format that is used to generate web request headers.

Message processing actions

You use message processing actions to create actions for your processes and set up their parameters. You can set up message processing actions on the **Message processing actions** page (**Tax** > **Setup** > **Electronic messages** > **Message processing actions**).

The following tables describe the fields on the **Message processing actions** page.

General FastTab

FIELD	DESCRIPTION
Action type	Select the type of action. For information about the available options, see the Message processing action types section.
Format mapping	Select the ER format that should be called for the action. This field is available only for actions of the **Electronic reporting export**, **Electronic reporting import**, and **Electronic reporting export message** types.
Format mapping for URL path	Select the ER format that should be called for the action. This field is available only for actions of the **Web service** type. It's used to compose the path of the URL address that will be added to the base internet address that is specified for the selected web server.
Message item type	Select the type of records that the action should be evaluated for. This field is available for actions of the **Message item execution level**, **Electronic reporting export**, **Electronic reporting import**, and **Web service** types, and also some other types. If you leave this field blank, all the message item types that are defined for the message processing are evaluated.
Executable class	Select executable class settings that were previously created. This field is available only for actions of the **Message item execution level** and **Message item execution level** types.
Populate records action	Select a populate records action that was previously set up. This field is available only for actions of the **Populate records** type.

FIELD	DESCRIPTION
Web service	Select a web service that was previously set up. This field is available only for actions of the **Web service** type.
File name	Specify the name of the file that will be the result of the action. This file can be the response from the web server or the report that is generated. This field is available only for actions of the **Web service** and **Electronic reporting export message** types.
Show dialog	Set this option to **Yes** if a dialog box must be shown to users before report generation. This field is available only for actions of the **Electronic reporting export message** type.

Message processing action types

The following options are available in the **Action type** field:

- **Create message** – Use this action type to let users manually create messages on the **Electronic message** page. An initial status can't be set up for an action of this type.
- **Populate records** – An action of the **Populate records** type must previously be set up. Associate this action type with a populate records action to enable that action to be included in processing. It's assumed that this action type is used either for the first action in message processing (when no electronic message was created in advance) or for an action that adds message items to a message that was previously created by an action of **Create message** type. Therefore, for actions of this type, a result status can be set up only for message items. An initial status can be set up only for messages.
- **Message execution level** – Use this action type to set up an executable class that should be evaluated at the message level.
- **Message item execution level** – Use this action type to set up an executable class that should be evaluated at the message item level.
- **Electronic reporting export** – Use this action type for actions that should generate a report that is based on an exporting ER configuration at the message item level.
- **Electronic reporting export message** – Use this action type for actions that should generate a report that is based on an exporting ER configuration at the message level (for example, when a message doesn't have any message items).
- **Electronic reporting import** – Use this action type for actions that should generate a report that is based on an importing ER configuration.
- **Message level user processing** – Use this action type for actions that assume some manual action by the user at the message level. For example, the user might update the status of messages.
- **User processing** – Use this action type for actions that assume some manual action by the user at the message item level. For example, the user might update the status of messages items.
- **Web service** – Use this action type for actions that should transmit a generated report to a web service. This action type isn't used for Italian Purchase and Sales Invoices Communication reporting. For actions of **Web service** type, the **Message processing actions** page includes a **Miscellaneous details** FastTab, where you can specify a confirmation text. This confirmation text will be shown to users before requests to the selected web service are addressed.
- **Request verification** – Use this action type to request verification from a server.

Initial statuses FastTab

> **NOTE**
>
> The **Initial statuses** FastTab isn't available for actions that have an initial action type of **Create message**.

FIELD	DESCRIPTION
Message item status	Select the message item status that the selected message processing action should be evaluated for.
Description	A description of the selected message item status.

Result statuses FastTab

FIELD	DESCRIPTION
Message status	Select the message statuses that the selected message processing action should be evaluated for. This field is available only for message processing actions that are evaluated at the message level. For example, it's available for actions of the **Electronic reporting export** and **Electronic reporting import** types, but it isn't available for actions of the **User processing** and **Message item execution level** types.
Description	A description of the selected message status.
Response type	The response type of the selected message status.
Message item status	Select the resulting statuses that should be available after the selected message processing action is evaluated. This field is available only for message processing actions that are evaluated at the message item level. For example, it's available for actions of the **User processing** and **Message item execution level** types. For message processing actions that are evaluated at the message level, this field shows the message item status that was set up for the selected message status.

The following table shows the result statuses that must be set up for different action types and response types.

ELECTRONIC MESSAGE ACTION TYPE/RESPONSE TYPE	SUCCESSFULLY EXECUTED	BUSINESS ERROR	TECHNICAL ERROR	USER DEFINED	CANCEL
Create message	X				
Electronic reporting export	X				
Electronic reporting import					
Web service	X		X		
User processing					

ELECTRONIC MESSAGE ACTION TYPE/RESPONSE TYPE	SUCCESSFULLY EXECUTED	BUSINESS ERROR	TECHNICAL ERROR	USER DEFINED	CANCEL
Message execution level					
Populate records					
Message item execution level					
Request verification	X	X	X		
Electronic reporting export message	X				
Message level user processing					

Electronic message processing

Electronic message processing is a basic concept of the Electronic messages functionality. It aggregates actions that should be evaluated for the electronic message. The actions can be linked via an initial status and a result status. Alternatively, actions of the **User processing** type can be started independently. On the **Electronic message processing** page (**Tax** > **Setup** > **Electronic messages** > **Electronic message processing**), you can also select additional fields that should be supported for the processing on either the message level or the message items level.

The **Action** FastTab lets you add predefined actions to the processing. You can specify whether an action must be run separately, or whether it can be started by the processing. To specify that an action in the processing can be initialized only by a user, set the **Run separately** field to **Yes** for that action. If an action should be started by the processing for messages or message items that are in the status that is defined as the initial status for the action, set the **Run separately** field to **No**. Actions of the **User action** type must always be run separately.

Sometimes, several actions must be aggregated into a sequence, even though the first action is set up so that it runs separately. For example, a user must initialize report generation, but immediately after the report is generated, it must be sent to a web service, and the response from the web service must be reflected in the system. In this situation, you can create an inseparable sequence for the actions that must always run together. On the **Action** FastTab, select **Inseparable sequences** above the grid, and create a sequence. Then, for all the actions that must run together, select the sequence in the **Inseparable sequence** field. In this case, the **Run separately** field can be set to **Yes** for the first action in the sequence but **No** for all the other action.

The **Message item additional fields** FastTab lets you add predefined additional fields that are related to message items. You must add additional fields for each type of message item that the fields are related to.

The **Message additional fields** FastTab lets you add predefined additional fields that are related to messages.

The **Security roles** FastTab lets you set up the security roles that are predefined in the system for specific processing. Users who have a specific role will see only processing that is defined for that role.

The **Batch** FastTab lets you set up processing to work in a batch regime.

Work with the Electronic messages functionality

If you're working at the message level, the **Electronic messages** page (**Tax** > **Inquiries and reports** > **Electronic messages** > **Electronic messages**) is more useful. If you're operating at the data collection (message item) level, the **Electronic message items** page (**Tax** > **Inquires and reports** > **Electronic messages** > **Electronic message items**) is more useful.

Electronic messages

The **Electronic messages** page presents the processing that is available to you, based on your role. Security roles are associated with processing in the setup of that processing. For each processing that is available to you, the page shows electronic messages and information that is related to them.

The **Messages** FastTab shows electronic messages for the selected processing. Depending on the status of the selected message and predefined processing, you can run some actions by using the buttons above the grid:

- **New** – This button is associated with actions of the **Create message** type.
- **Delete** – This button is available if the **Allow delete** check box is selected for the current status of the selected message.
- **Collect data** – This button is associated with actions of the **Populate records** type.
- **Generate report** – This button is associated with actions of the **Electronic reporting export message** type.
- **Send report** – This button is associated with actions of the **Web service** type.
- **Import response** – This button is associated with actions of the **Electronic reporting import** type.
- **Update status** – This button is associated with actions of the **Message level user processing** type.
- **Message items** – Open the **Electronic message items** page.

The **Action log** FastTab shows information about all the actions that have been run for the selected message. If an action caused an error, information about the error is attached to the related line in the grid. To review the information about the error, select the line in the grid, and then select the **Attachment** button (the paper clip symbol) in the upper-right corner on the page.

The **Message additional fields** FastTab shows all the additional fields that are defined for messages in the processing setup. It also shows the values of those additional fields.

The **Message items** FastTab shows all the message items that are related to the selected message. Depending on the status of the selected message item, you can run some actions by using the buttons above the grid:

- **Delete** – This button is available if the **Allow delete** check box is selected for the current status of the selected message item.
- **Update status** – This button is associated with actions of the **User processing** type.
- **Original document** – Open a page that shows the original document for the selected message item.

All reports that have already been generated and received for a message are attached to that message. To review the attachments that are related to a message, select the message, and then select the **Attachment** button (the paper clip symbol) in the upper-right corner of the page.

The **Attachments** page shows all the attachments that are related to the selected message. To view a file, select it in the list on the left, and then select **Open** on the Action Pane.

You can also review attachments that are related to a specific action that was previously run for a message. On

the **Electronic messages** page, select the message on the **Messages** FastTab, select the action on **Action log** FastTab, and then select the **Attachment** button in the upper-right corner of the page.

You can also run either the whole processing or just a specific action by selecting **Run processing** on the Action Pane.

Electronic message items

The **Electronic message items** page presents all message items and a log of the actions that have been run for each message item. It also shows the additional fields that are defined for the message items, and the values of those additional fields.

The following table describes the fields on the **Message items** tab.

FIELD	DESCRIPTION
Processing	The name of the processing that was used to create the message item.
Message item	The ID of the message item. This ID is assigned automatically, based on the **Message item** number sequence that is defined on the **General ledger parameters** page.
Message item date	The date when the message item was created.
Message item type	The type of message item. Several types of messages items can be set up for the same processing (for example, **Incoming invoices** and **Outgoing invoices**). This field can be filled in automatically only when an invoice is added to the Message items table.
Message item status	The actual status of the message item. The available statuses vary, depending on the type of message item. Here are some examples: • **Populated** – A record was added to the Message items table. • **Evaluated** – Additional attributes were calculated for the message item. • **Reported** – The message item was successfully added to a report. • **Excluded** – This status can be useful if you must exclude some message items from a report before it's exported.
Transmission date	For processing that automatically transmits a generated report outside the system, the date when the message item was transmitted.
Document number	This field is filled in automatically, based on the setup of the populate records action. This field can be filled in automatically only when an invoice is added to the Message items table.
Account number	The account number of a customer or vendor (or another field value, depending on the field that is defined on the populate records action). This field can be filled in automatically only when an invoice is added to the Message items table.

FIELD	DESCRIPTION
Message	The number of the message. This number is assigned automatically, based on the **Message** number sequence that is defined on the **General ledger parameters** page.
Message status	The actual status of the electronic message.
Next action	The next actions that can be started for the current status of the message item.

The **Additional fields** tab shows the additional fields for the selected message item, and their values.

Run processing

Select **Run processing** on the Action Pane to run the processing for message items. To run a specific action, in the **Run processing** dialog box, set the **Choose action** option to **Yes**, and then select an action. To run the whole processing, leave the **Choose action** option set to **No**.

Generate report

Select **Generate report** on the Action Pane to generate a report. This button is associated with actions of the **Electronic reporting export** type.

Update status

Select **Update status** on the Action Pane to update the status of one or more message items. In the **Update status** dialog box, use the **Records to include** FastTab to select the message items to update. Make sure that you correctly define the selection criteria, because message item statuses will be updated according to these criteria, the initial status of the selected action, and the **New status** value that you specify. After a status update is completed, it will be difficult to determine which items were updated. Therefore, it will be difficult to roll back the status update.

Electronic messages

Select **Electronic messages** on the Action Pane to review an electronic message that is related to the selected message item.

You can also review all the files that are related to a specific message item. Select the **Message** field for the message item, or select **Electronic messages** on the Action Pane. Then, on the **Electronic message** page, select the message to review files for, and then select the **Attachment** button (the paper clip symbol) in the upper-right corner of the page.

The **Attachments** page shows all the attachments that are related to the message. To view a file, select it in the list on the left, and then select **Open** on the Action Pane.

Original document

Select **Original document** on the Action Pane to open the original document for the selected message item.

Example: Set up and run processing to call a simple ER exporting format to generate an Excel report

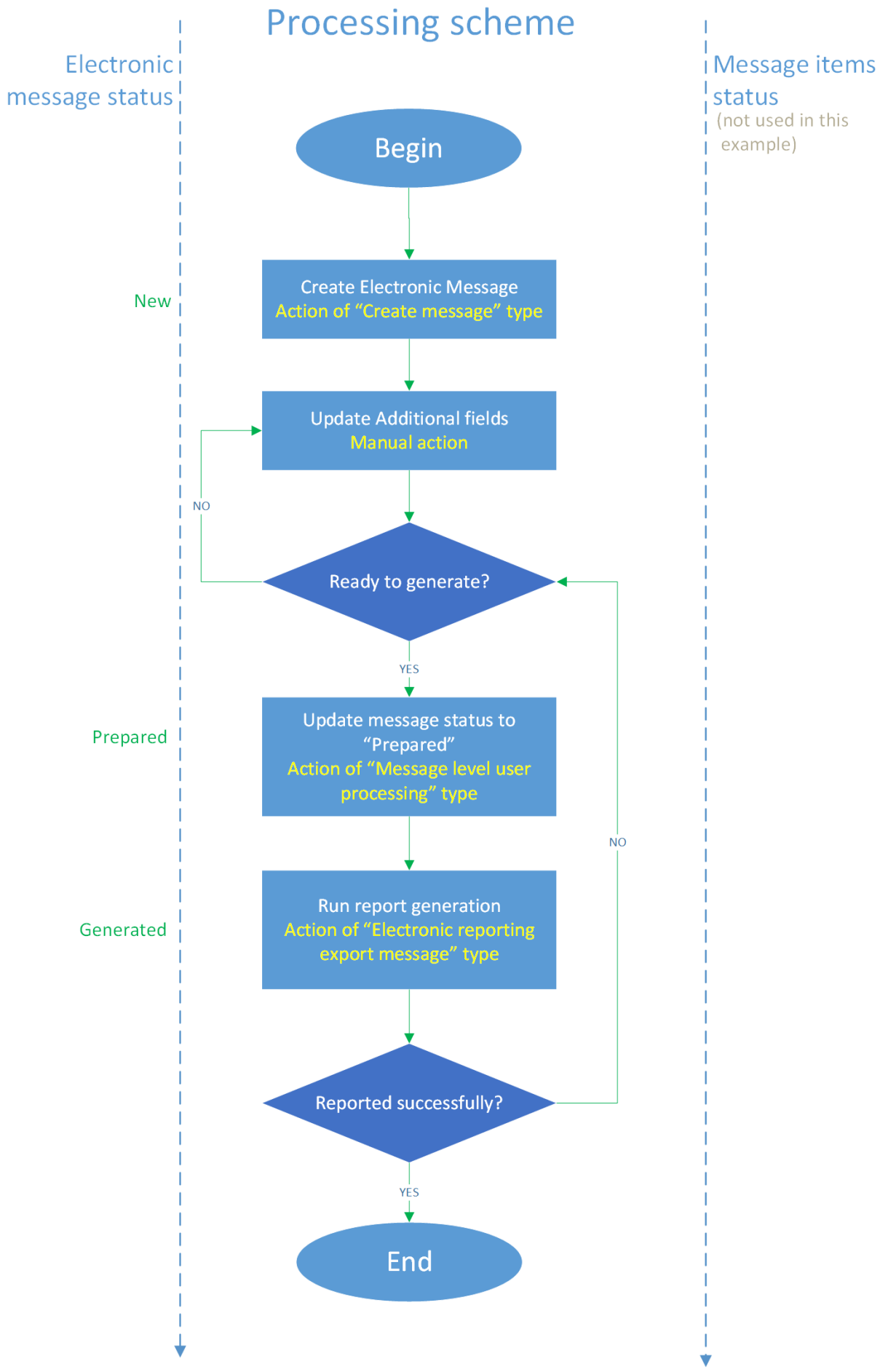
After you've created your ER format, mapped it to data sources, and completed it, you can run it by using the **Electronic reporting** workspace. A report is generated, and you can save it locally.

To control the following aspects of the reporting process, you must set up electronic messaging processing:

- Log information about who generated the report.
- Log information about when the report was generated.
- Save the reports that were generated for previous periods.

This section provides an example that shows how you can set up electronic messaging to generate a report that is based on an exporting ER format for Excel. If you want to follow this example, the ER Excel exporting format must already be created, mapped to data sources, and completed. Additionally, a number sequence must already be set up for electronic messages.

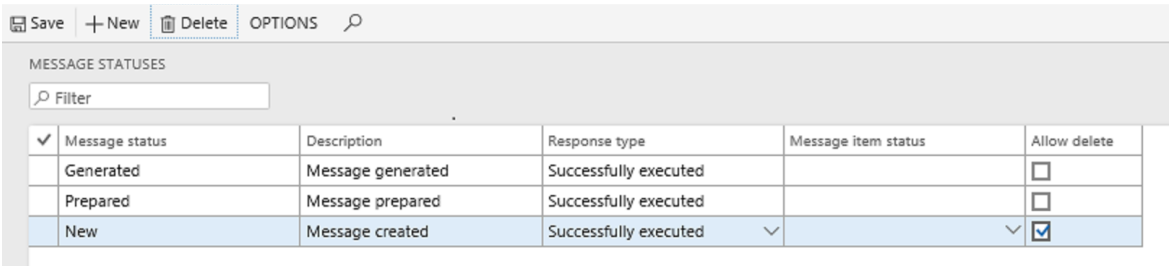
When you build processing, it's helpful if you first define the processing actions and statuses that will be set up. The following illustration shows what the processing looks like for this example.



Create message statuses

1. Go to **Tax** > **Setup** > **Electronic messages** > **Message statuses**.
2. Create the following message statuses:

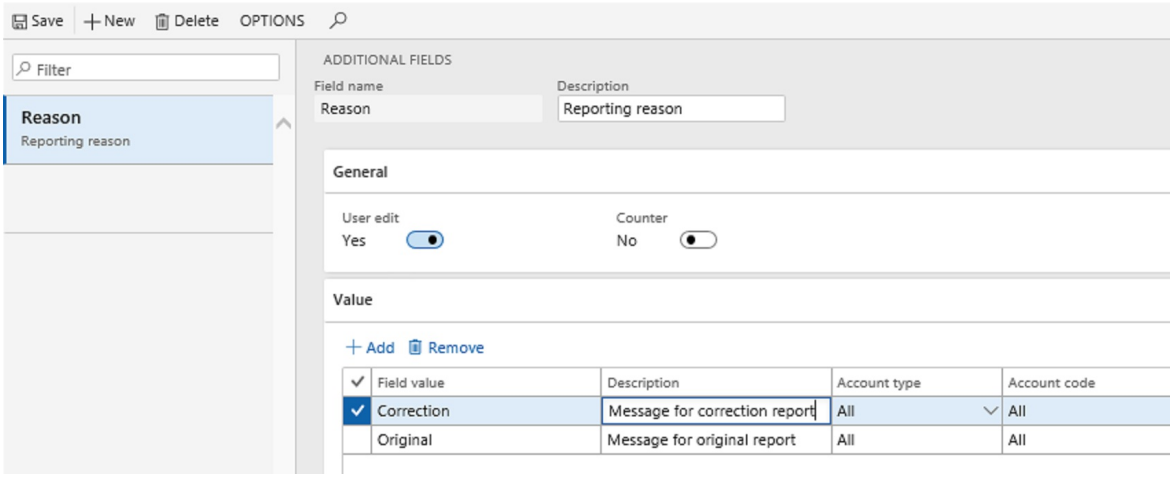
- New
- Prepared
- Generated



3. On the line for the **New** status, select the **Allow delete** check box to let users delete messages that have this status.

Create additional fields

1. Go to **Tax > Setup > Electronic messages > Additional fields**.
2. Add an additional field and its values. Here is an example.



3. Set the **User edit** option to **Yes** to let users edit the field.

Create message processing actions

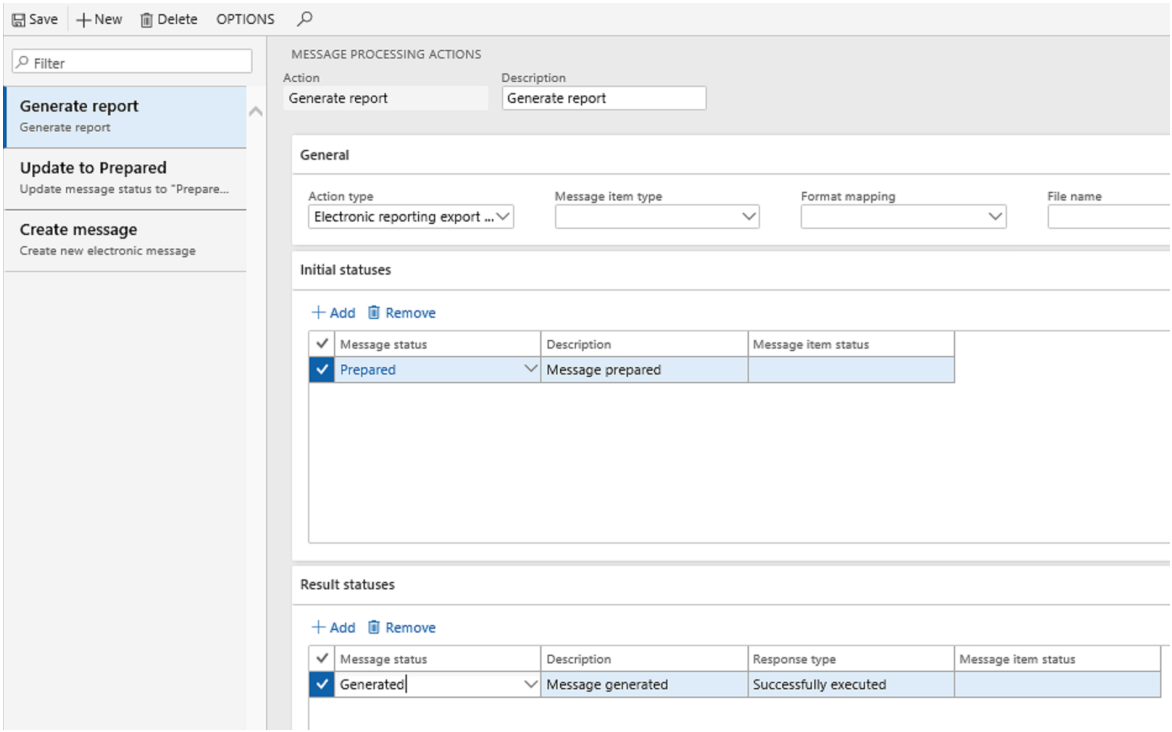
For this example, you will create the following actions:

- **Create message**
- **Update to Prepared**
- **Generate report**
- **Update to initial status** (optional)

1. Go to **Tax > Setup > Electronic messages > Message processing actions**.
2. Create an action that is named **Create message**. On the **General** FastTab, in the **Action type** field, select **Create message**.
3. Create an action that is named **Update to Prepared**, and set the following fields:
   - On the **General** FastTab, in the **Action type** field, select **Message level user processing**.
   - On the **Initial statuses** FastTab, in the **Message status** field, select **New**.
   - On the **Result statuses** FastTab, in the **Message status** field, select **Prepared**. In the **Response type** field, enter **Successfully executed**.
4. Create an action that is named **Generate report**, and set the following fields:
   - On the **General** FastTab, in the **Action type** field, select **Electronic reporting export**. In the

Format mapping field, select the exporting ER format. The options are **Excel**, **XML**, **JSON**, **Text**, and **Other**.

- On the **Initial statuses** FastTab, in the **Message status** field, select **Prepared**.
- On the **Result statuses** FastTab, in the **Message status** field, select **Generated**. In the **Response type** field, enter **Successfully executed**.



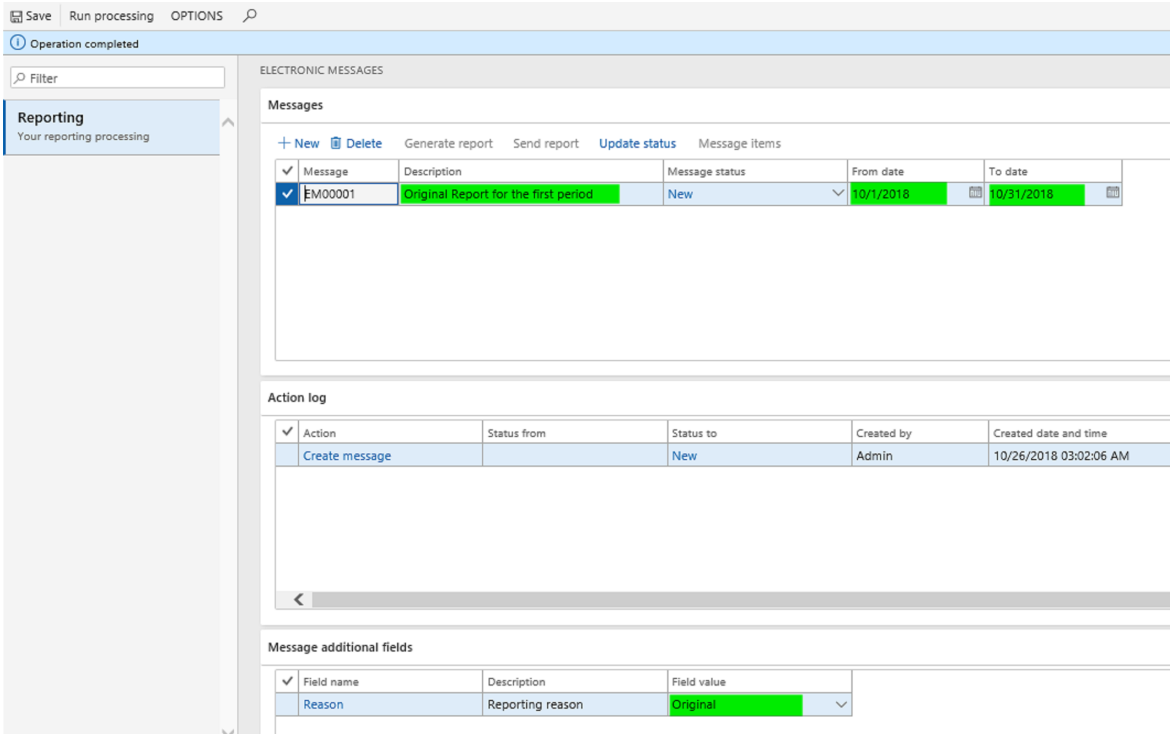
5. Optional: To let users regenerate a report several times, you can create an **Update to initial status** action and set the following fields:

   - On the **General** FastTab, in the **Action type** field, select **Message level user processing**.
   - On the **Initial statuses** FastTab, in the **Message status** field, select **Generated**.
   - On the **Result statuses** FastTab, add a separate line for each of the two message statuses (**Prepared** and **New**). For both lines, set the **Response type** field to **Successfully executed**.

Electronic message processing

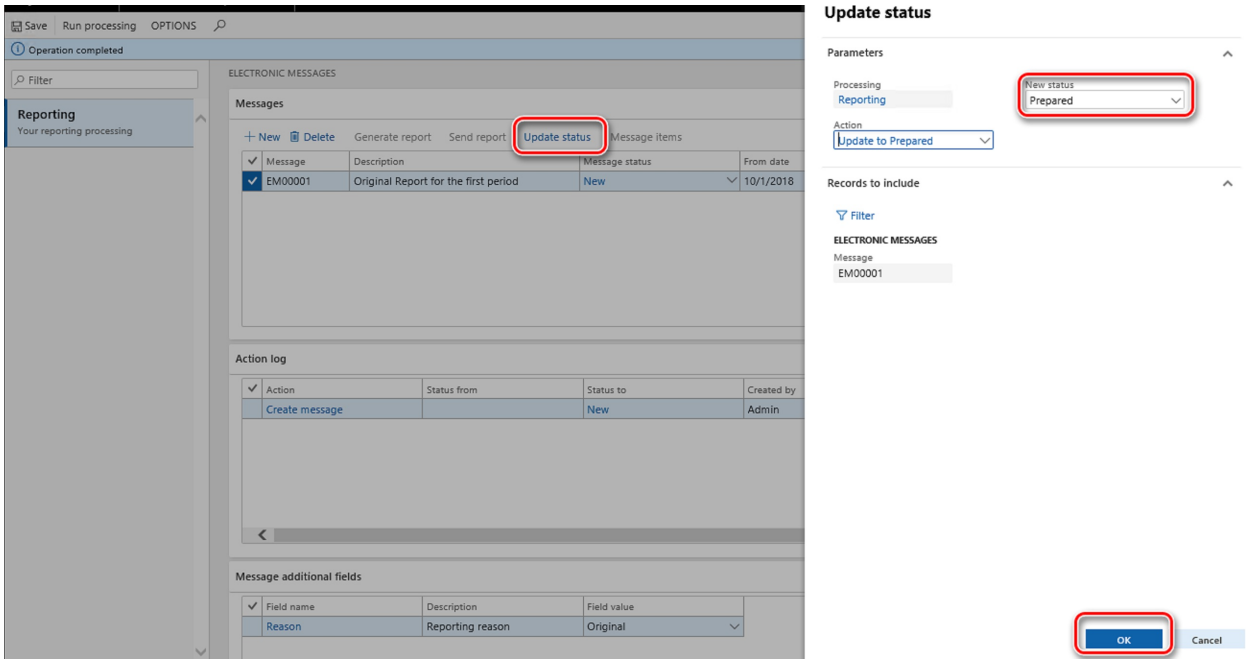
In this example, all the actions should be set up so that they run separately. The assumption is that the user will initialize every action.

1. Go to **Tax > Setup > Electronic messages > Electronic message processing**.
2. Add a record for your processing, and add all previously defined actions and an additional field.
3. Optional: On the **Security roles** FastTab, define security roles for your processing to limit access to specific reporting.
4. Go to **Tax > Inquires and reports > Electronic messages > Electronic messages**.
5. Select **New** to create a message. At this point, you can add dates and a description. You can also update the value of the additional field as you require.



The grid on the **Action log** FastTab is automatically filled in with a log of all actions that are performed on the message.

You can now either delete or update the message status. To update the message status, select **Update status**. In the **New status** field, select **Prepared**, and then select **OK**.



The message status is updated to **Prepared**, and you can now generate the report by selecting **Generate report**. The report is generated, and the message status and action log are updated. To view the generated report, select the **Attachment** button (the paper clip symbol) in the upper-right corner of the page.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Data management overview

2/18/2021 • 19 minutes to read • Edit Online

This topic describes how you can use the data management framework to manage data entities and data entity packages in Finance and Operations.

The data management framework consists of the following concepts:

- **Data entities** - A data entity is a conceptual abstraction and encapsulation of one or more underlying tables. A data entity represents a common data concept or functionality, for example, Customers or Vendors. Data entities are intended to be easily understood by users familiar with business concepts. After data entities are created, you can reuse them through the Excel Add-in, use them to define import/export packages, or use them for integrations.
- **Data project** - A project that contains configured data entities, which include mapping and default processing options.
- **Data job** - A job that contains an execution instance of the data project, uploaded files, schedule (recurrence), and processing options.
- **Job history** - Histories of source to staging and staging to target jobs.
- **Data package** - A single compressed file that contains a data project manifest and data files. This is generated from a data job and used for import or export of multiple files with the manifest.

The data management framework supports using data entities in the following core data management scenarios:

- Data migration
- Set up and copy configurations
- Integration

Data entities

Data entities provide conceptual abstraction and encapsulation of underlying table schema that represent data concepts and functionalities. In Microsoft Dynamics AX 2012, most tables, like the Customer and Vendor tables, were de-normalized and split into multiple tables. This was beneficial from a database design point of view, but made it difficult for implementers and ISV's to use without a thorough understanding of the physical schema. Data entities were introduced as part of data management to be used as a layer of abstraction to easily understand by using business concepts. In previous versions there were multiple ways to manage data, such as Microsoft Excel Add-ins, AIF, and DIXF. The concept of data entities combines those different concepts into one. After data entities are created, you should be able to reuse them for an Excel Add-ins, import/export, or integration. The following table shows core data management scenarios.

Data Migration	• Migrate reference, master, and document data from legacy or external systems.
Setup and copy configuration	• Copy configuration between company/environments. • Configure processes or modules using the Lifecycle Services (LCS) environment.

Integration	• Real-time service based integration. • Asynchronous integration.

Data migration

Using the data management framework, you can quickly migrate reference, master, and document data from legacy or external systems. The framework is intended to help you quickly migrate data by using the following features:

- You can select only the entities you need to migrate.
- If an import error occurs, you can skip selected records and choose to proceed with the import using only the good data, opting to then fix and import the bad data later. You will be able to partially continue and use errors to quickly find bad data.
- You can move data entities straight from one system to another, without having to go through Excel, or XML.
- Data imports can be easily scheduled using a batch, which offers flexibility when it is required to run. For example, you can migrate customer groups, customers, vendors, and other data entities in the system at any time.

Set up and copy configuration

You can use the data management framework to copy configurations between companies or environments, and configure processes or modules using Microsoft Dynamics Lifecycle Services (LCS).

Copying configurations is intended to make it easier to start a new implementation, even if your team doesn't deeply understand the structure of data that needs to be entered, or data dependencies, or which sequence to add data to an implementation.

The data management framework allows you to:

- Move data between two similar systems
- Discover entities and dependencies between entities for a given business process or module
- Maintain a reusable library of data templates and datasets
- Use data packages to create incremental data entities. Data entities can be sequenced inside the packages. You can name data packages, which can be easily identifiable during import or export. When building data packages, data entities can be mapped to staging tables in grids or by using a visual mapping tool. You can also drag-and-drop columns manually.
- View data during imports, so you can compare data, and ensure that it is valid.

Working with data entities

The following sections provide quick snapshots of the different functionalities of data management using data entities. The goal is to help to you strategize and make effective decisions on how to best utilize the available tools during data migration. You will also find tips and tricks on how to effectively use each area during data migration. A list of available data entities for each area can also be found with the suggested data sequences, showing data dependencies. Microsoft provides data packages that can be found on Lifecycle Services (LCS) as an initial guide. The information in this document can be used as a guide for creating your own packages. The description of each data entity shows what the object contains and if it is needed during data migration.

Sequencing

There are two types of sequencing that should be considered when working with data entities.

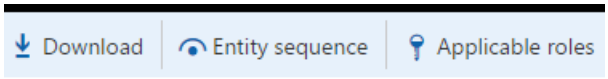
- Sequencing data entities within a data package
- Sequencing the order of data package imports

Sequence data entities within a data package

1. When a user adds data entities to a data project, by default, a sequence is set for the order in which the entities will load. The first entity added to the project will be set as the first entity to load, the next entity added will be second, the next entity will be third, and so on.

   For example, if a user added two entities in this order, **Sales tax codes** and **Sales Tax groups**, then **Sales tax codes** is assigned an entity sequence of **1.1.1**, and **Sales tax groups** is assigned an entity sequence of **1.1.2**. The sequence level indicates that the second entity will not start the import process until the first level is finished.

2. To view or edit a sequence, click the **Entity sequence** button on the Action Pane of the data project.



3. In the Definition group entity sequence, you can see the execution units and the sequence. You can change sequence by selecting the data entity in the list, setting a different Execution unit or Sequence in level, and then clicking **Update selected**. After clicking **Update selected**, the entity will move up or down in the entity list.

Example

The following screenshot shows the entity sequence that is set for the Sales Tax CodeGroups data package.

Definition group entity sequence

Entity	Execution unit	Level in execution unit	Sequence in level	Fail level on error	Fail execution u...
Sales tax codes	1	1	1	☐	☐
Sales tax code values	1	1	2	☐	☐
Sales tax code limits	1	1	3	☐	☐
Sales tax groups	1	1	4	☐	☐
Sales tax group details	1	1	5	☐	☐
Sales tax item groups	1	1	6	☐	☐
Sales tax exempt numbers	2	1	1	☐	☐
Sales tax exempt code	3	1	1	☐	☐
Sales tax reporting codes	4	1	1	☐	☐

In order to successfully import sales tax codes and groups, the sales tax codes and details have to be loaded first, before sales tax groups can be imported. Sales tax codes and groups are all in Execution unit = 1, but the sequences are in the order that they will be imported. Other related sales tax entities that are not dependent upon other data entities being loaded are included in the package. For example, sales tax exempt numbers is set in its own Execution unit = 2. This data entity will start loading immediately because there are no dependencies on other entities loading before it.

Sequence data package imports

In order to successfully load data, it's important to set the correct order for importing data packages, because of dependencies that exist within and across modules. The numbering format that has been created for the data packages within LCS are as follows:

- First segment: Module
- Second segment: Data type (setup, master, transaction)
- Third segment: Sequence number

The following tables provide more information about the default numbering format.

Module numbers

Module	Module Reference
System administration	01
General ledger	03
Public Sector	04
HRM	05
Accounts payable	10
Accounts receivable	11
Budgeting	12
Cash and bank management	13
Compliance and internal controls	14
Cost accounting	15
Fixed assets	16
Inventory management	19
Master planning	20
Organization administration	21
Payroll	22
Procurement and sourcing	23
Product information management	24
Production control	25
Project management and accounting	26
Retail	27
Sales and marketing	28
Service management	29
Trade allowance management	31
Transportation management	32
Travel and expense	33
Warehouse management	34

Data type numbers

Data Type	Data Type Reference
Setup	1
Master	4
Transaction	8

Sequence number

Numbering Format	
Module # . Data Type Reference . 001 (Sequence Number)	
01.1.001	System / Setup Data / Seq 001
01.1.002	System / Setup Data / Seq 002
01.4.001	System / Master Data / Seq 001
01.4.002	System / Master Data / Seq 002
03.1.001	General Ledger / Setup Data / Seq 001

Data packages follow the sequence number, followed by the module abbreviation, and then a description. The following example shows General ledger data packages.

Mapping

When working with data entities, mapping an entity to a source is automatic. The automatic mapping of fields

can be overridden if needed.

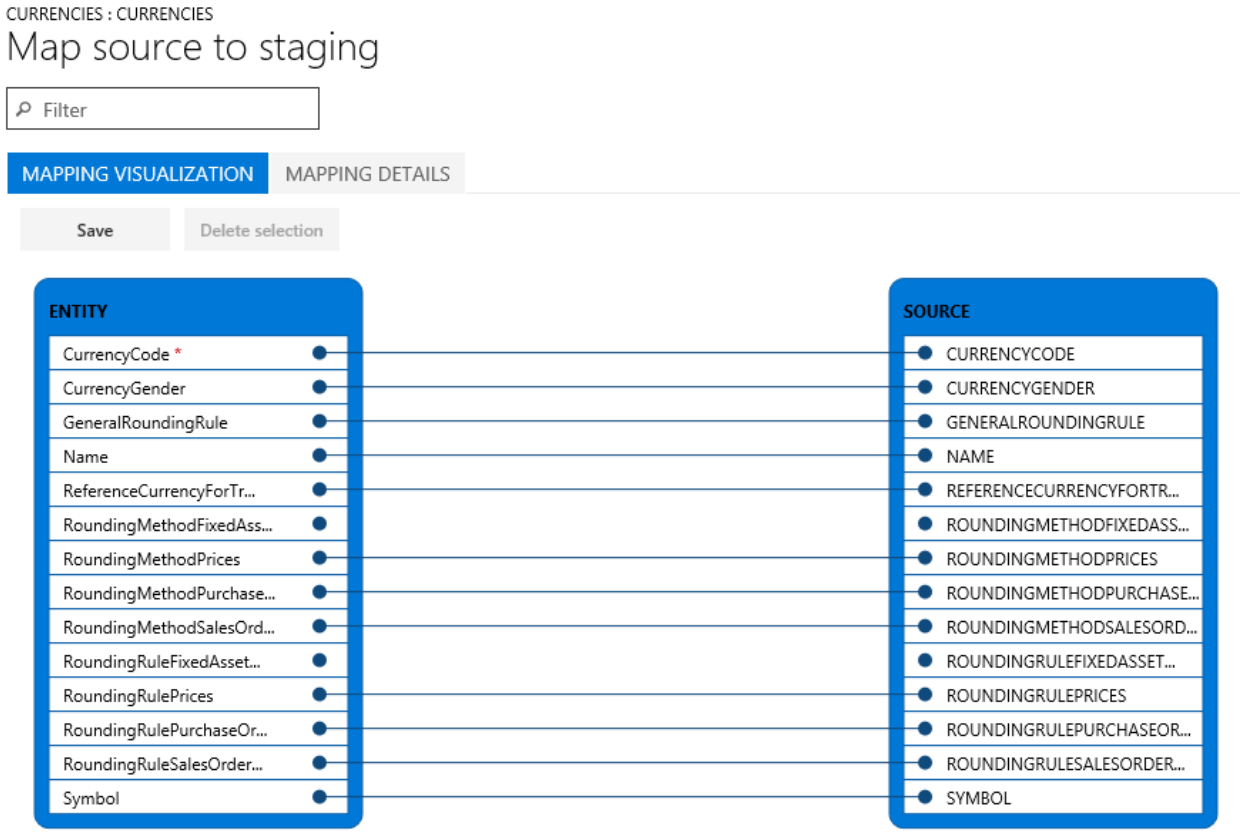
View mapping

To view how an entity is mapped, locate the tile for the entity in the project, and then click **View map**.

We provide mapping visualization view (default) and mapping details view. A red asterisk (*) identifies any required fields in an entity. These fields must be mapped in order to work with the entity. Other fields can be unmapped as required when working with the entity.

- To unmap a field, highlight the field in either column (**Entity** or **Source**), click **Delete selection**, and then click **Save**. After saving, close the form to return to the project.

The field mapping from source to staging can also be edited after import using the same process.



Regenerate a map

If you have extended an entity (added fields) or if the automatic mapping appears to be incorrect, the mapping of the entity can be regenerated in the **Mapping** form.

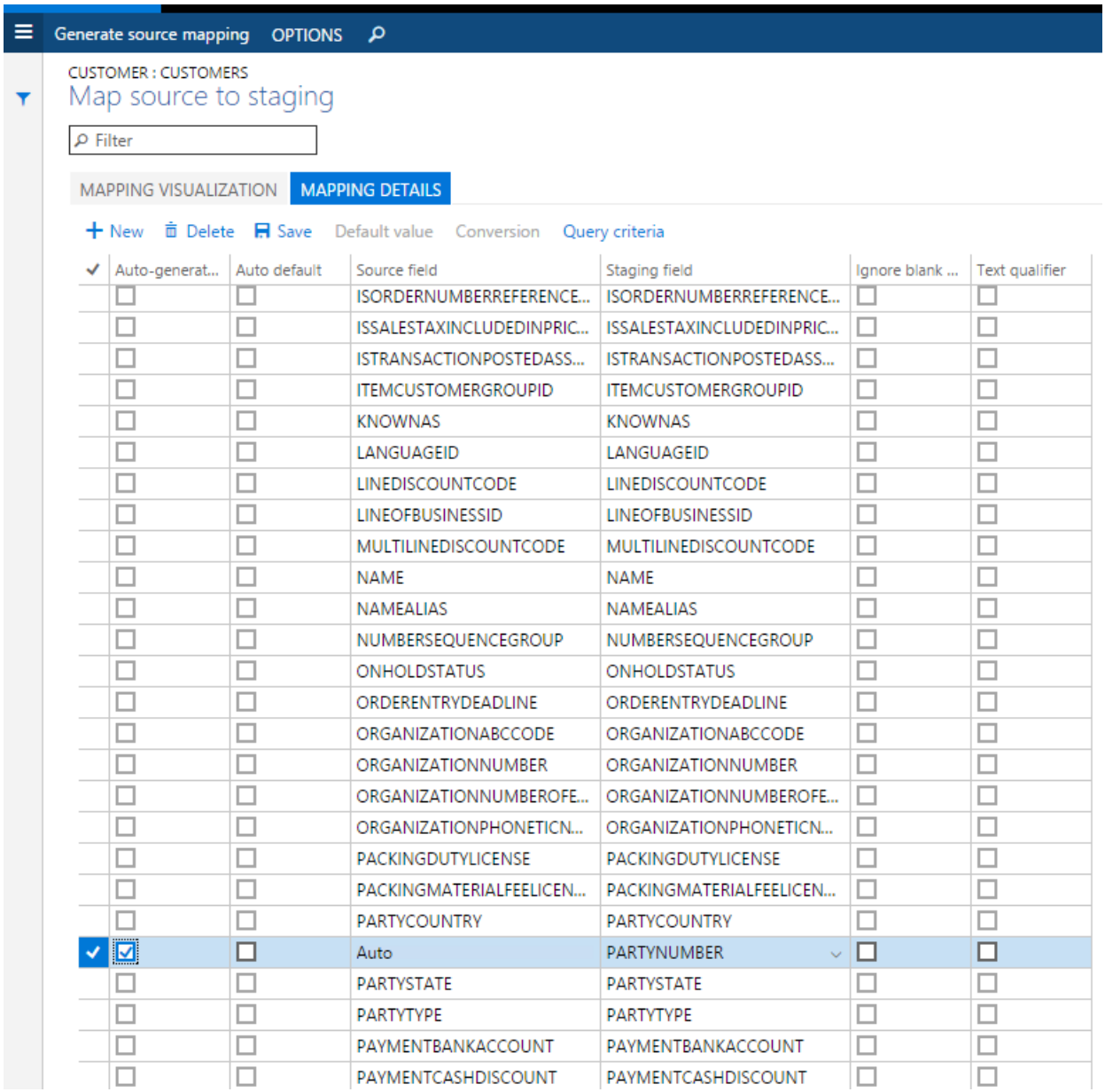
1. To do this, click **Generate source mapping**.

   A message will display asking, "Do you want to generate the mapping from scratch ?"

2. Click **Yes** to regenerate the mapping.

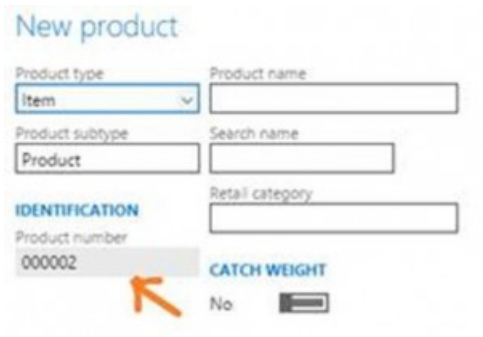
Generate data

If you have fields in entities that you want the system to generate data for on import, instead of providing the data in the source file, you can use the auto-generated functionality in the mapping details for the entity. For example, if you want to import customers and customer address information, but the address information was not previously imported with the Global Address Book entities, you can have the entity auto-generate the party number upon import and the GAB information will be created. To access this functionality, view the map of the entity and click the **Mapping details** tab. Select the fields that you want to auto-generate. This will change the source field to **Auto**.

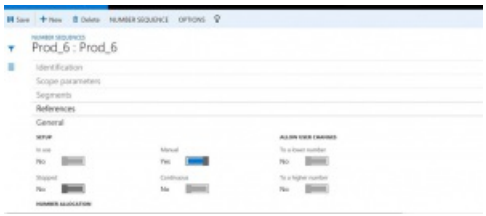


Turn off automatically generated number sequences

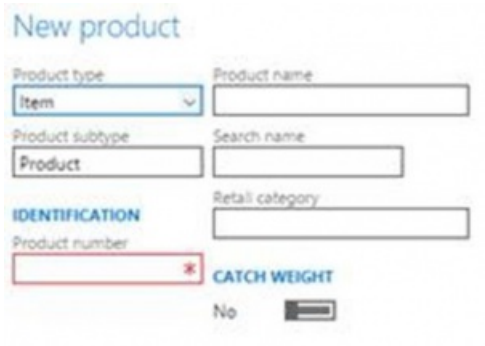
Many entities support automatic generation of identifiers based on number sequence setup. For example, when creating a product, the product number is automatically generated and the form does not allow you to edit values manually.



It is possible to enable manual assignment of number sequences for a specific entity.

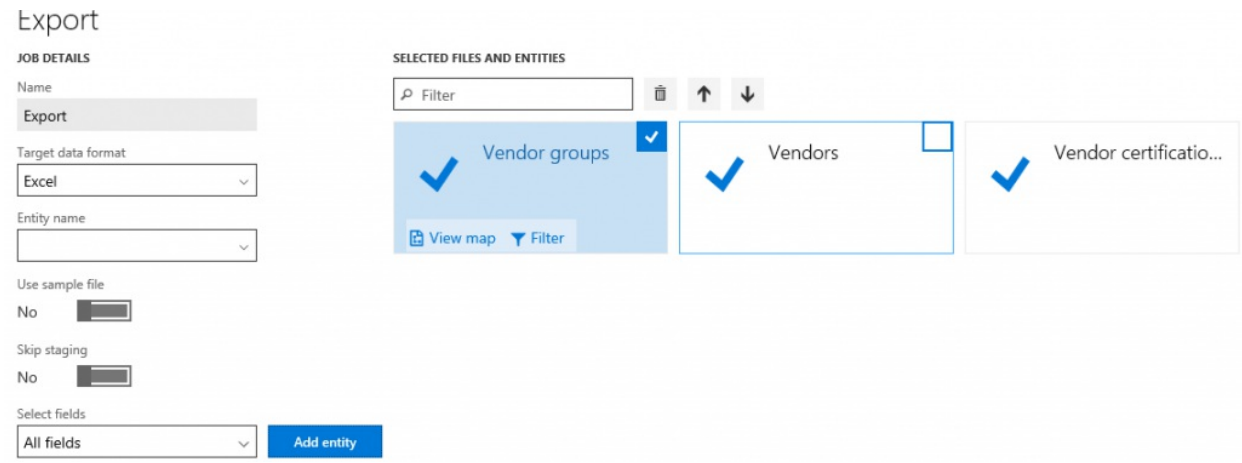


After you have enabled manual assignment, you can provide manually assigned numbers instead.

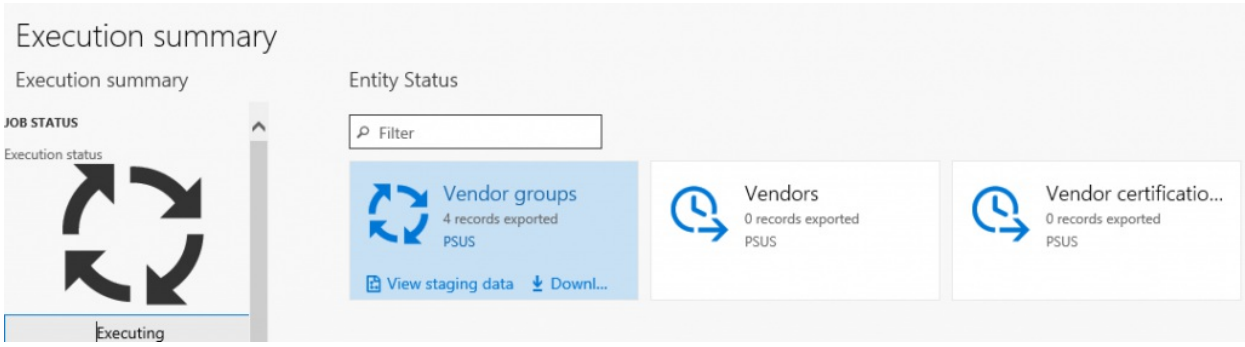


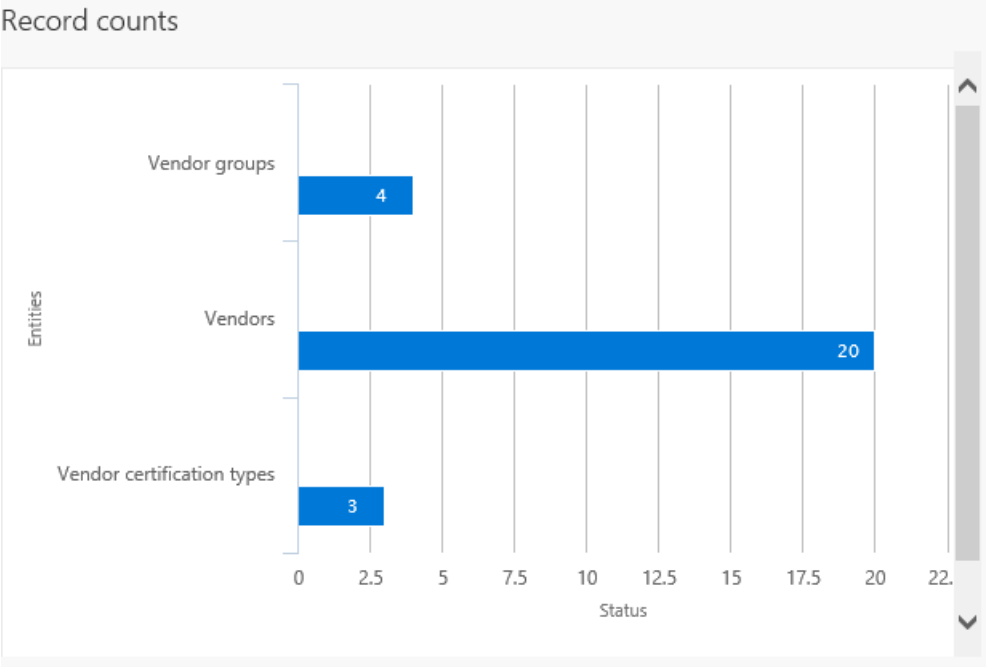
Export

Export is the process of retrieving data from a system using data entities. The export process is done through a project. When exporting, you have a lot of flexibility as to how the export project is defined. You can choose which data entities to export, but also the number of entities, the file format used (there are 14 different formats to choose for export), and apply a filter to each entity to limit what is exported. After the data entities have been pulled into the project, the sequencing and mapping described earlier can be performed for each export project.



After the project is created and saved you can export the project to create a job. During the export process, you can see a graphical view of the status of the job and the record count. This view shows multiple records so you can review the status of each record prior to downloading the actual files.





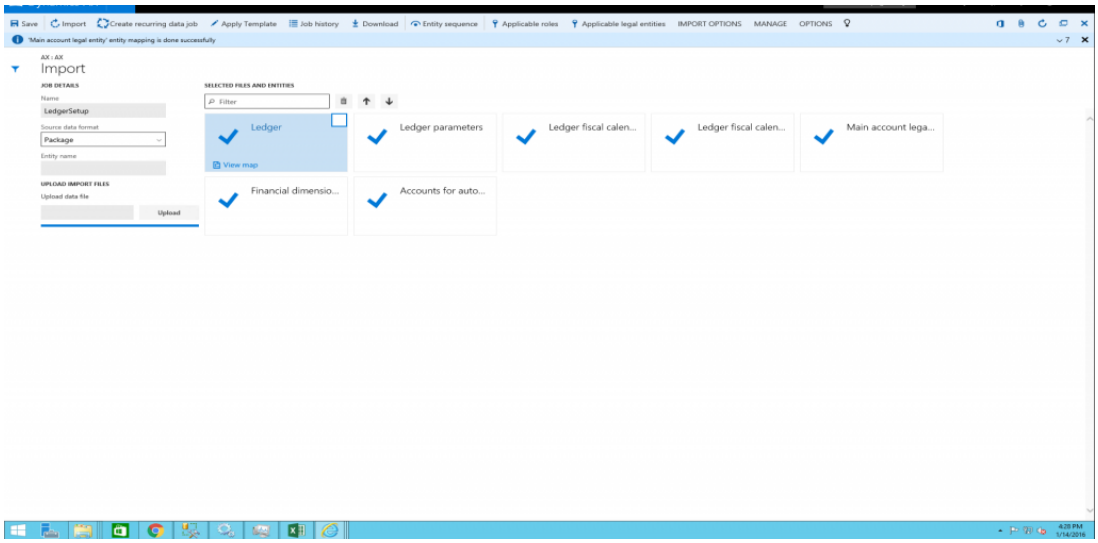
After the job is completed you can choose how to download the files: each data entity can be a separate file, or by combining the files into a package. If there are multiple data entities in the job, choosing the package option will speed up the upload process. The package is a zip file, containing a data file for each entity as well as a package header and manifest. These additional documents are used when importing in order to add the data files to the correct data entities and sequence the import process.

Import

Import is the process of pulling data into a system using data entities. The import process is done through the **Import** tile in the **Data Management** workspace. Data can be imported either for individual entities or for a group of logically related entities that are sequenced in the correct order. The file formats vary depending on the type of import. For an entity, it can be an Excel file that is comma-separated, tab-separated, or text. For a data package, it is a .zip file. In both cases, the files are exported using the above mentioned export process.

Import a data package

1. Log into the environment using a login with sufficient privileges (typically this is the Administrator role).
2. On the dashboard, click the **Data Management** workspace.
3. Click the **Import** tile.
4. On the next page, do the following:
    a. Provide a name.
    b. In the **Source Data Format** field, select **Package**.
    c. Click the **Upload** button and choose the appropriate package file from the location for the data being imported. This will import all the files from the package.

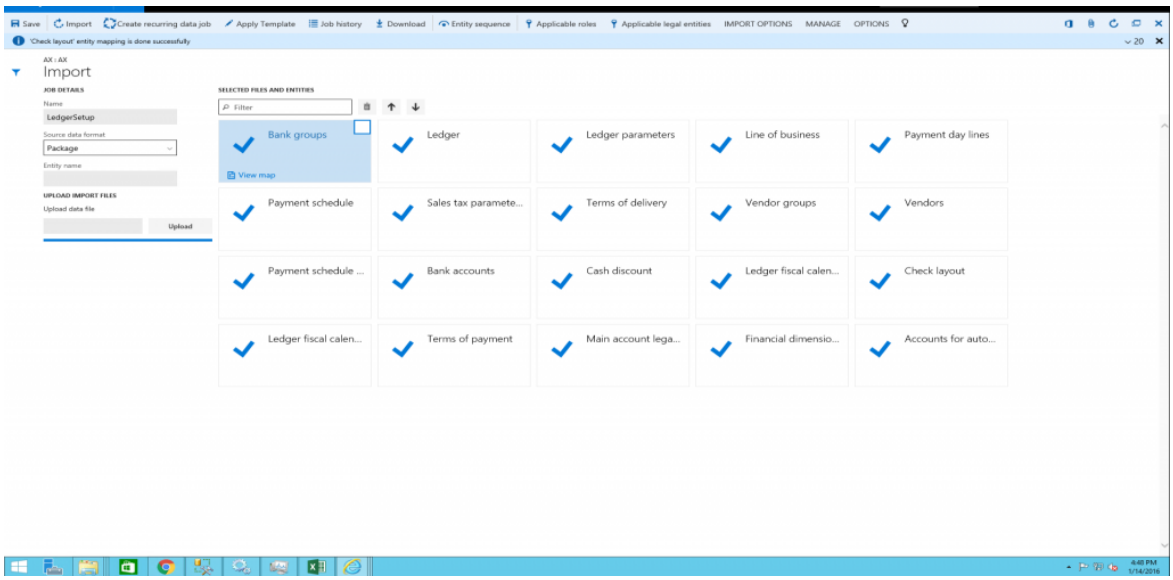


d. Click **Save**, and then click **Import**.

Import multiple data packages

Use one of the following methods to import multiple data packages.

- Create a new job for each package, and then repeat steps 4(a) through 4(d) above, for each package.
- Create one job to import multiple packages in a sequence. Repeat steps 4(a) through 4(c) above, and then repeat step 4(c) for all packages that need to be imported. After you select the packages, execute step 4(d) to import the data from the selected data packages through a single job.



After you click **Import**, the data will be imported through staging tables. The progress of the import can be tracked using the **Refresh** button in the upper-right corner of the screen.

Troubleshoot data package processing

This section provides troubleshooting information for the different stages of data package processing.

- Status and error details of a scheduled job can be found under the **Job history** section in the **Data management** form.
- Status and error details of previous runs for data entities can be displayed by selecting a data project and clicking **Job history**. In **the Execution history** form, select a job, and click **View staging data** and **View execution log**. The previous runs include data project runs that were executed as batch jobs or manually.

Export process troubleshooting

- If you get an error during the export process, click **View execution log** and review the log text, staging log details, and Infolog for more information.
- If you get an error during the export process with a note directing you to not skip staging, turn off the **Skip staging** option, and then add the entity. If you are exporting multiple data entities, you can use the **Skip staging** button for individual data entities.
- Prior to platform update 32, there was a 256 MB limit for the file size that can be handled via export. If there are a large number of records that will be exported, be sure that the resulting file size does not exceed this limit. An alternate way to handle such scenarios would be to use filters on the entity to export only a subset of data. If this is not feasible, then bring your own database must be considered for the overall solution. This limitation no longer exists starting in Platform update 32.

Import process troubleshooting

When uploading data entity files:

- If data entities do not display in **Selected files and entities** after you click **Upload** during the import process, wait a few minutes, and then check whether the OLEDB driver is still installed. If not, then reinstall the OLEDB driver. The driver is Microsoft Access Database Engine 2010 Redistributable – AccessDatabaseEngine_x64.exe.
- If data entities display in **Selected Files and Entities** with a warning after you click **Upload** during the import process, verify and fix the mapping of individual data entities by clicking **View map**. Update the mapping and click **Save** for each data entity.

During data entity import:

- If data entities fail (shown with a red X or yellow triangle icon on the data entity tile) after you click **Import**, click **View staging data** on each tile under the **Execution summary** page to review the errors. Sort and scroll through the records with Transfer status = Error to display the errors in the Message section. Download the staging table. Fix a record (or all records) directly in staging by clicking **Edit, Validate all, and Copy data to target**, or fix the import file (not staging file) and reimport the data.
- If data entities fail (shown with a red x or yellow triangle icon on the data entity tile) after you click **Import**, and **View staging data** shows no data, go back to the **Execution summary** page. Go to **View execution log**, select the data entity, and review the **Log text, Staging log details, and Infolog** for more information. **Staging log details** will display **Error column** (field) details and **Log description** will describe errors in detail.
- If data entities fail, you can check the import file to see if there's an extra line in the file with text which displays, "This is a string that is inserted into Excel as a dummy cell to make the column to support more than 255 characters. By default, an Excel destination component will not support more than 255 characters. The default type of Excel will be set based on the first few rows". This line is added during data export. If this line exists, delete this line, re-package the data entity, and try to import.

Features flighted in data management and enabling flighted features

The following features are enabled via flighting. *Flighting* is a concept that allows a feature to be ON or OFF by default.

FLIGHT NAME	DESCRIPTION
DMFEnableAllCompanyExport	Enables BYOD export from all companies in the same export job (supported for BYOD only and not files). By default, this is OFF. This flight is no longer needed after Platform update 27 because this feature can be turned ON using a parameter in data management framework parameters.

FLIGHT NAME	DESCRIPTION
DMFExportToPackageForceSync	Enables synchronous execution of data package API export. By default, it's asynchronous.
EntityNamesInPascalCaseInXMLFiles	Enables behavior where entity names are in Pascal Case in the XML files for entities. By default, the names are in upper case.
DMFByodMissingDelete	Enables the old behavior where under certain conditions, certain delete operations were not synced to BYOD using change tracking.
DMFDisableExportFieldsMappingCache	Disables caching logic when building target field mapping.
EnableAttachmentForPackageApi	Enables attachments functionality in the package API.
FailErrorOnBatchForExport	Enables fail on error at execution unit or level for export jobs.
IgnorePreventUploadWhenZeroRecord	Disables 'prevent upload when zero records' functionality.
DMFInsertStagingLogToContainer	By default this is ON. This improves performance and functional issues with error logs in the staging table.
ExportWhileDataEntityListIsBeingRefreshed	When enabled, additional validations are made on mappings when a job is scheduled while entity refresh is in progress. By default, this is OFF.
DMFDisableXSLTTransformationForCompositeEntity	This can disable the application of transformations on composite entities.
DMFDisableInputFileCheckInPackageImport	Additional validations are made to ensure if any entity file is missing from a data package, error message is shown. This is the default behavior. If required, this can be turned OFF by this flight.
FillEmptyXMLFileWhenExportingCompositeEntity	Prior to Platform update 15, when exporting composite entities that did not have any records to export, the XML file generated did not have any schema elements. This behavior can be changed to output empty schema by enabling this flight. By default, the behavior will still be to output empty schema.
EnableNewNamingForPackageAPIExport	A fix was made to ensure unique names are used for the execution ID when ExportToPackage is used for export scenarios. Duplicate execution ID's were being created when ExportToPackage was called in quick succession. To preserve compatibility, this behavior is OFF by default. Turning this flight ON will enable this new behavior where new naming convention for execution ID's will ensure unique names.
DMFDisableDoubleByteCharacterExport	A fix was made to ensure that data can be exported when the format is configured to use code page 932 setting. If an issue is encountered in relation to double byte exports, this fix can be turned off by disabling this flight to unblock, if applicable.

FLIGHT NAME	DESCRIPTION
DisablePendingRecordFromJobStatus	A fix was made to ensure that pending records are taken into consideration while evaluating the final status of an import job. If implementations have a dependency on the status evaluation logic and this change is considered a breaking change for an implementation, this new logic can be disabled using this flight.
DMFDisableEnumFieldDefaultValueMapping	A fix was made to ensure that default values set in advanced mapping for enum fields are successfully saved in the data package manifest file when generating the data package. This makes it possible for the data package to be used as a template for integrations when such advanced mappings are used. This fix is protected by this flight and can be disabled if the previous behavior is still needed (which is to always set the value to 0 in the data package manifest).
DMFXsltEnableScript	This flight only applies to Platform update 34 and non-production environments. A fix was made in Platform update 34 to prevent scripting in XSLT. However, this resulted in breaking some functionality that was dependent on scripting. As a result, this flight has been turned ON by Microsoft in all production environments as a preventive measure. In non-production environments, this must be added by customers if they encounter XSLT failures related to scripting. From Platform update 35 onward, a code change was made to revert the Platform update 34 change so this flight does not apply from Platform update 35 onward. Even if you enabled this flight in Platform update 34, upgrading to Platform update 35 will not cause any negative impact due to this flight being ON from Platform update 34.
DMFExecuteSSISInProc	This flight is OFF by default. This is related to a code fix that was made to run SQL Server Integration Services (SSIS) out of in-process to optimize memory utilization of SSIS when running DIXF jobs. However, this change has caused a regression in a scenario where if the DIXF data project name has an apostrophe (') in it, then the job will fail with an error. If you encounter this issue, removing the (') in the data project name will resolve the failure. However, if for some reason the name cannot be changed, then this flight can be enabled to overcome this error. Enabling this flight will make SSIS run in-process as before, which could lead to higher memory consumption when running DIXF jobs.

The following steps enable a flight in a non-production environment. Execute the following SQL command.

For enabling flights in a production environment, a support case must be logged with Microsoft.

- After running the SQL statement, ensure that the following is set in the web.config file on each of the AOS's. add key="DataAccess.FlightingServiceCatalogID" value="12719367"
- After making the above change, perform an IISReset on all AOS's.

```
INSERT INTO SYSFLIGHTING
([FLIGHTNAME]
,[ENABLED]
,[FLIGHTSERVICEID]
,[PARTITION]
,[RECID]
,[RECVERSION]
)
VALUES ('name', 1, 12719367, PARTITION, RECID, 1)
```

- Partition - Partition ID from the environment, which can be obtained by querying (select) for any record. Every record will have a partition ID that must be copied and used here.
- RecID - Same ID as partition. However, if multiple flights are enabled, then this can be partition ID + 'n' to ensure it has a unique value
- RecVersion = 1

Additional resources

- Data entities overview

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Data entities overview

2/18/2021 • 11 minutes to read • Edit Online

This topic defines and provides an overview of data entities. It includes information about the capabilities of data entities, the scenarios that they support, the categories that are used for them, and the methods for creating them.

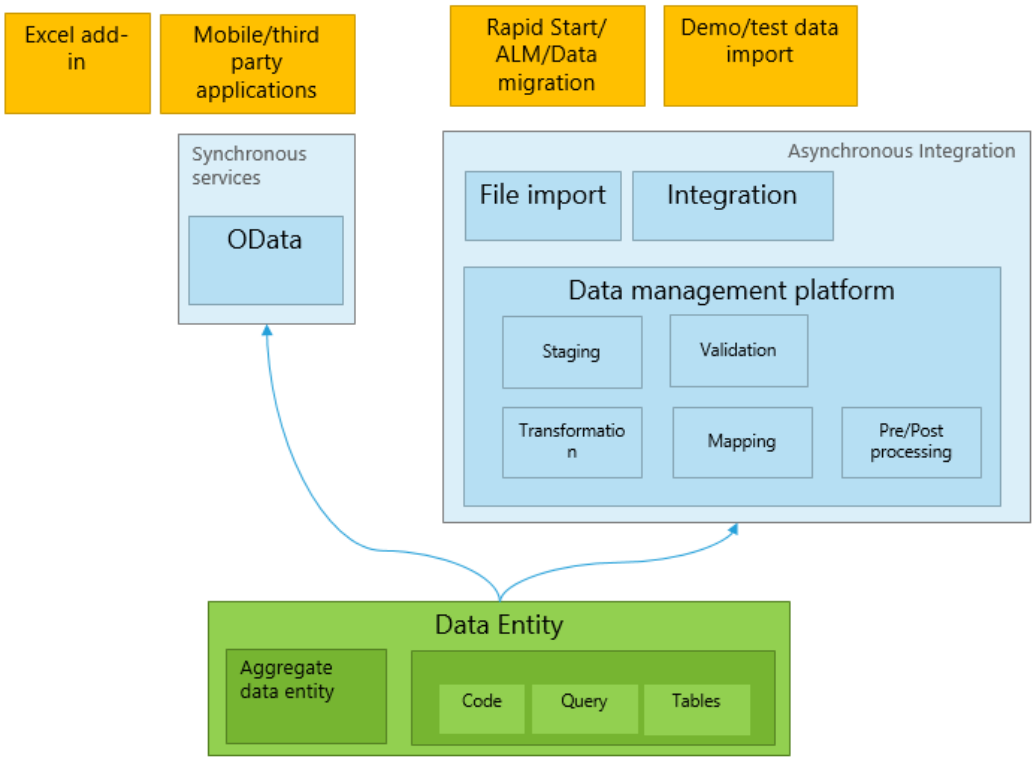
Overview

A *data entity* is an abstraction from the physical implementation of database tables. For example, in normalized tables, a lot of the data for each customer might be stored in a customer table, and then the rest might be spread across a small set of related tables. In this case, the data entity for the customer concept appears as one de-normalized view, in which each row contains all the data from the customer table and its related tables. A data entity encapsulates a business concept into a format that makes development and integration easier. The abstracted nature of a data entity can simplify application development and customization. Later, the abstraction also insulates application code from the inevitable churn of the physical tables between versions. **To summarize:** Data entity provides conceptual **abstraction** and **encapsulation** (de-normalized view) of underlying table schemas to represent key data concepts and functionalities.

Capabilities

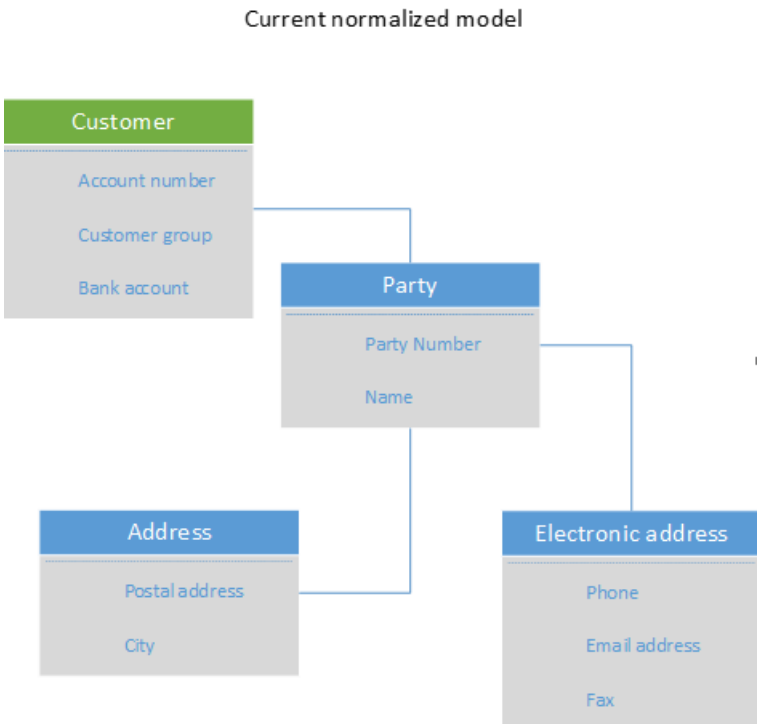
A data entity has the following capabilities:

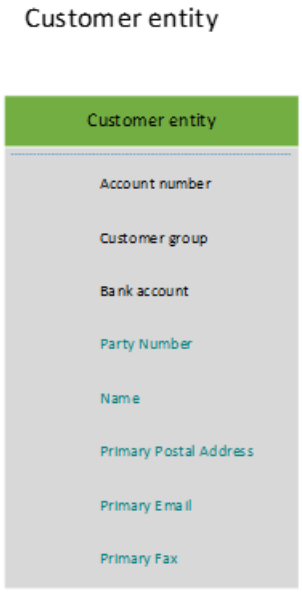
- It replaces diverging and fragmented concepts of AXD, Data Import/Export Framework (DIXF) entities, and aggregate queries with single concept.
- It provides a single stack to capture business logic, and to enable scenarios such as import/export, integration, and programmability.
- It becomes the primary mechanism for exporting and importing data packages for Application Lifecycle Management (ALM) and demo data scenarios.
- It can be exposed as OData services, and then used in tabular-style synchronous integration scenarios and Microsoft Office integrations.



Entity example

A consumer wants to access data that is related to a customer object, but this data is currently scattered across multiple normalized tables, such as DirParty, CustTable, LogisticPostalAddress, and LogisticElectronicAddress. Therefore, the process of reading and writing customer data is very tedious. Instead, the following customer entity can be designed to encapsulate the entire underlying physical schema into a single de-normalized view. This enables simpler read/write operations and also enables abstraction of any internal interaction between the tables.





Supported scenarios

Data entities support all the following scenarios.

Integration scenarios

Synchronous service (OData)

Data entities enable public application programming interfaces (APIs) on entities to be exposed, which enables synchronous services. Synchronous services are used for the following purposes:

- Office integration
- Third-party mobile apps

Asynchronous integration

Data entities also support asynchronous integration through a data management pipeline. This enables asynchronous and high-performing data insertion and extraction scenarios. Here are some examples:

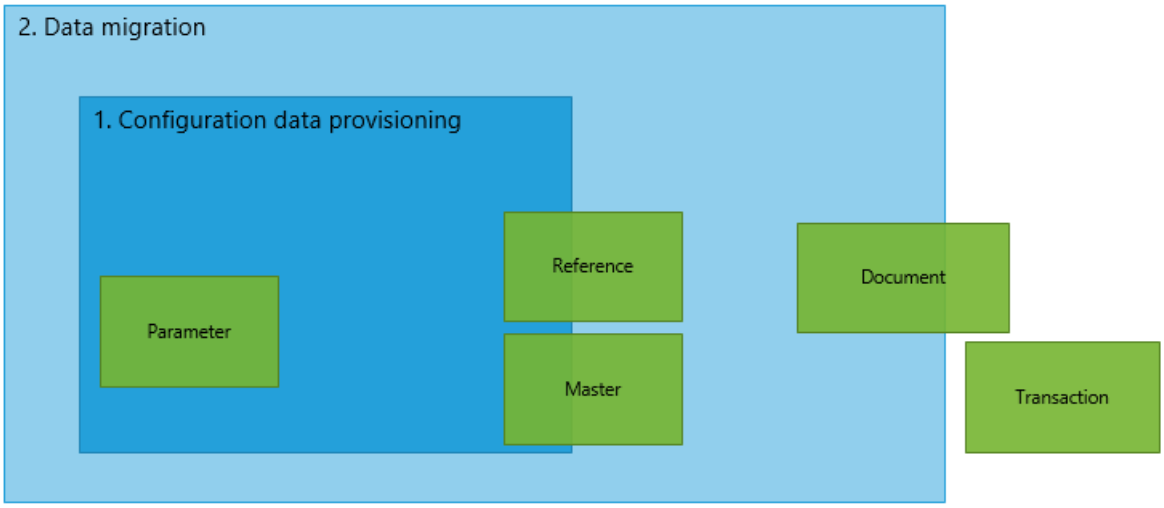
- Interactive file-based import/export
- Recurring integrations (file, queue, and so on)

Business intelligence

- Aggregate data
- Standardized key performance indicators (KPIs)

Application Lifecycle Management

Besides integration and business intelligence (BI) scenarios, data entities also initially support two critical ALM scenarios. The following two progressive levels of an ALM scenario show the scope of coverage by data entities.



Configuration data provisioning

A system implementer will use both a guided data collection wizard and bulk data input mechanisms to **bootstrap the initial deployment** (or module) with configuration data through Microsoft Dynamics Lifecycle Services (LCS). Configuration primarily targets to cover the following entity categories:

- All of Parameter
- Reference
- System parameter
- Number sequence
- Currency

Data migration from legacy or external systems

After the initial deployment is up and running, the system implementer will **migrate existing data assets of the customer** into the application, especially the following assets:

- Master data (for example, customers and vendors)
- Subsets of documents (for example, sales orders)

Categories of entities

Entities are categorized based on their functions and the type of data that they serve. The following are five categories for data entities.

Parameter

- Functional or behavioral parameters.
- Required to set up a deployment or a module for a specific build or customer.
- Can include data that is specific to an industry or business. The data can also apply to a broader set of customers.
- Tables that contain only one record, where the columns are values for settings. Examples of such tables exist for Account payable (AP), General ledger (GL), client performance options, workflows, and so on.

Reference

- Simple reference data, of small quantity, which is required to operate a business process.
- Data that is specific to an industry or a business process.
- Examples include units, dimensions, and tax codes.

Master

- Data assets of the business. Generally, these are the "nouns" of the business, which typically fall into categories such as people, places, and concepts.

- Complex reference data, of large quantity. Examples include customers, vendors, and projects.

Document

- Worksheet data that is converted into transactions later.
- Documents that have complex structures, such a several line items for each header record. Examples include sales orders, purchase orders, open balances,and journals.
- The operational data of the business.

Transaction

- The operational transaction data of the business.
- Posted transactions. These are non idempotent items such as posted invoiced and balances. Typically, these items are excluded during a full dataset copy to reduce the volume of data that is copied/migrated. Migrating completed transactions can also lead to further complexity in trying to preserve the referential integrity of related data in the new system. In general, transactions from a completed business process are not migrated in detail but in summary.
- Examples include pending invoices.

Building an entity

There are multiple ways to create an entity. For example, you can use a wizard, or you can build an entity from a table.

Building an entity by using a wizard

The simplest way to build an entity is to use a wizard. This wizard lets you select a root data source and expand to other related data sources, and then select fields for the entity. To start the wizard, add a new item of type **Data entity** to your project. For step-by-step instructions for using the wizard to build an entity, see Build and consume data entities. The following table provides information about the properties that you set for an entity in the wizard.

PROPERTY	DESCRIPTION
Primary data source	The root data source (table or view) that is used to construct the entity. You can add more related data sources, based on this root data source.
Data entity name	The name of the entity.
Entity category	The type of entity. Entity categories are similar to table groups for tables. The available categories include **Parameter**, **Reference**, **Master**, **Document**, and **Transaction**.
Public entity name	The public resource name for the entity.
Public collection name	The public resource set name.
Enable public API	Select this option to enable the entity for OData services.
Enable data management capabilities	Select this option to enable the entity for asynchronous integrations such as data import/export and connector integration.

PROPERTY	DESCRIPTION
Staging table	The name of the staging table that will be generated for the entity. The staging table is used in asynchronous integrations and high-volume scenarios.

Adding data sources

When you build an entity, you start with a root data source. However, you can add additional data sources. You can either manually add new data sources, or select a surrogate foreign key field in the root data source to automatically expand the required data sources.

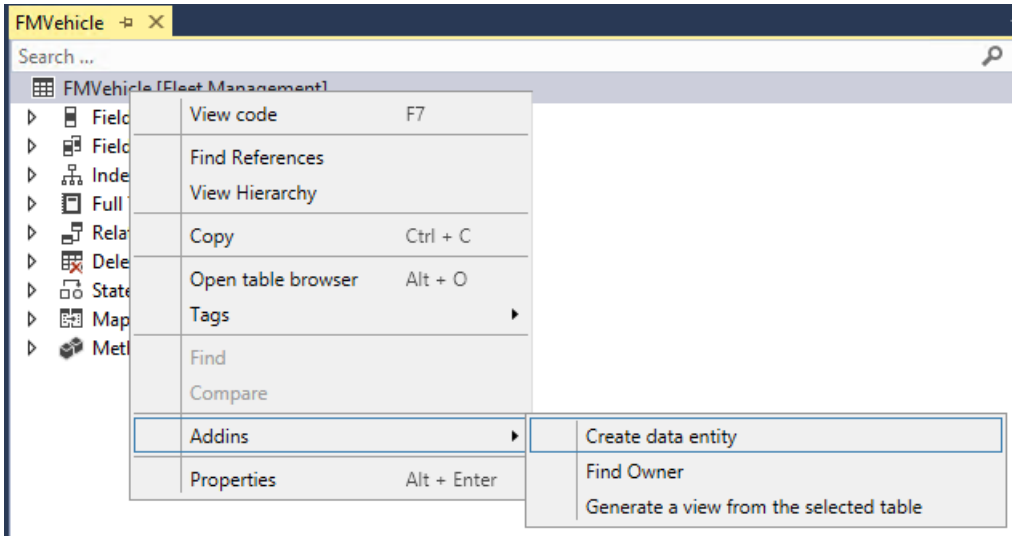
Output

When you complete the wizard, it produces the following items:

- Data entity
- Staging table (optional, if data management was enabled)

Building an entity from a table

You can quickly create an entity from a table, and then customize the properties, data sources, and fields later. Right-click the table, and then select **Addins** > **Create data entity**.



Entity list refresh

Entities in an environment must be refreshed using the following guidelines.

- When a new environment is deployed and the user navigates to the data management workspace, entity list refresh starts automatically.
- When code packages are deployed to an environment where data management has already been used, entity list refresh must be manually started from **Data management** > **Framework parameters** > **Entity settings** > **Refresh entity list**.
- When configuration keys are modified, entity list must be refreshed manually from **Data management** > **Framework parameters** > **Entity settings** > **Refresh entity list**.

Refreshing the entity list ensures all entities are available in the environment and that the entities have the latest metadata.

Configuration keys and data entities

Before you use data entities to import or export data, we recommended that you first determine the impact of configuration keys on the data entities that you are planning to use.

To learn more about configuration keys, see the License codes and configuration keys report.

Configuration key assignments

Configuration keys can be assigned to one or all of the following artifacts.

- Data entities
- Tables used as data sources
- Table fields
- Data entity fields

The following table summarizes how configuration key values, on the different artifacts that underlie an object, change the expected behavior of the object.

CONFIGURATION KEY SETTING ON DATA ENTITY	CONFIGURATION KEY SETTING ON TABLE	CONFIGURATION KEY SETTING ON TABLE FIELD	CONFIGURATION KEY ON DATA ENTITY FIELD	EXPECTED BEHAVIOR
Disabled	Not evaluated	Not evaluated	Not evaluated	If the configuration key for the data entity is disabled, the data entity will not be functional. It does not matter whether the configuration keys in the underlying tables and fields are enabled or disabled.
Enabled	Disabled	Not evaluated	Not evaluated	If the configuration key for a data entity is enabled, the data management framework checks the configuration key on each of the underlying tables. If the configuration key for a table is disabled, that table will not be available in the data entity for functional use. If a table's configuration key is disabled, the table and data entity configuration key settings are not evaluated. If the primary table in the entity has its configuration key disabled, then the system will act as though the entity's configuration key were disabled.

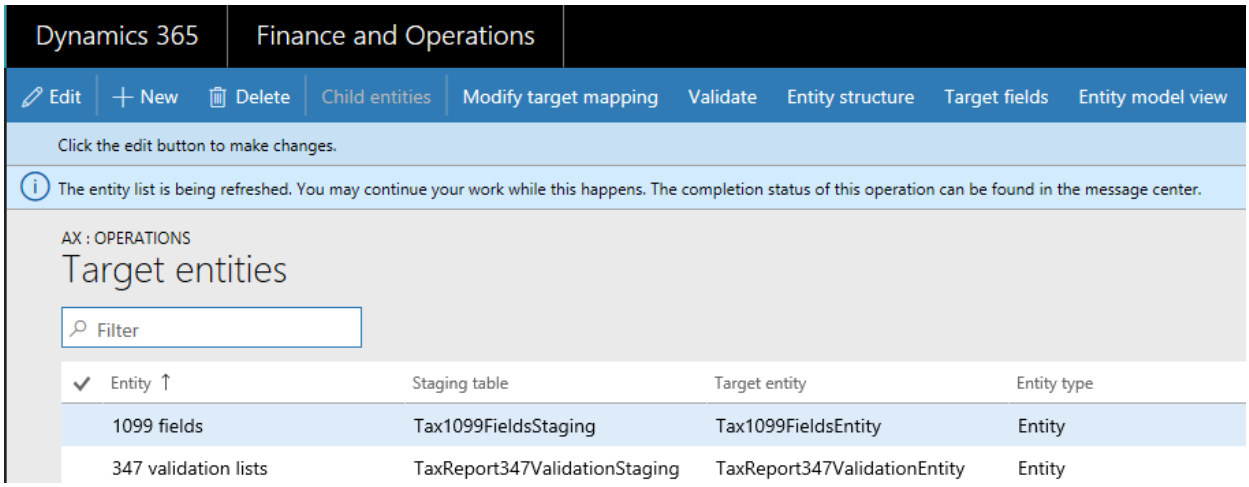
CONFIGURATION KEY SETTING ON DATA ENTITY	CONFIGURATION KEY SETTING ON TABLE	CONFIGURATION KEY SETTING ON TABLE FIELD	CONFIGURATION KEY ON DATA ENTITY FIELD	EXPECTED BEHAVIOR
Enabled	Enabled	Disabled	Not evaluated	If the configuration key for a data entity is enabled, and the underlying tables configuration keys are enabled, the data management framework will check the configuration key on the fields in the tables. If the configuration key for a field is disabled, that field will not be available in the data entity for functional use even if the corresponding data entity field has the configuration key enabled.
Enabled	Enabled	Enabled	Disabled	If the configuration key is enabled at all other levels, but the entity field configuration key is not enabled, then the field will not be available for use in the data entity.

NOTE

If an entity has another entity as a data source, then the above semantics are applied in a recursive manner.

Entity list refresh

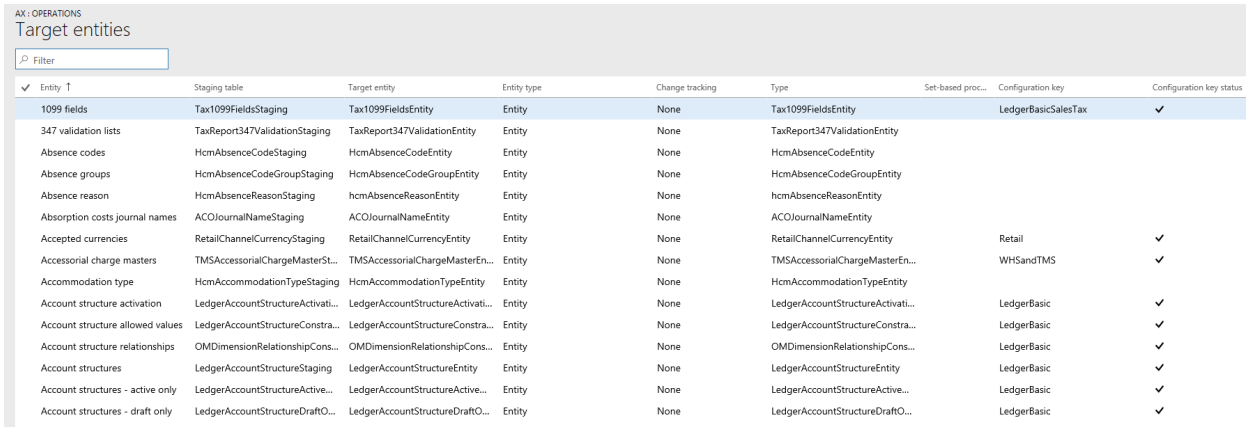
When the entity list is refreshed, the data management framework builds the configuration key metadata for runtime use. This metadata is built using the logic described above. We strongly recommend that you wait for the entity list refresh to complete before using jobs and entities in the data management framework. If you don't wait, the configuration key metadata may not be up to date and could result in unexpected outcomes. When the entity list is being refreshed, the following message is shown in the entity list page.



Data entity list page

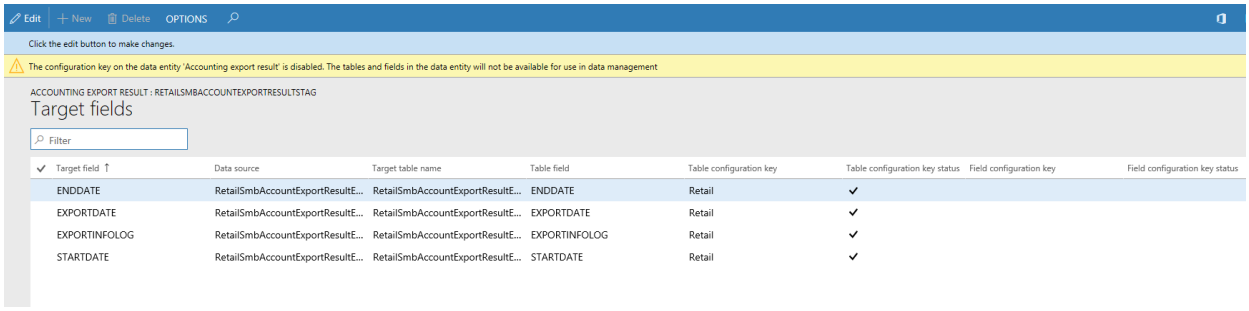
The data entity list page in the **Data management** workspace shows the configuration key settings for the entities. Start from this page to understand the impact of configuration keys on the data entity.

This information is shown using the metadata that is built during entity refresh. The configuration key column shows the name of the configuration key that is associated with the data entity. If this column is blank it means that there is no configuration key associated with the data entity. The configuration key status column shows the state of the configuration key. If it has a checkmark, it means the key is enabled. If it is blank, it means either the key is disabled or there is no key associated.



Target fields

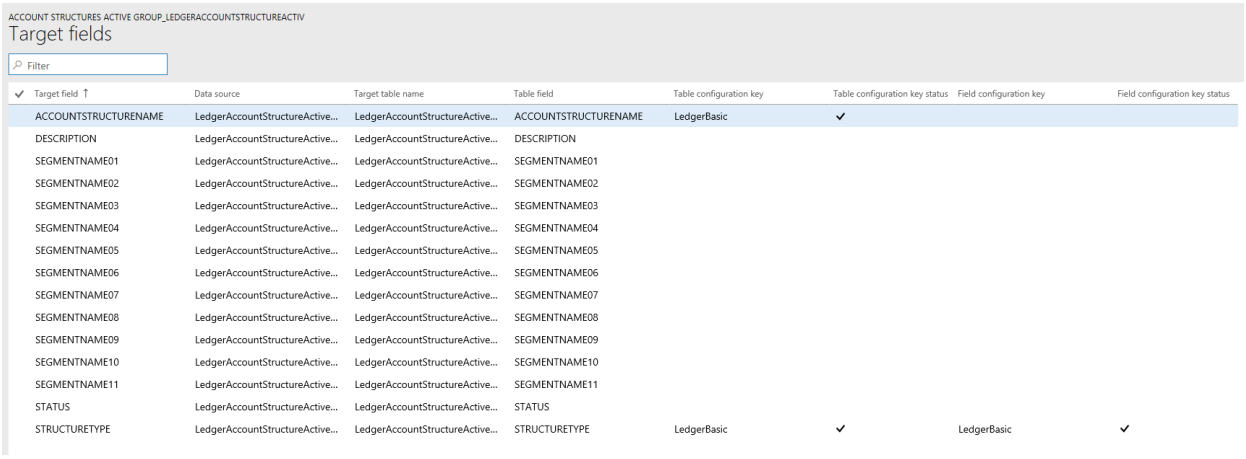
The next step is to drill into the data entity to view the impact of configuration keys on tables and fields. The target fields form for a data entity shows the configuration key and the key status information for the related tables and fields in the data entity. If the data entity itself has its configuration key disabled, a warning message is shown informing that the tables and fields in the target fields form for this entity will not be available, regardless of their configuration key status.



Child entities

Certain entities have other entities as data sources, or are composite data entities: configuration key information for these entities is shown in the **Child entities** form. Use this form in the similar way to the entities list page

described above. The target fields form for the child entity also behaves like what is described above.



Target field	Data source	Target table name	Table field	Table configuration key	Table configuration key status	Field configuration key	Field configuration key status
ACCOUNTSTRUCTURENAME	LedgerAccountStructureActive...	LedgerAccountStructureActive...	ACCOUNTSTRUCTURENAME	LedgerBasic	✓		
DESCRIPTION	LedgerAccountStructureActive...	LedgerAccountStructureActive...	DESCRIPTION				
SEGMENTNAME01	LedgerAccountStructureActive...	LedgerAccountStructureActive...	SEGMENTNAME01				
SEGMENTNAME02	LedgerAccountStructureActive...	LedgerAccountStructureActive...	SEGMENTNAME02				
SEGMENTNAME03	LedgerAccountStructureActive...	LedgerAccountStructureActive...	SEGMENTNAME03				
SEGMENTNAME04	LedgerAccountStructureActive...	LedgerAccountStructureActive...	SEGMENTNAME04				
SEGMENTNAME05	LedgerAccountStructureActive...	LedgerAccountStructureActive...	SEGMENTNAME05				
SEGMENTNAME06	LedgerAccountStructureActive...	LedgerAccountStructureActive...	SEGMENTNAME06				
SEGMENTNAME07	LedgerAccountStructureActive...	LedgerAccountStructureActive...	SEGMENTNAME07				
SEGMENTNAME08	LedgerAccountStructureActive...	LedgerAccountStructureActive...	SEGMENTNAME08				
SEGMENTNAME09	LedgerAccountStructureActive...	LedgerAccountStructureActive...	SEGMENTNAME09				
SEGMENTNAME10	LedgerAccountStructureActive...	LedgerAccountStructureActive...	SEGMENTNAME10				
SEGMENTNAME11	LedgerAccountStructureActive...	LedgerAccountStructureActive...	SEGMENTNAME11				
STATUS	LedgerAccountStructureActive...	LedgerAccountStructureActive...	STATUS				
STRUCTURETYPE	LedgerAccountStructureActive...	LedgerAccountStructureActive...	STRUCTURETYPE	LedgerBasic	✓	LedgerBasic	✓

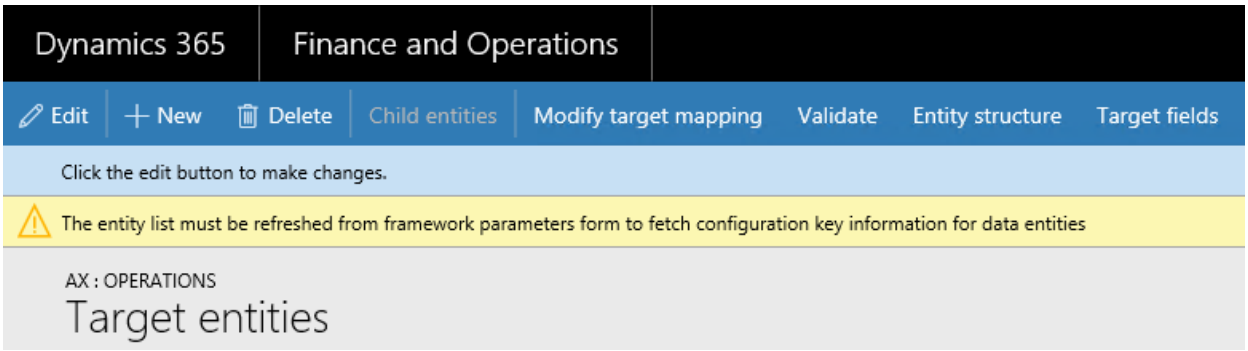
Run time validations for configuration keys

Using the configuration key metadata built during entity refresh list, run time validations are performed in the following use cases.

- When a data entity is added to a job.
- When user clicks **Validate** on the entity list.
- When the user loads a data package into a data project.
- When the user loads a template into a data project.
- When an existing data project is loaded.
- When a template is loaded into a data project.
- Before the export/import job is executed (batch, non-batch, recurring, OData).
- When the user generates mapping.
- When the user maps fields in the mapping UI.
- When the user adds only 'importable fields'.

Managing configuration key changes

Anytime that you update configuration keys at the entity, table, or field level, the entity list in the data management framework must be refreshed. This process ensures that the framework picks up the latest configuration key settings. Until the entity list is refreshed, the following warning will be shown in the entity list page. The updated configuration key changes will take effect immediately after the entity list is refreshed. We recommend that you validate existing data projects and jobs to make sure that they function as expected after the configuration keys changes are put in effect.



Configuration data projects

2/18/2021 • 4 minutes to read • Edit Online

Configuration data projects are used to manage the movement of company configuration data between instances of your application. They are intended to support the following scenarios:

- **Export of configurations** – Create configurations of entities, and use the data management framework to export the configurations to a package.
- **Import of configurations** – Upload a configuration package, and use the data management framework to import the package.

Configuration data packages are created by using data import and export projects in the **Data management** workspace. The **Data import** and **Data export** pages let you add and remove the entities that you require in order to manage the movement of company and shared data. After you create the list of entities in your configuration, you can export or import the configuration by using the data management framework to create a package. You can export packages locally and then move them to another instance for import.

Configuration data templates are predefined lists of entities for each module area that can be used in a data project. You can create, view, and modify these templates by using the **Template** page in the **Data management** workspace.

> **IMPORTANT**
> Default configuration templates were delivered in Microsoft Dynamics 365 for Finance and Operations, Enterprise edition July 2017 update. The Configuration data project feature is available in Microsoft Dynamics 365 for Operations platform update 7. You can create and use your own templates in the current product release.

Process for working with configuration data projects

We recommend that you follow this process when you start to use configuration data projects.

1. Set up your system by getting the new data configuration user interface (UI) and setting default file extensions.
2. Set up configuration templates for both export and import.
3. Create and run a configuration data project for export.
4. Create and run a configuration data project for import.

The rest of this topic describes the **Data management** workspace and explains how to set up your system for data configuration projects.

If you're ready to set up a configuration template, see Configuration data templates. If you're ready to create and run a configuration data project, see Copy configuration data between companies or legal entities overview.

Data management workspace overview

The **Data management** workspace provides access to important tasks for data management. It also provides information about projects and project execution tasks.

After you've created a configuration data project, it appears in the **Data projects** grid in the **Data management** workspace. For each project, the type of configuration (import or export) and the project category (**Project**, **Configuration**, **Integration**, or **Other**) are shown. Use the selections to the left of the grid

to filter by the appropriate project.

To open a project, select the project, and then select **Load project** to open the **Import or export** page. Use the **Delete** button to delete the selected projects. You can also download the project definitions by using the **Download** button.

Use **Job history** to find more details about the projects that you've run. Use the data range filters to filter by the dates when data projects were run. You can view execution details by selecting a job and then using the **Execution details** menu.

For export tasks, you can download the data from the workspace by using the **Download page** button.

Set up your system to manage configuration data projects

Before you begin, make sure that you have access to the new data configuration UI, and that you've set up the default data sources.

Get the new UI

We have updated the **Data management** workspace, and the **Template**, **Data export**, and **Data import** pages, so that they support configuration management. However, the updated pages are available only in Enhanced view. Standard view hasn't been updated from previous releases.

Change to the new UI for a single user

The system's default view settings can be changed to save settings for each user per each legal entity. To switch the view for a single user, the user must select the **Enhanced view** button on each page in the **Data management** workspace.

Change to the new UI for all users in all legal entities

1. In the **Data management** workspace, select the **Framework parameters** tile.
2. Change the **View default** setting from **Standard** to **Enhanced** for all users and legal entities.

Set file extensions for default data sources

The **Template** and **Import/export** pages let you add entities by selecting a file that was created by using the data management framework. We have added the following default file name extensions for the various types of data sources:

- .xlsx for Microsoft Excel data sources
- .zip for package data sources
- .xml for XML data sources
- .csv for comma-separated values (CSV) data sources
- .txt for delimited data sources

If you want to use other file name extensions, you must update your data sources so that they have a default file name extension that will be used as the target data format.

1. In the **Data management** workspace, select the **Configure data sources** tile, and then select **Data sources**.
2. Add a default file name extension to the appropriate data source.

Copy configuration data between companies or legal entities overview

2/18/2021 • 32 minutes to read • Edit Online

There are two options for copying configuration data in Finance and Operations:

- To move data between instances, you must first export it from one company and then import it to another company.
- To move data from one legal entity to another legal entity in the same instance, you can use the **Copy into legal entity** feature.

Export a configuration

The **Data management** workspace is your hub for managing configuration data projects and exporting data packages. To build a configuration, you must define a data project and export the information that is represented by entities.

To create an export configuration data project, follow these step.

1. Open the **Data management** workspace. If you're in Standard view, select **Enhanced view**.
2. Select the **Export** tile.
3. Select **New** to create an export configuration data project, and enter an ID and name for the configuration.
4. Set the operation type for the data project to **Export**, and set the project category to **Configuration**.
5. Add the entities that represent the information that you want to export. You can add entities by using several methods:
   - **Add one entity** – Enter the first part of the name of the entity until it appears in the lookup.
   - **Add multiple entities** – Enter any part of the entity name, use the lookup for the module, enter any part of the tag name, or use the lookup for the entity category to show a list of entities. Press Tab to move focus away from the **Lookup** field and activate the filter. In the grid, select the entities to add.
   - **Add a file** – Browse to a file that contains a name that matches the name of an entity and a file name extension that matches the file name extension that is in your data sources.
   - **Add a template** – Select from a list of templates that you've loaded in your instance.
6. Select a target data format. The system stores the last data format that you selected. Alternatively, if you select a file, the system automatically sets the data format to the data source that matches the file name extension.

   > **NOTE**
   > Composite entities require XML format.

7. Select **Add**. If you load a template, and the project already includes an entity that matches an entity in the template, the entity in the project will be replaced by the entity in the template. Some templates are very large, and they might take a few seconds to be loaded.
8. Select **Remove entity** to remove one or more selected entities.

9. When you've completed the configuration, select **Export** to start the export. You can monitor the results on the **Execution summary** page that appears.

Before you export a configuration, you might want to use some additional features that can help control the export process:

- To organize the list, use the **Sort by** button to reorder the entities by unit, level, and sequence.
- To change the execution sequence of any of the entities, you can manually edit the unit, level, or sequence. Alternatively, you can use the **Resequence** button to update any entities that you've selected. The **Resequence** button appears only if you select more than one entity. You can change the unit, level, and sequence individually. Alternatively, you can enable multiple changes and make them all at the same time. If you want the unit, level, or sequence to remain unchanged when you change multiple parts of the sequence, set the increment to **0** (zero).
- To add filters to the entity, use the **Filter** button. If you add a filter, the **Filter** button changes to an **Edit** button. The data will be filtered before it's exported. If you've added a template, and the template includes filters, those filters will be added to your project. However, you can modify or remove them as you require.
- If you must change the entity mappings, use the **View map** button. If you've added a template, and the template includes mapping changes, those changes will be applied to your project. However, you can modify them as you require.
- To temporarily prevent the entity from being used when you export a data project, use the check box in the **Disable** column.
- To open the contents of the grid in a Microsoft Excel workbook, use the **Open in Excel** button. Modify the entities as you require, and then use the **Publish** button to upload the changes back.

When the export is completed, complete the following tasks:

- Use the **Download** button on the **Export** page to download the configuration settings.
- Use the **Download package** button in the **Data management** workspace or on the **Execution summary** page to download the configuration settings and the data that was exported.

Setup considerations for some entities that are used to export configurations

Currently, several entities require additional steps when you build configurations. Follow these recommendations as you build your configurations.

> **IMPORTANT**
> This list will be updated as the Copy configuration feature is improved.

Using special-purpose entities

The following entities require special handling when they are used in configurations.

AREA	ENTITY	ACTION
System setup	Global address book	The entity no longer exports the records that are created automatically when a company is created. The import no longer accepts those records as well, starting in the platform 9 release.

AREA	ENTITY	ACTION
GL Shared	Account structures active group	This composite entity will export and import only the active account structures. If you use any other account structure entities, the status of the active account structures will be changed to **Draft**, and you must activate them before they can be used.
	Advanced rule structures active group	This composite entity is used in combination with the account structures active group entity. It will export and import only the active advanced rule structures. If you use any other advanced rule structure entities, the status of the advanced rule structures will be changed to **Draft**, and you must activate them before they can be used.
	Financial dimension values	All dimension values will be exported, even values that are based on system-defined entities such as projects or customers. Remove the system-defined values before you import them. If you leave the system-defined values in the package, they won't be imported. However, they will be filled as you import the data that backs the system-defined dimension.
Workflow	Workflow version	Change the owner for every record in the package data to **Admin**, unless the users in the workflow are already imported.
	Workflow expression	Some workflow expressions might be too long for an Excel cell. Use XML instead of Excel as the export format.
Tax	Sales tax parameters	The default value for the marginal base calculations method is **Total** for sales tax parameters. The Ledger Parameters entity doesn't set that value. However, the marginal base that some tax codes use, **Line**, will fail validation. A new entity that is named the Sales tax parameters preset entity was created so that you can import the marginal base calculation method first. You can then import tax codes.

AREA	ENTITY	ACTION
Accounts receivable	Customers	The Customers entity was designed to be used for OData scenarios. For configurations, use the Customer definitions entity and the Customer details entity. The Customer definitions entities let you import the basic information about a customer. In this way, you enable entities that require that a customer have that information earlier in the import process. The Customer details entity contains additional information about a customer that you can add after parameters and reference data have been set up.
Inventory management	Warehouse locations	Some warehouse locations require a location profile ID. Location profile IDs require a location format. Currently, the location format information must be manually added before the warehouse location. The entities for the location format and location profile were added in version 7.2.3, (App update 3 of the July 2017 release).
Product information management	Products	The Products and Released Products entities should be used for configurations. The Product master and Released product master entities should be used for OData scenarios.
	Product document attachments	For attachments to product documents released product documents, you must never skip staging, because additional steps are performed in the staging environment. You must use a data package for export and import, because the export file must be accompanied by a resources folder that contains the attachments. The entities support images, documents, notes, and links. When you export, you will see an image file that has a name that resembles a globally unique identifier (GUID). This file is a valid data package that is required in order to complete the import.
	Product attribute values	Product attribute values are assigned only when a user opens the **Attribute values** page from the **Products details** page. Currently, you can't import the values unless this step was performed in the golden build.

AREA	ENTITY	ACTION
Procurement	Vendor catalog	See the "Vendor catalogs import" section of Vendor catalogs in Dynamics AX on the Supply Chain Management blog.
Project	Shared category	The Shared category entity now exposes the following fields: **CATEGORYID**, **CATEGORYNAME**, **EXPENSETYPE**, **USINEXPENSE**, **USINPRODUCTION**, and **USEINPROJECT**. If you change the value of the **USEINEXPENSE** field to **yes**, the **EXPENSETYPE** should be set to one of the valid values that are available in the **Expense type** field on the **Shared category** page.

Remove the mapping and apply filters for specific entity fields

In a golden build, customer-specific fields might not be set up. To help guarantee that the import works, you should unmap those fields. For example, workers are stored in many tables, but they might not be set up in a golden build. Filters might also be required for some fields in an entity.

The following entities might have to be unmapped or filtered.

AREA	ENTITY	ACTION
System setup	Operating unit	Unmap Manager personnel number unless workers have been imported.
	User information	Apply a filter where **ID** isn't equal to **Admin**. Unmap Person name, and use the User to person relationship entity to map system users to directory users.
Accounts payable	Vendors	Unmap Purchase site (DefaultPurchaseSite) and Warehouse (DefaultProcurementWarehouseID) unless they are set up. Unmap the vendor bank account ID. The Vendor bank account entity will set up the link to the bank account when it's imported.
Accounts receivable	Customer details	Unmap Employee responsible number unless workers have been imported. Unmap Collections contact person (CollectionsContactPersonID) unless workers and their contact information have been imported. Unmap the site (SiteID) and warehouse (WarehouseID) unless they have already been imported.
Inventory management	Warehouse current postal address	Unmap Picking store area and Input store area unless Commerce information has been imported.

AREA	ENTITY	ACTION
Product information management	Products	Unmap NMFCCode and STCCCode unless you are adding the transportation management template to your data project.
	Released products	Unmap the project category, default product color, default configuration, default product size, and default product style. This entity is self-referencing and hasn't yet been updated to load these fields in a single pass.
	Period template	The Period template entity is a shared entity. Although it can be filtered by legal entity, the Period template lines entity doesn't have a **Legal entity** field. To import a single legal entity, you can filter the period template. However, you must currently remove the period template lines that aren't related to that legal entity.
	Item coverage group	Unmap Period template ID unless it has already been added manually.
Procurement	Vendors	Unmap Purchase site (DefaultPurchaseSite) and Warehouse (DefaultProcurementWarehouseID) unless they are set up. Unmap the vendor bank account ID. The Vendor bank account entity will set up the link to the bank account when it's imported.
Sales and marketing	Leads	Unmap LeadOpeningPersonnelNumber, LeadClosingPersonnelNumber, and LeadResponsiblePersonnelNumber unless workers have been imported.
	Sales type document entry policies	Unmap IsAtpGenrallyIncludingPlannedOrders. The default for Master planning was changed to be Yes for Disable all planning processes when a legal entity is created.
Project management	Projects	Unmap WorkerArchitectPersonelNumber, WorkerRespFinancialPersonelNumber, WorkerResponsiblePersonnelNumber, and WorkerRespSalesPersonelNumber unless workers have been imported.

AREA	ENTITY	ACTION
Retail	POS visual profiles	Unmap Pallet because no entity is available at this time. The POS visual profiles entity was added to the Retail template in version 7.2.3, (App update 3 of the July 2017 release).
	Retail channel	Unmap Channel profile name (ChannelProfileName) and Live database connection profile name (LiveDatabaseConnectionProfileName) because no entity is available at this time. The Retail channel entity was added to the Retail template in version 7.2.3, (App update 3 of the July 2017 release).

Golden builds that have multiple legal entities

The templates can be used to export data from any company in your golden build. When you export both shared data and company data, the templates first export the shared data for all legal entities. They then export the data for the legal entity that you're currently using. You can then switch companies and export the company data for additional legal entities by using projects that don't include the shared entities.

If you're exporting from a golden build that contains multiple legal entities, but you want to import the data from only one of those legal entities, you must apply a filter on the legal entity fields, so that only the data that you require for that legal entity is exported. This filter must remove all data for all legal entities except the legal entity that you want. In some cases, you must complete additional steps to clean up the exported data.

Most of the changes that are listed in the following table occur in the **System setup** and **General ledger** areas. If you export a golden build that uses a single legal entity, you should not require these filters.

The following entities require filters or special handling when you export the data.

AREA	ENTITY	ACTION
System setup	Legal entities	Apply a filter to Company.

AREA	ENTITY	ACTION
	Number sequence code	The number sequence codes can be shared, or they can be specific to a legal entity. To import all number sequences, you must have the legal entities setup for the number sequences that are stored for a specific legal entity. If you want only shared number sequences, apply a filter to remove the number sequences that are specific to a legal entity. If you want the number sequences only for a specific legal entity, apply a filter to Company. Number sequences can also have scope. You must delete number sequences when the following conditions are met: • The **SCOPETYPE** value is **DataArea**, and the **SCOPEVALUE** value either isn't equal to the legal entity or is blank. • The **SCOPETYPE** value is **LegalEntity**, and the **SCOPEVALUE** value isn't equal to the legal entity. • The **SCOPETYPE** value is **DataAreaFiscalCalendar**, and the **SCOPEVALUE** value isn't equal to the legal entity.
	Number sequence references	Number sequence references can also have scope. You must delete the number sequence references when the following conditions are met: • The **SCOPETYPE** value is equal to **DataArea**, **DataAreaFiscalCalendar**, or **LegalEntity**. • The **SCOPEVALUE** value either isn't equal to the legal entity that you want in the data or is blank.
	Organization hierarchies	There is no legal entity filter. Manually remove any references to the other legal entities that you aren't importing. For example, if you've set up centralized payments, but you're loading only one legal entity, you must not import the Centralized payments hierarchy.

AREA	ENTITY	ACTION
Global address book	Multiple entities	The global address book contains data for all legal entities. All legal entities will be created when you import the data, unless you remove the data for the legal entities that you don't want to load. To help guarantee that other legal entities aren't created, delete all records for those other legal entities. Alternatively, apply a filter to the Legal entity data area (<> blank). You must also remove global address book records that are used in the other legal entities.
	Party postal addresses	Address records for legal entities other than the legal entity that you're importing must be manually deleted before import.
	Party contacts	Contact records for legal entities other than the legal entity that you're importing must be manually deleted before import.
	Workflow	Apply a filter on DataAreaID. Many of the workflow entities can't be filtered for legal entity. Therefore, import failures might occur if you don't remove the data in all workflow entities that are related to legal entities that you don't want. We recommend that you manually set up workflows for this scenario.
General ledger	Ledger	Apply a filter to Company.
	Ledger fiscal calendar year	Apply a filter to Ledger name.
	Ledger fiscal calendar period	Apply a filter to Ledger name.
	Main account legal entity overrides	Apply a filter to Company.
	Financial dimension value legal entity	Apply a filter to Legal entity.
	Financial dimension values	Apply a filter to Legal entity. However, you can't filter out system-defined dimension values. Therefore, you must delete them. Those system-defined dimension values will be restored when you import data for the tables that back the system-defined dimensions.
	Journal names	Apply a filter to Voucher series company ID.
	Ledger allocation basis source	Apply a filter to Legal entity.

AREA	ENTITY	ACTION
	Ledger allocation rule destination	Apply a filter to Company.
Accounts receivable	Customer write-off reason code	Apply a filter to Company.
Budget	Budget control configuration	Apply a filter to Legal entity.
	Budget control configuration activation	Apply a filter to Legal entity.
	Budget control cycle model	Apply a filter to Legal entity.
	Budget control dimension attribute	Apply a filter to Legal entity.
	Budget control documents and journals	Apply a filter to Legal entity.
	Budget control group	Apply a filter to Legal entity.
	Budget control group criteria	Apply a filter to Legal entity.
	Budget control message level	Apply a filter to Legal entity.
	Budget control over budget permissions	Apply a filter to Legal entity.
	Budget control rule	Apply a filter to Legal entity.
	Budget control rule criteria	Apply a filter to Legal entity.
	Budget cost elements	Apply a filter to Cost element data area ID.
	Budget dimensions	Apply a filter to Legal entity.
	Budget plan allocation schedule	Apply a filter to Ledger.
	Budget plan process	Apply a filter to Ledger.
	Budget plan stage rule	If you applied a filter to Budget plan process, errors might occur when you import stage rules. Because the entity doesn't currently contain a ledger name that can be filtered, it will contain all companies.
	Budget plan priority constraint	You will experience the same issue that is described for the Budget plan stage rule entity.
	Budget plan process administration	You will experience the same issue that is described for the Budget plan stage rule entity.

AREA	ENTITY	ACTION
	Budget transfer rules	Apply a filter to Legal entity.
Inventory management	Warehouse current postal address	Apply a filter to Company.
	Site current postal address	Apply a filter to Company.
	Tracking number groups	The entity automatically filters the Number sequence scope data area by the legal entity. Therefore, you don't require a filter. However, if you must change the legal entity, the legal entity is stored in the table.
Retail	POS registers	Apply a filter to Legal entity. This entity was added to the Retail template version 7.2.3, (App update 3 of the July 2017 release).
	Retail store address book	There is no legal entity filter for this entity so the export will include records for all legal entities. This entity was added to the Retail template inversion 7.2.3, (App update 3 of the July 2017 release).
	Retail locator group member	There is no legal entity filter for this entity so the export will include records for all legal entities. This entity was added to the Retail template in version 7.2.3, (App update 3 of the July 2017 release).
	Retail locator group owner	There is no legal entity filter for this entity so the export will include records for all legal entities. This entity was added to the Retail template in version 7.2.3, (App update 3 of the July 2017 release).
	Retail devices	There is no legal entity filter for this entity so the export will include records for all legal entities. This entity was added to the Retail template in version 7.2.3, (App update 3 of the July 2017 release).

Changing the legal entity value before import

If you want to change the legal entity ID to another value, the value of all fields that resemble the fields that were listed earlier must be changed to the value of the new legal entity. For example, for Legal entities, change Company from the exported value to a new value in the exported file.

The legal entity ID is stored in many places. Therefore, it can be difficult to make this change, and you might cause errors if you try.

> **NOTE**
>
> When you set up a data project to copy into a legal entity, a legal entity filter for the source legal entity is automatically added to any entity field that is determined to be a legal entity field.

Selecting a single legal entity for export

To export a single legal entity, you can create a Copy into legal entity data project and specify the legal entity to copy as the source legal entity. When you add entities or load templates, that type of project will automatically add legal entity filters. You can then download the package or create a template from it on the **Templates** page. The package or template can then be added to an export project and used to export the legal entity.

Import a configuration

The **Data management** workspace is also your hub for importing configuration data projects. You can build a configuration project from an existing data project that you exported. Alternatively, you can build a configuration project from individual files that contain data that is formatted correctly for import.

To import a configuration, follow these steps.

1. Open the **Data management** workspace. If you're in Standard view, select **Enhanced view**.
2. Select the **Import** tile.
3. Select **New** to create a configuration data project, and enter an ID and name for the configuration.
4. Set the operation type for the data project to **Import**, and set the project category to **Configuration**.
5. Add the entities that represent the information that you want to copy. You can add entities by using several methods:
   - **Add one entity** – Enter the first part of the name of the entity until it appears in the lookup.
   - **Add multiple entities** – Enter any part of the entity name, use the lookup for the module, enter any part of the tag name, or use the lookup for the entity category to show a list of entities. Press Tab to move focus away from the **Lookup** field and activate the filter. In the grid, select the entities to add.
   - **Add a file** – Browse to a file that contains a name that matches the name of an entity and a file name extension that matches the file name extension that is in your data sources, and the source data format will be set automatically. If you haven't set up the default file name extensions, you must select a source data format before you select the file.
   - **Add a template** – Select from a list of templates that you've loaded in your instance.

   When you load a package, the **Import** page first reads the list of entities from the package. A progress indicator shows how much of the package has been read. After the list of entities is read, the **Import** page starts to load the data in the package. This process can take some time.
6. Select **Remove entity** to remove any selected entities, as required.
7. After you've completed the configuration, select **Import** to start the import. You can monitor the results on the **Execution details** page that appears.

Before you import a configuration, you might want to use some additional features that can help control the import process:

- To organize the list, use the **Sort by** button to reorder the entities by unit, level, or sequence.
- To change the execution sequence of any of the entities, you can manually edit the unit, level, or sequence. Alternatively, you can use the **Resequence** button to update any entities that you've selected.
- If you must change the entity mappings, use the **View map** button.

- To temporarily prevent the entity from being used when you export a data project, use the check box in the **Disable** column.

Copy into a legal entity

The **Data management** workspace is also your hub for copying configuration information from one legal entity to another. The process resembles an export and import that occur in one step. As in an import, if information that exists in the source legal entity doesn't exist in the destination legal entity, the copy process adds it. If information already exists in the destination legal entity, the copy process updates it.

To copy a configuration from one legal entity to another legal entity in the same instance, follow these steps.

1. Open the **Data management** workspace. If you're in Standard view, select **Enhanced view**.
2. Select the **Copy into legal entity** tile.
3. Select **New** to create a configuration data project, and enter an ID and name for the configuration.
4. Set the operation type for the data project to **Copy into legal entity**, and set the project category to **Configuration**.
5. Select the legal entity that should be the source of the data to copy. By default, the legal entity that you're currently using is selected.
6. On the **Legal entities** FastTab, you can select existing legal entities as a destination, or you can create new legal entities:
   - **Select** – Select one or more legal entities in the list, and then select **Add selected**. The legal entities are added to the list of destination legal entities.
   - **Create** – Enter the legal entity ID, the legal entity name, and the region that the legal entity belongs in. Then select **Create legal entity**. The legal entity is created and added to the list of destination legal entities.

   > **NOTE**
   > The functionality for creating destination legal entities is available in Finance and Operations 7.2.3.

7. After you've added the destination legal entities, select **Yes** if the number sequences should be copied. The entities that are required in order to copy the number sequence codes and number sequence references will be added to the project. The execution unit, level, and sequence number for these entities are set to the numbers in the default System and Shared templates. If you aren't using the default templates, adjust the entity sequences so that they are first in the list.
8. If you selected **Yes** for number sequences, select **Yes** or **No** to specify whether those number sequences should be reset to the smallest value.
9. Add the entities that represent the information that you want to copy. You can add entities by using several methods:
   - **Add one entity** – Enter the first part of the name of the entity until it appears in the lookup.
   - **Add multiple entities** – Enter any part of the entity name, use the lookup for the module, enter any part of the tag name, or use the lookup for the entity category to show a list of entities. Press Tab to move focus away from the **Lookup** field and activate the filter. In the grid, select the entities to add.
   - **Add a file** – Browse to a file that contains a name that matches the name of an entity and a file name extension that matches the file name extension that is in your data sources.
   - **Add a template** – Select from a list of templates that you've loaded in your instance.

To help guarantee that the correct order is maintained, we recommend that you use the default templates. Then add and remove entities to match the data that you want to copy. You can remove the entities that you don't want to copy.

> **NOTE**
>
> - If an entity has a field that represents the legal entity, a filter will be applied to that entity, so that only the data for the source legal entity is included. The value for that field will be changed to the destination legal entity.
> - Document, transaction, and composite entities aren't available when you copy to a legal entity.

10. Select **Remove entity** to remove any selected entities, as required.
11. After you've completed the configuration, select **Copy into legal entity** to start the import. The copy process will export the data from the source legal entity into the destination legal entity. Each destination legal entity will have its own import data project. You can monitor the results on the **Execution summary** page that appears. All import projects that are related to the Copy into legal entity project will appear in a list on the left of the page.

Any errors that occur are shown on the **Execution summary** page, just as they are for an import project. You can edit the errors in the staging tables and resubmit the values for each data project.

Special considerations when you copy into a legal entity

When you copy into a legal entity, you have the same validation that occurs when you import a file. It's important that you test your copy in a test environment, so that you can identify any dependencies that will cause failures. If dependent information isn't included in your list of entities to copy, the entity will show errors when it tries to copy into the legal entity. For example, if a customer has a default site or warehouse, you must use one of these approaches:

- Import the sites and warehouses as part of the copy.
- Manually load the sites and warehouses before you copy the legal entity.
- Unmap the site and warehouse fields before you copy the information.

You might also experience import errors if you copy from one region to another region. For example, you can have 1099 fields in a legal entity in the US region. However, if you try to import those values into a legal entity in a German region, you will see errors on the import. If a template that you load has entities that are incorrect for a region, you will receive an error message that states that the incorrect entity wasn't loaded. However, the rest of the template will continue to be loaded. You should copy only information that is appropriate for the destination region.

The following entities require special handling when they are used to copy into a legal entity.

AREA	ENTITY	ACTION

AREA	ENTITY	ACTION
System setup	Number sequences	If you use a number sequence for vendors and customers on the parameters forms and then you copy customers and vendors, you need to make sure that the "Allow user to change to a lower number" settings on the number sequences to Yes. The "No" settings will cause the import to reject vendor and customer numbers since they were created before using numbers lower than the next available number sequence. If you use the Reset to smallest feature, you need to change the "Allow user to change to a higher number" since the vendor and customer numbers are higher than the next available number sequence.
System setup	Workflow	Workflow requires additional changes before it can be copied. Workflow copies are not supported at this time.
Accounts payable	Vendors	Vendors have many settings that are dependent on the values that come from other entities. For example, if you update the matching settings to require three-way matching but you have vendors set for two-way matching, the vendor will fail validation. If you use auto sequencing for new vendor, you will need to unmap the vendor account before you do the copy into legal entity. In addition, if an auto-sequenced vendor has an invoice account, that vendor account is not transformed and may fail. You may see similar issues in other areas such as vendors in posting accounts and approved vendor list by products.
Accounts receivable	Customers	Customers have many settings that are dependent on the values that come from other entities. For example, if you have a default warehouse and site for a customer, you must add sites and warehouses first or the customer will fail validation. The collections contact will also fail if the contact is not available in the new company. If you use auto sequencing for new customers, you will need to unmap the customer account before you do the copy into legal entity. In addition, if an auto-sequenced customer has an invoice account, that invoice account is not transformed and may fail. You may see similar issues in other areas such as customers in posting accounts.

AREA	ENTITY	ACTION
Budget	Budget cost elements	There is an issue when importing budget cost elements when using an annual amount and there are budget cost elements in the Earnings basis tab. The issue will be addressed in a future release.
Fixed assets	Fixed assets depreciation profile manual schedule	The fixed asset depreciation profile must be processed first. Change the sequence on your data project for fixed asset depreciation profile to 15 (instead of 10). We will update the default templates in the monthly application release 4 to this value.
General ledger	Intercompany accounting	The copy into legal entity feature does not support intercompany accounting at this time. The issue will be addressed in a future release.
General ledger	Journal names	Only the journal names for the source legal entity will be copied to the destination companies. If you select the lookup in the journal names form, you will only see the number sequences that are available for that legal entity. However, if you choose to enter a number sequence manually that is not on that list, it will not be included in the copy process.
General ledger	Ledger allocation rules destination	If you are running a version earlier than version 7.3, if you have cross company allocation rules, you will not see the destination rules for legal entities that do not match the source legal entity. The copy into legal entity feature will only copy the destination records when the destination is equal to the source. The issue has been resolved in version 7.3.
General ledger	Ledger fiscal calendar year/period	The process is currently exporting all of the legal entities instead of just the source legal entity. The copy process works correctly. The issue has been resolved in version 7.3.

AREA	ENTITY	ACTION
General ledger	Ledger parameters	If you check for continuous number sequences but you have number sequences that are used on journal names and are not continuous, the imports will fail. You should temporarily turn off that setting in the ledger parameters. In addition, the ledger parameters must be processed first. Change the sequence on your data project for ledger parameters to 15 (instead of 40). We will update the default templates in the monthly application release 3 to this value.
Inventory management	Inventory dimension parameters	There is an issue where an error on import is shown but the correct number of parameters are imported. The issue will be addressed in a future release.
Inventory management	Warehouse locations	Some warehouse locations require a location profile ID. Location profile IDs require a location format. Currently, the location format information must be manually added before the warehouse location. The entities for the location format and location profile were added in version 7.2.3, (App update 3 of the July 2017 release).
Master planning	Intercompany master plan associations	The copy into legal entity feature does not support intercompany master plan associations at this time. The issue will be addressed in a future release.
Retail	POS registers	This entity is global and cannot be copied to another legal entity.
Retail	Retail channel	This entity is global and cannot be copied to another legal entity.
Retail	Retail store address book	This entity has a dependency on Retail channel so it cannot be used.
Sales and marketing	Intercompany trading partnerships	The copy into legal entity feature does not support intercompany trading partnerships at this time. The issue will be addressed in a future release.

Rules

The Copy into legal entity feature supports rules, which let you modify data before it's added to the import staging table. For example, rules are used to change the target legal entity to the destination legal entity and modify number sequences. You can extend rules.

Rule extensions require three changes:

1. Add the name of the rule to the extensible **DMFRulesType** enumeration (enum).

2. For each project definition group, insert the enum that you just created (for example, **DMFRulesType::NewRule**) into the DMFRules table. Use your source legal entity, your rule type, and the definition group name. If you require that more data be stored, such as various options for your rule, you can create your own table and extend the DMFRules table.
3. Create your own class to handle the data that the rule acts on. For the DMFRules framework to instantiate the rule, you must decorate the class with the **[DMFRulesBaseFactoryAttribute(DMFRulesType::NewRule)]** attribute.

The class must also extend the **DMFRulesBase** class. This extension will require an implementation of the **DMFRulesBase.runRule(Common_staging)** method. The _staging record will be the buffer of the staging record that the rule is applied to.

Additional information about entities

Obsolete entities

As the application is updated, the functionality of an entity might have to be updated. A new entity might be created that has a different name. In this case, the original entity will be marked as obsolete. You will no longer be able to add obsolete entities to a new data project or template.

If you load a data package that contains an obsolete entity, you will receive a warning about the existence of obsolete entities. However, you will still be able to import your data. You can find the obsolete entity by selecting the **Obsolete** column and filtering on **Yes**.

Self-referencing entities

Some entities represent tables that have references to themselves. For example, when you create a cash discount, you can refer to a related cash discount that creates a tiered discount calculation. To import data, you must sequence the data so that the cash discount that is referred to in the **Next cash discount** field is imported before the cash discount that uses that cash discount.

A new **DMFImportExportSequencer** class sequences the data in self-referencing entities and enables the data to be loaded in a single pass. In the Cash Discount entity (CashDiscountEntity), you can view the code that is required in order to update entities.

The class has been added to several self-referencing entities. It will be added to more entities as required. Here are some other examples of self-referencing entities:

- Customers
- Customer definitions
- Customer details
- Tax codes
- Budget control groups
- Projects
- Product categories
- Warehouses
- Budget plan workflow stage
- Sales units

Adding entities in the appropriate country/region context

When you add entities, the mappings are created in the context of the country or region of the company that is currently active. If there are any issues with the mappings, a red *X* appears in the **View map** column. Select the red *X*, and repair the mappings as required.

NOTE

By default, the DAT company doesn't have a country/region context. Some entities, such as the entities that are used for transaction codes and 1099 fields, won't be mapped correctly if they are added to a data project for the DAT company, because a country/region context is expected.

Configuration data packages

2/18/2021 • 7 minutes to read • Edit Online

IMPORTANT

This topic applies only to the July 2017 release of Microsoft Dynamics 365 for Finance and Operations. If you are running a later release, refer to the topic Copy configuration data between companies or legal entities overview.

Configuration data packages are available as process data packages from Microsoft Dynamics Lifecycle Services (LCS). These data packages can help improve the repeatability of implementations and accelerate the configuration.

Data packages contain configuration entity spreadsheets. These entity spreadsheets contain best practice data that you can use to create an initial golden build. The data entities in the data packages are also sequenced appropriately to help guarantee a successful single-click import of the data.

The entity spreadsheets include three types of data:

- **Business data** – The spreadsheet contains standard business data for a mid-sized trade or retail company. This data combines best practices and business standards that can be used as a starting point for your configuration.
- **Sample data** – The spreadsheet contains data that can be used as an example for business-specific data. This data can be imported and used as an example, but it must be changed for individual business practices.
- **No data** – The spreadsheet doesn't contain any data. Several areas of the product are unique to each business and its business practices. These areas must be configured specifically for the organization. These spreadsheets should be reviewed and updated for the organization as appropriate.

For more information about the type of data that is included in each entity spreadsheet in the data packages see the Data packages section of this topic. You can modify individual spreadsheets before you import the data packages, or you can import the data packages as they have been supplied and then update your data in the system.

Using configuration data packages

You can access configuration data packages from LCS. You can either apply them to an LCS environment, or download them so that you can manually import them.

1. Open your LCS project, and open the Asset library.
2. In the list of asset types, select **Process data package**.
3. Click **Import**.
4. Select the configuration data package.
5. Click **Pick**.

At this point, you can use the **Consume** function to apply the process data package to an LCS environment.

You can also download the individual data package files from the **Data package** area. Use the **Data management** workspace to import the data packages from LCS. For more information about how to import and export configurations, see Copy configuration data between companies or legal entities overview.

Special considerations

System setup

The System data package must be imported before any other data package. By default, the System data package creates a new legal entity that is named **ST01**. The data packages for the module areas depend on this legal entity.

General ledger

A generic chart of accounts is included in the configuration data packages. When this data is used as it's defined in the Main account entity spreadsheet, posting profiles across the system are filled with default posting data. If you change the main accounts that are used for the chart of accounts, you must also update the individual posting profiles and posting accounts for each area.

Data packages: System

010 – System Setup

	SPREADSHEET CONTENT		
Entity spreadsheet	**Business data**	**Sample data**	**No data**
Address and contact information purpose	X		
Address books			X
Address format lines	X		
Address format	X		
Address parameters	X		
Calendar			X
Cities	X		
Counties	X		
CountryRegions	X		
Currencies	X		
Districts	X		
Exchange rates		X	
Global address book parameters	X		
Global address book			X
Legal entities		X	
Name affixes	X		

	SPREADSHEET CONTENT		
Name sequences	X		
Operating unit		X	
Organization hierarchy purposes			X
Organization hierarchy type			X
Organization hierarchy			X
Party contacts			X
Party postal addresses			X
Party relationships			X
Postal codes	X		
Relationship types	X		
States	X		
System parameters	X		
Team types			X
Teams			X
Unit conversions	X		
Unit translations	X		
Units	X		
User groups			X
User information			X

Data packages: Financials

020 – GL Shared

	SPREADSHEET CONTENT		
Entity spreadsheet	**Business data**	**Sample data**	**No data**
Account structure activation		X	

	SPREADSHEET CONTENT		
Account structure allowed values		X	
Account structures		X	
Advanced rule criteria		X	
Advanced rule structure activation		X	
Advanced rule structure allowed values		X	
Advanced rule structures		X	
Advanced rules		X	
Chart of accounts	X		
Consolidation groups and accounts		X	
Dimension attribute activation		X	
Financial dimension format		X	
Financial dimension sets		X	
Financial dimension translations			X
Financial dimension value translations			X
Financial dimension values			X
Financial dimensions		X	
Fiscal calendar period		X	
Fiscal calendar		X	
Main account categories	X		
Main account	X		
Number sequence code		X	
Number sequence group			X

	SPREADSHEET CONTENT		
Number sequence references		X	

025 – GL

	SPREADSHEET CONTENT		
Entity spreadsheet	**Business data**	**Sample data**	**No data**
Accounts for automatic transactions	X		
Balance control accounts			X
Calendar			X
Date intervals	X		
Default descriptions		X	
Dimension parameters	X		
Document types	X		
Financial dimension value legal entity overrides			X
Financial reasons	X		
Journal descriptions	X		
Journal names	X		
Ledger allocation basis source			X
Ledger allocation basis			X
Ledger allocation rule destination			X
Ledger allocation rule source			X
Ledger allocation rule			X
Ledger fiscal calendar period		X	
Ledger fiscal calendar year		X	
Ledger parameters	X		

	SPREADSHEET CONTENT		
Ledger		X	
Main account legal entity overrides			X
Organization email template message			X
Organization email template			X
Subledger journal transfer rule	X		
Workflow organization parameters			X
Working times			X

100 – Bank

	SPREADSHEET CONTENT		
Entity spreadsheet	**Business data**	**Sample data**	**No data**
Bank accounts		X	
Bank facility groups			X
Bank facility types			X
Bank groups		X	
Bank parameters	X		
Bank posting profiles			X
Bank statement import methods for general journal			X
Bank transaction groups	X		
Bank transaction type	X		
Check layout		X	
Customer charge groups			X
Payment purpose codes			X
Reconciliation matching rule sets			X

	SPREADSHEET CONTENT		
Reconciliation matching rules			X
Transaction code mapping			X

120 – AP

	SPREADSHEET CONTENT		
Entity spreadsheet	**Business data**	**Sample data**	**No data**
Aging period definitions			X
Business segments			X
Business subsegments			X
Buyer groups			X
Cash discount			X
Delivery charges groups			X
Destination code			X
Expedite codes			X
Item charge groups			X
Line discount vendor groups			X
Line of business		X	
Modes of delivery			X
Multiline discount vendor groups			X
Payment calendar rule			X
Payment calendar			X
Payment day lines		X	
Payment schedule lines		X	
Payment schedule		X	
Price vendor groups			X

	SPREADSHEET CONTENT		
Procurement charge codes			X
Product categories			X
Product category hierarchies			X
Product category hierarchy roles			X
Product category hierarchy translations			X
Purchase pools			X
Sales tax groups			X
Terms of delivery			X
Terms of payment	X		
Total discount vendor groups			X
Vendor bank accounts			X
Vendor charges group			X
Vendor exception groups			X
Vendor groups			X
Vendor invoice form printing configurations			X
Vendor invoice policy rule types			X
Vendor ledger accounts	X		
Vendor parameters	X		
Vendor payment fee			X
Vendor payment format			X
Vendor payment method specification			X
Vendor payment method		X	

	SPREADSHEET CONTENT		
Vendor posting profile	X		
Vendor price tolerance groups			X
Vendor reasons	X		
Vendor, form parameters			X
Vendors			X

130 – Tax

	SPREADSHEET CONTENT		
Entity spreadsheet	**Business data**	**Sample data**	**No data**
Sales tax authorities		X	
Sales tax code limits			X
Sales tax code values		X	
Sales tax codes		X	
Sales tax exempt code	X		
Sales tax exempt numbers			X
Sales tax group details		X	
Sales tax groups		X	
Sales tax item groups		X	
Sales tax ledger posting groups	X		
Sales tax parameters preset entity			X
Sales tax parameters	X		
Sales tax period setup		X	
Sales tax reporting codes			X
Transaction codes			X
Withholding tax code values		X	

	SPREADSHEET CONTENT		
Withholding tax codes		X	
Withholding tax group details			X
Withholding tax groups		X	
Withholding tax limits			X

140 – AR

	SPREADSHEET CONTENT		
Entity spreadsheet	**Business data**	**Sample data**	**No data**
Aging period definitions	X		
Business segments			X
Business subsegments			X
Cash discount			X
Collection letter form printing configurations			X
Collection letter setup	X		
Commission calculation			X
Commission customer group			X
Commission item group			X
Commission sales group			X
Customer account statement form printing configurations			X
Customer bank accounts			X
Customer charge groups			X
Customer definitions			X
Customer details			X
Customer facing form printing configurations			X

	SPREADSHEET CONTENT		
Customer groups		X	
Customer ledger accounts	X		
Customer parameters	X		
Customer payment fee			X
Customer payment method specification			X
Customer payment method		X	
Customer posting profiles	X		
Customer reasons			X
Customer write-off reason codes	X		
Customers			X
Delivery charges groups			X
Expedite codes			X
Free text invoice form printing configurations			X
Interest codes with fees			X
Interest codes with ranges			X
Interest codes	X		
Interest note form printing configurations			X
Item charge groups			X
Line discount customer groups			X
Line of business			X
Modes of delivery			X
Payment calendar			X
Payment day lines			X

	SPREADSHEET CONTENT		
Payment schedule lines			X
Payment schedule			X
Price customer groups			X
Printed form notes			X
Sales charge codes			X
Sales districts			X
Sales order pools			X
Statistics group			X
Terms of delivery			X
Terms of payment			X
Total discount customer groups			X

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Configuration data packages

2/18/2021 • 7 minutes to read • Edit Online

IMPORTANT

This topic applies only to the July 2017 release of Microsoft Dynamics 365 for Finance and Operations. If you are running a later release, refer to the topic Copy configuration data between companies or legal entities overview.

Configuration data packages are available as process data packages from Microsoft Dynamics Lifecycle Services (LCS). These data packages can help improve the repeatability of implementations and accelerate the configuration.

Data packages contain configuration entity spreadsheets. These entity spreadsheets contain best practice data that you can use to create an initial golden build. The data entities in the data packages are also sequenced appropriately to help guarantee a successful single-click import of the data.

The entity spreadsheets include three types of data:

- **Business data** – The spreadsheet contains standard business data for a mid-sized trade or retail company. This data combines best practices and business standards that can be used as a starting point for your configuration.
- **Sample data** – The spreadsheet contains data that can be used as an example for business-specific data. This data can be imported and used as an example, but it must be changed for individual business practices.
- **No data** – The spreadsheet doesn't contain any data. Several areas of the product are unique to each business and its business practices. These areas must be configured specifically for the organization. These spreadsheets should be reviewed and updated for the organization as appropriate.

For more information about the type of data that is included in each entity spreadsheet in the data packages see the Data packages section of this topic. You can modify individual spreadsheets before you import the data packages, or you can import the data packages as they have been supplied and then update your data in the system.

Using configuration data packages

You can access configuration data packages from LCS. You can either apply them to an LCS environment, or download them so that you can manually import them.

1. Open your LCS project, and open the Asset library.
2. In the list of asset types, select **Process data package**.
3. Click **Import**.
4. Select the configuration data package.
5. Click **Pick**.

At this point, you can use the **Consume** function to apply the process data package to an LCS environment.

You can also download the individual data package files from the **Data package** area. Use the **Data management** workspace to import the data packages from LCS. For more information about how to import and export configurations, see Copy configuration data between companies or legal entities overview.

Special considerations

System setup

The System data package must be imported before any other data package. By default, the System data package creates a new legal entity that is named **ST01**. The data packages for the module areas depend on this legal entity.

General ledger

A generic chart of accounts is included in the configuration data packages. When this data is used as it's defined in the Main account entity spreadsheet, posting profiles across the system are filled with default posting data. If you change the main accounts that are used for the chart of accounts, you must also update the individual posting profiles and posting accounts for each area.

Data packages: System

010 – System Setup

	SPREADSHEET CONTENT		
Entity spreadsheet	**Business data**	**Sample data**	**No data**
Address and contact information purpose	X		
Address books			X
Address format lines	X		
Address format	X		
Address parameters	X		
Calendar			X
Cities	X		
Counties	X		
CountryRegions	X		
Currencies	X		
Districts	X		
Exchange rates		X	
Global address book parameters	X		
Global address book			X
Legal entities		X	
Name affixes	X		

	SPREADSHEET CONTENT		
Name sequences	X		
Operating unit		X	
Organization hierarchy purposes			X
Organization hierarchy type			X
Organization hierarchy			X
Party contacts			X
Party postal addresses			X
Party relationships			X
Postal codes	X		
Relationship types	X		
States	X		
System parameters	X		
Team types			X
Teams			X
Unit conversions	X		
Unit translations	X		
Units	X		
User groups			X
User information			X

Data packages: Financials

020 – GL Shared

	SPREADSHEET CONTENT		
Entity spreadsheet	**Business data**	**Sample data**	**No data**
Account structure activation		X	

	SPREADSHEET CONTENT		
Account structure allowed values		X	
Account structures		X	
Advanced rule criteria		X	
Advanced rule structure activation		X	
Advanced rule structure allowed values		X	
Advanced rule structures		X	
Advanced rules		X	
Chart of accounts	X		
Consolidation groups and accounts		X	
Dimension attribute activation		X	
Financial dimension format		X	
Financial dimension sets		X	
Financial dimension translations			X
Financial dimension value translations			X
Financial dimension values			X
Financial dimensions		X	
Fiscal calendar period		X	
Fiscal calendar		X	
Main account categories	X		
Main account	X		
Number sequence code		X	
Number sequence group			X

	SPREADSHEET CONTENT		
Number sequence references		X	

025 – GL

	SPREADSHEET CONTENT		
Entity spreadsheet	**Business data**	**Sample data**	**No data**
Accounts for automatic transactions	X		
Balance control accounts			X
Calendar			X
Date intervals	X		
Default descriptions		X	
Dimension parameters	X		
Document types	X		
Financial dimension value legal entity overrides			X
Financial reasons	X		
Journal descriptions	X		
Journal names	X		
Ledger allocation basis source			X
Ledger allocation basis			X
Ledger allocation rule destination			X
Ledger allocation rule source			X
Ledger allocation rule			X
Ledger fiscal calendar period		X	
Ledger fiscal calendar year		X	
Ledger parameters	X		

	SPREADSHEET CONTENT		
Ledger		X	
Main account legal entity overrides			X
Organization email template message			X
Organization email template			X
Subledger journal transfer rule	X		
Workflow organization parameters			X
Working times			X

100 – Bank

	SPREADSHEET CONTENT		
Entity spreadsheet	**Business data**	**Sample data**	**No data**
Bank accounts		X	
Bank facility groups			X
Bank facility types			X
Bank groups		X	
Bank parameters	X		
Bank posting profiles			X
Bank statement import methods for general journal			X
Bank transaction groups	X		
Bank transaction type	X		
Check layout		X	
Customer charge groups			X
Payment purpose codes			X
Reconciliation matching rule sets			X

	SPREADSHEET CONTENT		
Reconciliation matching rules			X
Transaction code mapping			X

120 – AP

	SPREADSHEET CONTENT		
Entity spreadsheet	**Business data**	**Sample data**	**No data**
Aging period definitions			X
Business segments			X
Business subsegments			X
Buyer groups			X
Cash discount			X
Delivery charges groups			X
Destination code			X
Expedite codes			X
Item charge groups			X
Line discount vendor groups			X
Line of business		X	
Modes of delivery			X
Multiline discount vendor groups			X
Payment calendar rule			X
Payment calendar			X
Payment day lines		X	
Payment schedule lines		X	
Payment schedule		X	
Price vendor groups			X

	SPREADSHEET CONTENT		
Procurement charge codes			X
Product categories			X
Product category hierarchies			X
Product category hierarchy roles			X
Product category hierarchy translations			X
Purchase pools			X
Sales tax groups			X
Terms of delivery			X
Terms of payment	X		
Total discount vendor groups			X
Vendor bank accounts			X
Vendor charges group			X
Vendor exception groups			X
Vendor groups			X
Vendor invoice form printing configurations			X
Vendor invoice policy rule types			X
Vendor ledger accounts	X		
Vendor parameters	X		
Vendor payment fee			X
Vendor payment format			X
Vendor payment method specification			X
Vendor payment method		X	

	SPREADSHEET CONTENT		
Vendor posting profile	X		
Vendor price tolerance groups			X
Vendor reasons	X		
Vendor, form parameters			X
Vendors			X

130 – Tax

	SPREADSHEET CONTENT		
Entity spreadsheet	**Business data**	**Sample data**	**No data**
Sales tax authorities		X	
Sales tax code limits			X
Sales tax code values		X	
Sales tax codes		X	
Sales tax exempt code	X		
Sales tax exempt numbers			X
Sales tax group details		X	
Sales tax groups		X	
Sales tax item groups		X	
Sales tax ledger posting groups	X		
Sales tax parameters preset entity			X
Sales tax parameters	X		
Sales tax period setup		X	
Sales tax reporting codes			X
Transaction codes			X
Withholding tax code values		X	

	SPREADSHEET CONTENT		
Withholding tax codes		X	
Withholding tax group details			X
Withholding tax groups		X	
Withholding tax limits			X

140 – AR

	SPREADSHEET CONTENT		
Entity spreadsheet	**Business data**	**Sample data**	**No data**
Aging period definitions	X		
Business segments			X
Business subsegments			X
Cash discount			X
Collection letter form printing configurations			X
Collection letter setup	X		
Commission calculation			X
Commission customer group			X
Commission item group			X
Commission sales group			X
Customer account statement form printing configurations			X
Customer bank accounts			X
Customer charge groups			X
Customer definitions			X
Customer details			X
Customer facing form printing configurations			X

	SPREADSHEET CONTENT		
Customer groups		X	
Customer ledger accounts	X		
Customer parameters	X		
Customer payment fee			X
Customer payment method specification			X
Customer payment method		X	
Customer posting profiles	X		
Customer reasons			X
Customer write-off reason codes	X		
Customers			X
Delivery charges groups			X
Expedite codes			X
Free text invoice form printing configurations			X
Interest codes with fees			X
Interest codes with ranges			X
Interest codes	X		
Interest note form printing configurations			X
Item charge groups			X
Line discount customer groups			X
Line of business			X
Modes of delivery			X
Payment calendar			X
Payment day lines			X

	SPREADSHEET CONTENT		
Payment schedule lines			X
Payment schedule			X
Price customer groups			X
Printed form notes			X
Sales charge codes			X
Sales districts			X
Sales order pools			X
Statistics group			X
Terms of delivery			X
Terms of payment			X
Total discount customer groups			X

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Configuration data templates

2/18/2021 • 9 minutes to read • Edit Online

Configuration data templates are predefined lists of entities for each module area that can be used in a data project. You can create, view, and modify these templates by using the **Template** page in the **Data management** workspace.

> **IMPORTANT**
>
> Default configuration templates available out-of-the- box will always have the latest version of an entity. Templates can be created from an existing data project as needed.

Create a new configuration data template

The **Template** page in the **Data management** workspace provides tools that let you create a template of entities. This page resembles the configurations page, and the two features work in a similar manner. You must use **Enhanced view** to take advantage of the new features.

To create a template, follow these steps.

1. Select **New** to create a template. Enter an ID and name for the template. Notice that the status is set to **Draft**.
2. Add or remove entities as you require.
3. Organize the list by using the **Sort by** button to reorder your entities by entity group, or by unit, level, and sequence.
4. To change the sequence of any of the entities, manually edit the unit, level, or sequence. Alternatively, use the **Resequence** button to update any entities that you've selected. The **Resequence** button appears only if you select more than one entity.
5. To add filters to an entity, use the **Filter** button. Then review the results of the filters by using the **Preview** button. If you add a filter, the **Filter** button is changed to an **Edit** button.
6. If you don't want all fields to be mapped, you can use the **View map** button to exclude fields from the mapping.
7. Select **Validate template** to change the status to **Validated**.

Your template can now be used in a project. However, you might want to use some additional features to control the export process.

- Use the **Open in Excel** button to open the contents of the grid in a Microsoft Excel workbook. Modify the entities as you require, and then select **Publish** to upload the changes back.
- If you exported a template and want to bring that template back, select **Import template**, browse to the template file, and then select **Create template** to load it.
- To replace the contents of an open template, select **Replace from template**, browse to the template file that has the entities to import, and then select **Create template** to load the template file. The values in the open template will be overwritten.
- To create a template from a project, select **New** to create a template. Enter an ID and name for the template, and then select **Replace template from project**. In the list of projects that appears, select a project, and then select **Create template** to bring the project entities from that project into the open template. The values in the open template will be overwritten.

Default data templates

In July 2017 update, we released predefined templates to help you create configuration data projects. The templates are sequenced, so that the data that the entities generate will be processed in the correct order. Our predefined templates are also designed to maintain the correct sequence when more than one template is added to the same data project. For more information, see the "Sequencing in the default templates" section.

Default templates are delivered together with each new release. Our long-term goal is to provide the templates in Microsoft Dynamics Lifecycle Services (LCS), so that you can push them to an instance. However, for the current releases, select the **Templates** tile in the **Data management** workspace, and then select **Load default templates** to load the templates. To see the **Load default templates** menu, you must use **Enhanced view**.

After the templates are loaded, you can change them to suit your business requirements. If you ever want to retrieve the original default templates, you can use the **Load default templates** button to add them back to your system. The templates will then be replaced with the latest versions. If you've made changes to the templates, you can make a copy of the old templates by exporting them.

Note that system administrator access is required in order to load default templates and import templates. This requirement helps guarantee that all entities are correctly loaded into the template.

How entities are sequenced for processing

Whether you're creating your own templates or using the default templates, it's important that you understand how templates are sequenced for processing during export and import.

Units, levels, and sequences

The unit, level, and sequence of an entity are used to control the order that the data is exported or imported in.

- Entities that have different units are processed in parallel.
- In the same unit, entities are processed in parallel if they have the same level.
- In the same level, entities are processed according to their sequence order in the level.
- After one level has been processed, the next level is processed.
- We use only unit 1 to help guarantee that all dependencies are handled in the correct order.

Entities are categorized by module name, entity category, and tag.

- The module represents the module that the entity is typically used for.
- The entity category represents the type of information in the category. For example, a category might include parameters or reference data.
- Tags provide additional details about the function of the entity in the feature area. For example, entities for the general ledger have **General ledger** as their module name. There are several tasks within the general ledger, and the tags represent those tasks. For example, **Setup** and **Journals** are tags that represent tasks that can be done for the general ledger.

Sequencing in the default templates

Our long-term goal for sequencing is to automatically sequence all the entities for every configuration. Until we reach that goal, for the entities in each of our default templates, we have created sequences that represent the dependency order between entities.

The following table shows how the templates were set up to handle dependencies. Note that the entities do **not** have to be processed in the order that we have sequenced them in. You can sequence them differently. However, you might inadvertently change the order that entities are processed in. If an entity requires data that hasn't been imported by another entity, you might receive errors because of missing dependent data.

MODULE	UNIT	LEVEL
System setup	1	10
Global address book	1	15
General ledger shared	1	20
Workflow	1	22
General ledger	1	25
Band 1 for dependencies	1	30
Band 2 for dependencies	1	40
Band 3 for dependencies	1	50
Band 4 for dependencies	1	60
Band 5 for dependencies	1	70
Band 6 for dependencies	1	80
Band 7 for dependencies	1	90
Bank	1	100
Accounts payable	1	120
Tax	1	130
Accounts receivable	1	140
Fixed assets	1	150
Budgeting	1	160
Inventory management	1	300
Product management	1	310
Procurement	1	320
Sales and Marketing	1	330
Quality management	1	395
Warehouse management	1	400
Transportation management	1	405

MODULE	UNIT	LEVEL
Production control	1	410
Process manufacturing	1	412
Costing	1	420
Retail (See the note.)	1	500
Expense management	1	600
Project accounting	1	650

> **NOTE**
> The Retail template is scheduled to be released in Finance and Operations, App update 3.

We reserved levels 10 through 22 for shared system entities, so that those entities are processed first. Almost all systems also use the company-specific general ledger entities. Therefore, we reserved level 25 for those entities. These levels represent the minimum basic setup that is required for most shared data in a configuration.

After the basic setup is completed, many entities can be loaded in parallel across all the modules. These entities don't have to be loaded in silos by module. Instead, we set up bands of dependencies between the data for different entities. We added the entities that have no dependencies to band 30. We then added band 40 for entities that have a dependency on the entities in band 30. We continued the process for bands 50 through 90.

After we organized the basic entities so that they can be processed in parallel, we organized the remaining entities by module, in the order that the modules should be processed in. However, many entities have many dependencies, some of which are complex. For example, the Vendor posting profiles entity might require Vendors or Items entities. Although the Vendor posting profiles entity is in the **Accounts payable** module, it must be processed after the **Product management** module. In that case, if Vendors entities are 1.130.10 and Items entities are 1.300.10, the Vendor posting profiles entity must be moved so that it's after that sequence (for example, 1.310.20).

The sequences that we have implemented are a guideline, not a requirement. There is no required relationship between a level and a module. You can rearrange the entities if the sequence doesn't work for your implementation. To add your own templates to a configuration, you can follow the preceding guidelines to help guarantee that your template is correctly merged into a project that uses default templates.

Templates that have the same entity

Some entities are required in more than one template. For example, you must have payment terms in both the Accounts Payable and Accounts Receivable templates. However, you might require only the Accounts Receivable template. We added the entity to both templates for situations where you require only one of them.

A data project can include only one instance of an entity. If you add a template, and the template contains an entity that already exists in a data project, the entity in that template replaces the entity that is currently in the project.

You can use this capability to override the default templates without changing them. For example, the **worker** field hasn't been mapped in your data project, but you have your own template that adds workers. In this case, you can build a template that includes the entities that have the **worker** field. In that template, you can map the **worker** field. Any entities in the data project that don't have the field mapped will then be replaced.

Merged templates

We have created larger templates that cover multiple module areas. You can use the larger templates or any combination of smaller templates to build a data project. The following combined templates are available:

- System and Shared, which include system setup, global address book, shared general ledger, and workflow
- Financials, which includes general ledger, bank, accounts payable, tax, accounts receivable, fixed assets, and budgeting
- Supply Chain Management, which includes inventory management, product management, procurement, sales and marketing, limited warehouse management, production control, and costing

The Expense and Project Management templates aren't included in a larger template. However, they are designed so that they can easily be merged into a project that uses other templates.

The Workers template includes the entities needed to add workers and re-map entities where the worker mapping was removed.

Master data

Many default templates include entities for master data such as customers, vendors, and released products. These entities are included to indicate the correct sequence of entities that you will require after you've loaded parameters and reference data. Master entities are most often sequenced in the module bands that are numbered 100 and above. In the grid, the entity category for these entities will be **Master**.

If you don't want to include master data in your configuration, remove those entities from your project.

Enable change tracking for entities

2/18/2021 • 2 minutes to read • Edit Online

Change tracking enables incremental export of data from Finance and Operations by using Data management. In an incremental export, only records that have changed are exported. To enable incremental export, you must enable change tracking on entities. If you don't enable change tracking on an entity, you can only enable a full export each time. Although, change tracking can be enabled for BYOD and non-BYOD scenarios, it must be noted that only for bring your own database (BYOD) use cases, change tracking also tracks deletes if the entity supports this. Deletion is tracked only for the root data source in the entity.

Enable change tracking for BYOD

You can enable change tracking when you publish one or more entities to a data store (BYOD).

1. In the **Data management** workspace, select **Configure entity export to database**.
2. Select the database to export data to, and then select **Publish**.

   You can publish one or more entities to your database. Select **Show published only** to see a list of entities that have previously been published.
3. Select an entity that is published, and then select **Change tracking**.
4. Select the appropriate option for change tracking for your environment.

   An entity can be modeled by using more than one table. The options let you specify the granularity at which changes can be tracked in an entity.

OPTION	HOW CHANGES ARE TRACKED
Enable primary table	Changes that are made to any fields in the primary table trigger a change in entity. Changes that are made to fields in secondary tables don't trigger a change in entity.
Enable entire entity	Changes that are made to any fields in any table in the entity trigger a change in entity.
Enable Custom query	Uses a custom query that identifies the tables on which changes must be tracked. The custom query is defined in the entity.

> **NOTE**
> If a change is triggered, the change is tracked on the entire record and not at the field level. The entire entity record is exported to the destination. Regardless of the option that you select, the number of fields in the entity is the number that is exported to the destination.

Enable change tracking for non-BYOD scenarios

For non-BYOD scenarios, change tracking can be enabled from the data entities list page in data management by selecting the entities and clicking **Change tracking**.

Custom query for change tracking

The following example shows how to add a static method to an entity. You must make sure that the method returns a query, and that the root node is the same as the entity. For example, for the Customer entity, the root node is custTable, and the change tracking query for it is also custtable.

- You must enable change tracking on the tables that are part of the query.
- Create a join between the entity and the change tracking query (on the root table) to determine which records have changed in the entity.

```
public static Query defaultCTQuery()
{
 Query q = new Query();

 QueryBuildDataSource custDs = q.addDataSource(tableNum(CustTable));

 QueryBuildDataSource partyDs = custDs.addDataSource(tableNum(DirPartyTable));
 partyDs.relations(true);

 QueryBuildDataSource locationDs = partyDs.addDataSource(tableNum(DirPartyLocation));
 locationDs.addRange(fieldNum(DirPartyLocation, IsPrimary)).value(queryValue(NoYes::Yes));
 locationDs.addLink(fieldNum(DirPartyTable, RecId), fieldNum(DirPartyLocation, Party));

 QueryBuildDataSource addressDs = locationDs.addDataSource(tableStr(LogisticsPostalAddress));
 addressDs.addLink(fieldNum(DirPartyLocation, Location), fieldNum(LogisticsPostalAddress, Location));

 return q;
}
```

Find information about standard data entities

2/18/2021 • 2 minutes to read • Edit Online

The application ships with many default data entities. Data entities are frequently updated, so for documentation, we rely on the data entity templates to indicate which order data entities should be imported in, and on reports for a list of data entities that ship with each release.

Configuration data packages

Configuration data packages on Microsoft Dynamics Lifecycle Services (LCS) contain configuration entity spreadsheets. Configuration entity spreadsheets contain best practice data that you can use to create an initial golden build of an implementation. The data entities in the data packages are also sequenced appropriately using an XML file to help guarantee a successful single-click import of the data. We recommend that you download and review the configuration data packages to understand how we recommend that you order your data imports. For more information, see Configuration data templates and Copy configuration data between companies or legal entities overview.

Reports

Microsoft provides the following reports for data entities, which can be downloaded from Technical reference reports:

- Aggregate data entities: Lists the aggregate data entities, and the fields that each contains.
- Aggregate measures: Lists the aggregate measures.
- Config keys: Lists the configuration keys.
- Config key groups: Lists the configuration key groups.
- Data entities: Lists each data entity. The report indicates the data source of the entity and the fields included in the entity. The report also indicates whether the data entity is public.
- Data entities fields: Lists each field in a data entity, and the table that it originates from.
- KPIs: Lists the KPIs.
- License codes: Lists the license code associated with each configuration key.
- Menu items: Lists the menu items associated with each configuration key.
- SSRS reports: Lists each report. The report indicates the data set used for each report, as well as the filters and fields available on each report.
- Tables: Lists each table and its table group.
- Workflow type: Lists each type of workflow. The report also describes what each type of workflow is used for and indicates whether the workflows of each type are associated with a specific company in the organization or with the whole organization.

Scripts

You can download the scripts to run these reports from fin-ops-doc-scripts.

Additional resources

Data entities overview

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Data templates with multiple worksheets

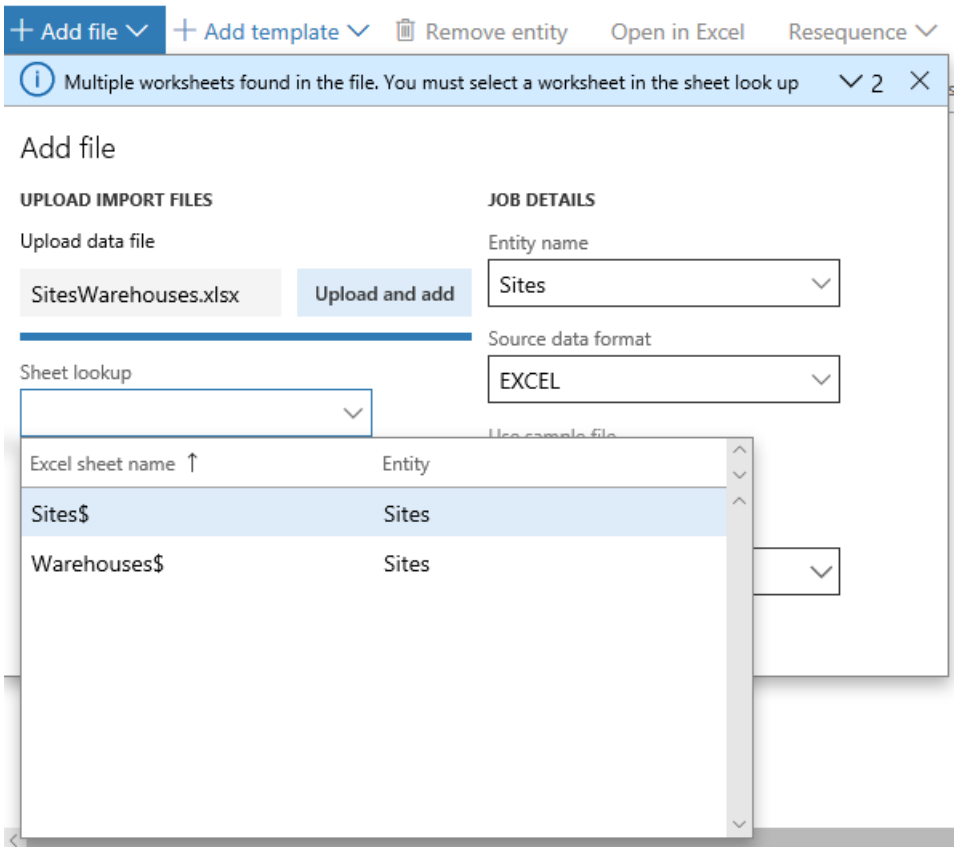
2/18/2021 • 2 minutes to read • Edit Online

Data management in the application supports Microsoft Excel-based templates for data entities. These templates can contain one or more worksheets. Templates with multiple worksheets are often used when it is convenient to manage data in a single file and import it to multiple data entities. An example would be sites and warehouses.

Upload a file once and map it to all entities

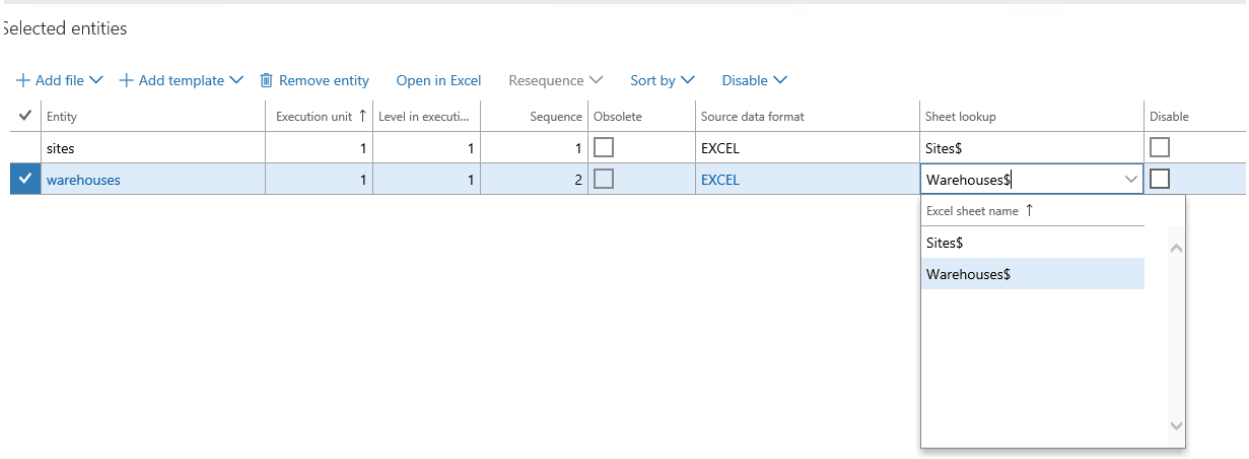
Let's take an example where there is one Excel file with worksheets called **Sites** and **Warehouses**. To set up the data import project, you would add the first data entity, **Sites** and then upload the file. You will be able to select **Sites** as the worksheet to be used for this entity.

If you add the second entity **Warehouses** without leaving the **Add file** form, the worksheet lookup will let you select the **Warehouses** worksheet without having to upload the file again. The only reason to upload a new file would be if the **Warehouses** data was in a different file.



Fix worksheet to entity mapping

The mapping of the worksheet to a data entity in the import job can be fixed from the grid. The **Worksheet** column in the grid shows the worksheets from the file that was mapped. You can choose a different worksheet from the drop-down menu. If the chosen worksheet is already mapped to an entity in the data project, the system asks you to confirm the change. We recommend that you fix all mappings in the grid.



Re-map to a new file

In cases where a new version of the same file or a completely new file must be uploaded for existing entities in a data project, you must use the **Add file** experience, and add the entities again as if they were being added for the first time. The system will confirm that you want to overwrite the existing entities in the data project before proceeding. Entities that are not added again (or overwritten) will continue to hold the previous mappings from the previous file.

Upload a file using Run project

You can upload an Excel file while using the **Run project** option to execute an import project. You must be careful to upload only files that have the same worksheets as the existing mappings on the data entities in the data project. If a worksheet is not found in the newly uploaded file, the system displays an error and will stop the import. If the mapping to the worksheet must be changed for an entity, then the mappings in the data project must be first updated from within the data project before using the file in the **Run project** experience.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Enable change tracking for entities

2/18/2021 • 2 minutes to read • Edit Online

Change tracking enables incremental export of data from Finance and Operations by using Data management. In an incremental export, only records that have changed are exported. To enable incremental export, you must enable change tracking on entities. If you don't enable change tracking on an entity, you can only enable a full export each time. Although, change tracking can be enabled for BYOD and non-BYOD scenarios, it must be noted that only for bring your own database (BYOD) use cases, change tracking also tracks deletes if the entity supports this. Deletion is tracked only for the root data source in the entity.

Enable change tracking for BYOD

You can enable change tracking when you publish one or more entities to a data store (BYOD).

1. In the **Data management** workspace, select **Configure entity export to database**.
2. Select the database to export data to, and then select **Publish**.

   You can publish one or more entities to your database. Select **Show published only** to see a list of entities that have previously been published.
3. Select an entity that is published, and then select **Change tracking**.
4. Select the appropriate option for change tracking for your environment.

   An entity can be modeled by using more than one table. The options let you specify the granularity at which changes can be tracked in an entity.

OPTION	HOW CHANGES ARE TRACKED
Enable primary table	Changes that are made to any fields in the primary table trigger a change in entity. Changes that are made to fields in secondary tables don't trigger a change in entity.
Enable entire entity	Changes that are made to any fields in any table in the entity trigger a change in entity.
Enable Custom query	Uses a custom query that identifies the tables on which changes must be tracked. The custom query is defined in the entity.

> **NOTE**
> If a change is triggered, the change is tracked on the entire record and not at the field level. The entire entity record is exported to the destination. Regardless of the option that you select, the number of fields in the entity is the number that is exported to the destination.

Enable change tracking for non-BYOD scenarios

For non-BYOD scenarios, change tracking can be enabled from the data entities list page in data management by selecting the entities and clicking **Change tracking**.

Custom query for change tracking

The following example shows how to add a static method to an entity. You must make sure that the method returns a query, and that the root node is the same as the entity. For example, for the Customer entity, the root node is custTable, and the change tracking query for it is also custtable.

- You must enable change tracking on the tables that are part of the query.
- Create a join between the entity and the change tracking query (on the root table) to determine which records have changed in the entity.

```
public static Query defaultCTQuery()
{
 Query q = new Query();

 QueryBuildDataSource custDs = q.addDataSource(tableNum(CustTable));

 QueryBuildDataSource partyDs = custDs.addDataSource(tableNum(DirPartyTable));
 partyDs.relations(true);

 QueryBuildDataSource locationDs = partyDs.addDataSource(tableNum(DirPartyLocation));
 locationDs.addRange(fieldNum(DirPartyLocation, IsPrimary)).value(queryValue(NoYes::Yes));
 locationDs.addLink(fieldNum(DirPartyTable, RecId), fieldNum(DirPartyLocation, Party));

 QueryBuildDataSource addressDs = locationDs.addDataSource(tableStr(LogisticsPostalAddress));
 addressDs.addLink(fieldNum(DirPartyLocation, Location), fieldNum(LogisticsPostalAddress, Location));

 return q;
}
```

Configuration keys and data entities

2/18/2021 • 5 minutes to read • Edit Online

Before you use data entities to import or export data, we recommended that you first determine the impact of configuration keys on the data entities that you are planning to use.

To learn more about configuration keys, see the License codes and configuration keys report.

Configuration key assignments

Configuration keys can be assigned to one or all of the following artifacts.

- Data entities
- Tables used as data sources
- Table fields
- Data entity fields

The following table summarizes how configuration key values on the different artifacts that underlie an object change the expected behavior of the object.

CONFIGURATION KEY SETTING ON DATA ENTITY	CONFIGURATION KEY SETTING ON TABLE	CONFIGURATION KEY SETTING ON TABLE FIELD	CONFIGURATION KEY ON DATA ENTITY FIELD	EXPECTED BEHAVIOR
Disabled	Not evaluated	Not evaluated	Not evaluated	If the configuration key for the data entity is disabled, the data entity will not be functional. It does not matter whether the configuration keys in the underlying tables and fields are enabled or disabled.

CONFIGURATION KEY SETTING ON DATA ENTITY	CONFIGURATION KEY SETTING ON TABLE	CONFIGURATION KEY SETTING ON TABLE FIELD	CONFIGURATION KEY ON DATA ENTITY FIELD	EXPECTED BEHAVIOR
Enabled	Disabled	Not evaluated	Not evaluated	If the configuration key for a data entity is enabled, the data management framework checks the configuration key on each of the underlying tables. If the configuration key for a table is disabled, that table will not be available in the data entity for functional use. If a table's configuration key is disabled, the table and data entity configuration key settings are not evaluated. If the primary table in the entity has its configuration key disabled, then the system will act as though the entity's configuration key were disabled.
Enabled	Enabled	Disabled	Not evaluated	If the configuration key for a data entity is enabled, and the underlying tables configuration keys are enabled, the data management framework will check the configuration key on of the fields in the tables. If the configuration key for a field is disabled, that field will not be available in the data entity for functional use even if the corresponding data entity field has the configuration key enabled.

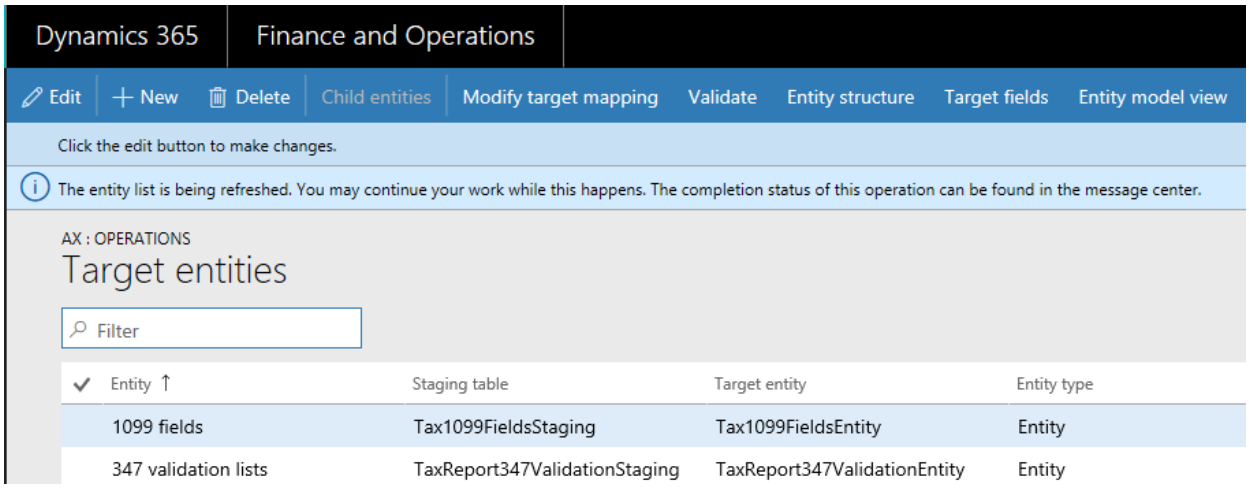
CONFIGURATION KEY SETTING ON DATA ENTITY	CONFIGURATION KEY SETTING ON TABLE	CONFIGURATION KEY SETTING ON TABLE FIELD	CONFIGURATION KEY ON DATA ENTITY FIELD	EXPECTED BEHAVIOR
Enabled	Enabled	Enabled	Disabled	If the configuration key is enabled at all other levels, but the entity field configuration key is not enabled, then the field will not be available for use in the data entity.

NOTE

If an entity has another entity as a data source then, the above semantics are applied in a recursive manner.

Entity list refresh

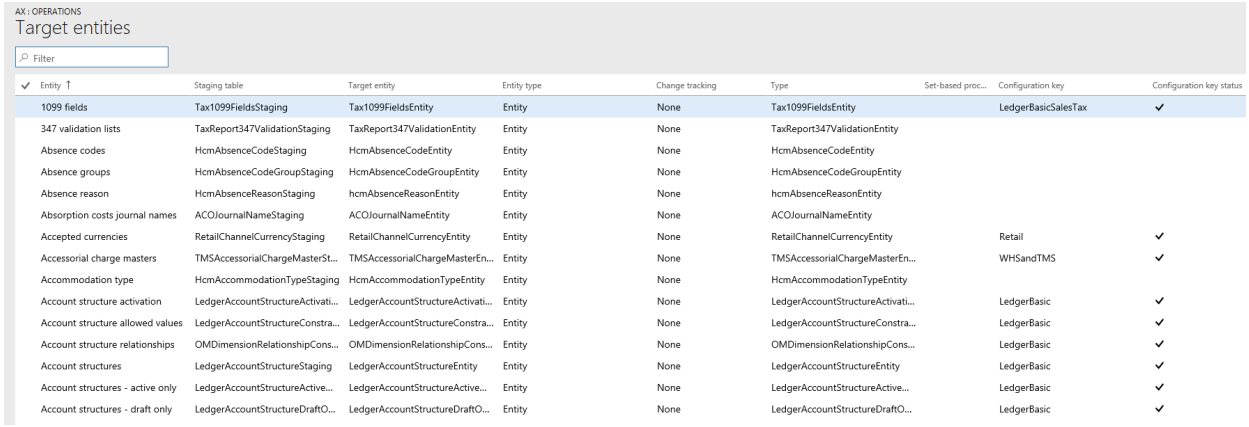
When the entity list is refreshed, the data management framework builds the configuration key metadata for runtime use. This metadata is built using the logic described above. We strongly recommend that you wait for the entity list refresh to complete before using jobs and entities in the data management framework. If you don't wait, the configuration key metadata may not be up to date and could result in unexpected outcomes. When the entity list is being refreshed, the following message is shown in the entity list page.



Data entity list page

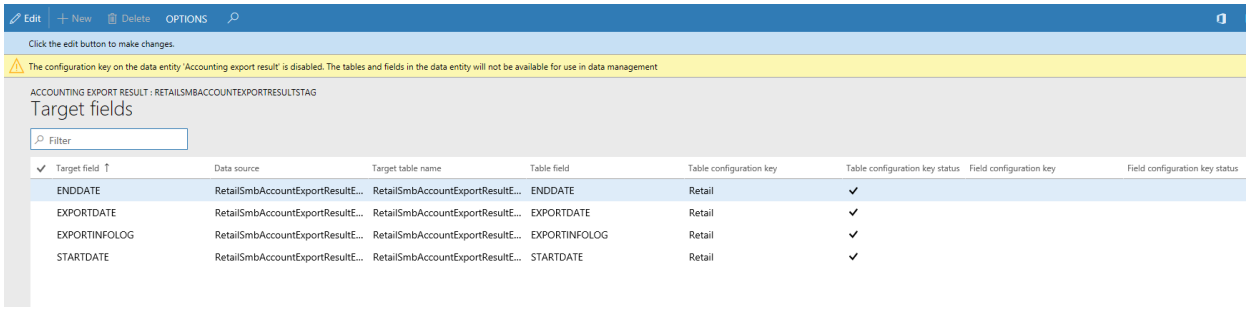
The data entity list page in the Data management workspace shows the configuration key settings for the entities. Start from this page to understand the impact from configuration keys on the data entity.

This information is shown using the metadata that is built during entity refresh. The configuration key column shows the name of the configuration key that is associated with the data entity. If this column is blank it means that there is no configuration key associated with the data entity. The configuration key status column shows the state of the configuration key. If it has a checkmark, it means the key is enabled. If it is blank, it means either the key is disabled or there is no key associated.



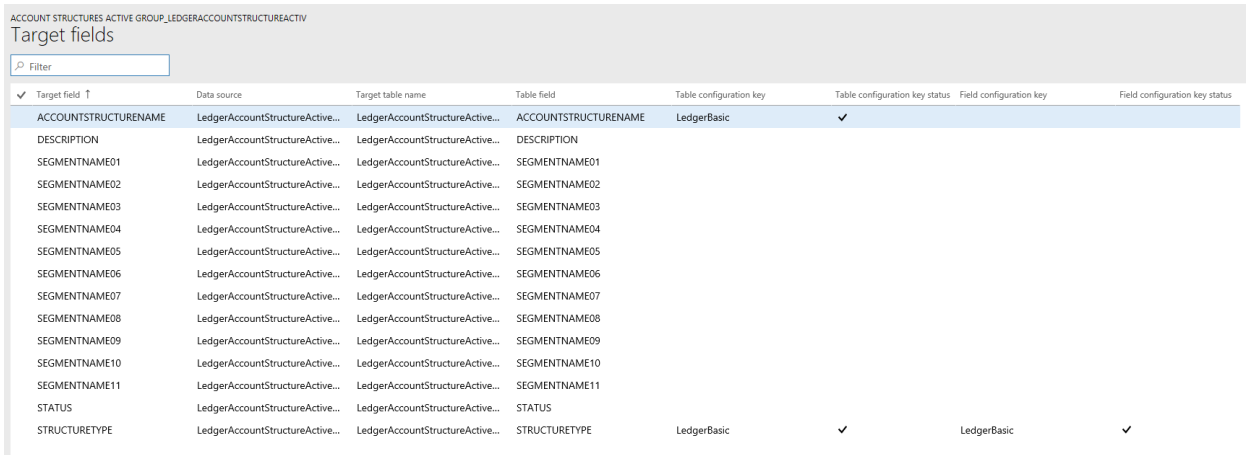
Target fields

The next step is to drill into the data entity to view the impact of configuration keys on tables and fields. The target fields form for a data entity shows configuration key and the key status information for the related tables and fields in the data entity. If the data entity itself has its configuration key disabled, a warning message is shown informing that the tables and fields in the target fields form for this entity will not be available at all regardless of their configuration key status.



Child entities

Certain entities have other entities as data sources, or are composite data entities: configuration key information for these entities is shown in the Child entities form. Use this form in the similar way to the entities list page described above. The target fields form for the child entity also behaves like what is described above.



Using data entities

After understanding the full impact, if any, of configuration keys on the data entities that you would like to use, you can now proceed to using the data entities by adding them to data projects.

Run time validations for configuration keys

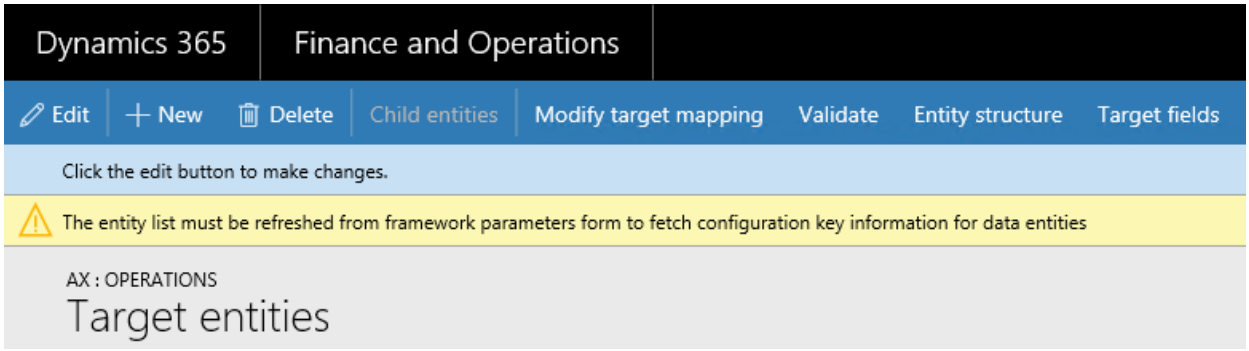
Using the configuration key metadata built during entity refresh list, run time validations are performed in the following use cases.

- When a data entity is added to a job

- When user clicks 'validate' on the entity list
- When the user loads a data package into a data project
- When the user loads a template into a data project
- When an existing data project is loaded
- When a template is loaded into a data project
- Before the export/import job is executed (batch, non-batch, recurring, OData)
- When the user generates mapping
- When the user maps fields in the mapping UI
- When the user adds only 'importable fields'

Managing configuration key changes

Anytime that you update configuration keys at the entity, table or field level, the entity list in the data management framework must be refreshed. This process ensures that the framework picks up the latest configuration key settings. Until the entity list is refreshed, the following warning will be shown in the entity list page. The updated configuration key changes will take effect immediately after the entity list is refreshed. We recommend that you validate existing data projects and jobs to make sure that they function as expected after the configuration keys changes are put in effect.



Configuration data packages

2/18/2021 • 7 minutes to read • Edit Online

IMPORTANT

This topic applies only to the July 2017 release of Microsoft Dynamics 365 for Finance and Operations. If you are running a later release, refer to the topic Copy configuration data between companies or legal entities overview.

Configuration data packages are available as process data packages from Microsoft Dynamics Lifecycle Services (LCS). These data packages can help improve the repeatability of implementations and accelerate the configuration.

Data packages contain configuration entity spreadsheets. These entity spreadsheets contain best practice data that you can use to create an initial golden build. The data entities in the data packages are also sequenced appropriately to help guarantee a successful single-click import of the data.

The entity spreadsheets include three types of data:

- **Business data** – The spreadsheet contains standard business data for a mid-sized trade or retail company. This data combines best practices and business standards that can be used as a starting point for your configuration.
- **Sample data** – The spreadsheet contains data that can be used as an example for business-specific data. This data can be imported and used as an example, but it must be changed for individual business practices.
- **No data** – The spreadsheet doesn't contain any data. Several areas of the product are unique to each business and its business practices. These areas must be configured specifically for the organization. These spreadsheets should be reviewed and updated for the organization as appropriate.

For more information about the type of data that is included in each entity spreadsheet in the data packages see the Data packages section of this topic. You can modify individual spreadsheets before you import the data packages, or you can import the data packages as they have been supplied and then update your data in the system.

Using configuration data packages

You can access configuration data packages from LCS. You can either apply them to an LCS environment, or download them so that you can manually import them.

1. Open your LCS project, and open the Asset library.
2. In the list of asset types, select **Process data package**.
3. Click **Import**.
4. Select the configuration data package.
5. Click **Pick**.

At this point, you can use the **Consume** function to apply the process data package to an LCS environment.

You can also download the individual data package files from the **Data package** area. Use the **Data management** workspace to import the data packages from LCS. For more information about how to import and export configurations, see Copy configuration data between companies or legal entities overview.

Special considerations

System setup

The System data package must be imported before any other data package. By default, the System data package creates a new legal entity that is named **ST01**. The data packages for the module areas depend on this legal entity.

General ledger

A generic chart of accounts is included in the configuration data packages. When this data is used as it's defined in the Main account entity spreadsheet, posting profiles across the system are filled with default posting data. If you change the main accounts that are used for the chart of accounts, you must also update the individual posting profiles and posting accounts for each area.

Data packages: System

010 – System Setup

	SPREADSHEET CONTENT		
Entity spreadsheet	**Business data**	**Sample data**	**No data**
Address and contact information purpose	X		
Address books			X
Address format lines	X		
Address format	X		
Address parameters	X		
Calendar			X
Cities	X		
Counties	X		
CountryRegions	X		
Currencies	X		
Districts	X		
Exchange rates		X	
Global address book parameters	X		
Global address book			X
Legal entities		X	
Name affixes	X		

	SPREADSHEET CONTENT		
Name sequences	X		
Operating unit		X	
Organization hierarchy purposes			X
Organization hierarchy type			X
Organization hierarchy			X
Party contacts			X
Party postal addresses			X
Party relationships			X
Postal codes	X		
Relationship types	X		
States	X		
System parameters	X		
Team types			X
Teams			X
Unit conversions	X		
Unit translations	X		
Units	X		
User groups			X
User information			X

Data packages: Financials

020 – GL Shared

	SPREADSHEET CONTENT		
Entity spreadsheet	**Business data**	**Sample data**	**No data**
Account structure activation		X	

	SPREADSHEET CONTENT		
Account structure allowed values		X	
Account structures		X	
Advanced rule criteria		X	
Advanced rule structure activation		X	
Advanced rule structure allowed values		X	
Advanced rule structures		X	
Advanced rules		X	
Chart of accounts	X		
Consolidation groups and accounts		X	
Dimension attribute activation		X	
Financial dimension format		X	
Financial dimension sets		X	
Financial dimension translations			X
Financial dimension value translations			X
Financial dimension values			X
Financial dimensions		X	
Fiscal calendar period		X	
Fiscal calendar		X	
Main account categories	X		
Main account	X		
Number sequence code		X	
Number sequence group			X

	SPREADSHEET CONTENT		
Number sequence references		X	

025 – GL

	SPREADSHEET CONTENT		
Entity spreadsheet	**Business data**	**Sample data**	**No data**
Accounts for automatic transactions	X		
Balance control accounts			X
Calendar			X
Date intervals	X		
Default descriptions		X	
Dimension parameters	X		
Document types	X		
Financial dimension value legal entity overrides			X
Financial reasons	X		
Journal descriptions	X		
Journal names	X		
Ledger allocation basis source			X
Ledger allocation basis			X
Ledger allocation rule destination			X
Ledger allocation rule source			X
Ledger allocation rule			X
Ledger fiscal calendar period		X	
Ledger fiscal calendar year		X	
Ledger parameters	X		

	SPREADSHEET CONTENT		
Ledger		X	
Main account legal entity overrides			X
Organization email template message			X
Organization email template			X
Subledger journal transfer rule	X		
Workflow organization parameters			X
Working times			X

100 – Bank

	SPREADSHEET CONTENT		
Entity spreadsheet	**Business data**	**Sample data**	**No data**
Bank accounts		X	
Bank facility groups			X
Bank facility types			X
Bank groups		X	
Bank parameters	X		
Bank posting profiles			X
Bank statement import methods for general journal			X
Bank transaction groups	X		
Bank transaction type	X		
Check layout		X	
Customer charge groups			X
Payment purpose codes			X
Reconciliation matching rule sets			X

	SPREADSHEET CONTENT		
Reconciliation matching rules			X
Transaction code mapping			X

120 – AP

	SPREADSHEET CONTENT		
Entity spreadsheet	**Business data**	**Sample data**	**No data**
Aging period definitions			X
Business segments			X
Business subsegments			X
Buyer groups			X
Cash discount			X
Delivery charges groups			X
Destination code			X
Expedite codes			X
Item charge groups			X
Line discount vendor groups			X
Line of business		X	
Modes of delivery			X
Multiline discount vendor groups			X
Payment calendar rule			X
Payment calendar			X
Payment day lines		X	
Payment schedule lines		X	
Payment schedule		X	
Price vendor groups			X

	SPREADSHEET CONTENT		
Procurement charge codes			X
Product categories			X
Product category hierarchies			X
Product category hierarchy roles			X
Product category hierarchy translations			X
Purchase pools			X
Sales tax groups			X
Terms of delivery			X
Terms of payment	X		
Total discount vendor groups			X
Vendor bank accounts			X
Vendor charges group			X
Vendor exception groups			X
Vendor groups			X
Vendor invoice form printing configurations			X
Vendor invoice policy rule types			X
Vendor ledger accounts	X		
Vendor parameters	X		
Vendor payment fee			X
Vendor payment format			X
Vendor payment method specification			X
Vendor payment method		X	

	SPREADSHEET CONTENT		
Vendor posting profile	X		
Vendor price tolerance groups			X
Vendor reasons	X		
Vendor, form parameters			X
Vendors			X

130 – Tax

	SPREADSHEET CONTENT		
Entity spreadsheet	**Business data**	**Sample data**	**No data**
Sales tax authorities		X	
Sales tax code limits			X
Sales tax code values		X	
Sales tax codes		X	
Sales tax exempt code	X		
Sales tax exempt numbers			X
Sales tax group details		X	
Sales tax groups		X	
Sales tax item groups		X	
Sales tax ledger posting groups	X		
Sales tax parameters preset entity			X
Sales tax parameters	X		
Sales tax period setup		X	
Sales tax reporting codes			X
Transaction codes			X
Withholding tax code values		X	

	SPREADSHEET CONTENT		
Withholding tax codes		X	
Withholding tax group details			X
Withholding tax groups		X	
Withholding tax limits			X

140 – AR

	SPREADSHEET CONTENT		
Entity spreadsheet	**Business data**	**Sample data**	**No data**
Aging period definitions	X		
Business segments			X
Business subsegments			X
Cash discount			X
Collection letter form printing configurations			X
Collection letter setup	X		
Commission calculation			X
Commission customer group			X
Commission item group			X
Commission sales group			X
Customer account statement form printing configurations			X
Customer bank accounts			X
Customer charge groups			X
Customer definitions			X
Customer details			X
Customer facing form printing configurations			X

	SPREADSHEET CONTENT		
Customer groups		X	
Customer ledger accounts	X		
Customer parameters	X		
Customer payment fee			X
Customer payment method specification			X
Customer payment method		X	
Customer posting profiles	X		
Customer reasons			X
Customer write-off reason codes	X		
Customers			X
Delivery charges groups			X
Expedite codes			X
Free text invoice form printing configurations			X
Interest codes with fees			X
Interest codes with ranges			X
Interest codes	X		
Interest note form printing configurations			X
Item charge groups			X
Line discount customer groups			X
Line of business			X
Modes of delivery			X
Payment calendar			X
Payment day lines			X

	SPREADSHEET CONTENT		
Payment schedule lines			X
Payment schedule			X
Price customer groups			X
Printed form notes			X
Sales charge codes			X
Sales districts			X
Sales order pools			X
Statistics group			X
Terms of delivery			X
Terms of payment			X
Total discount customer groups			X

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Data import and export jobs overview

2/18/2021 • 16 minutes to read • Edit Online

To create and manage data import and export jobs, you use the **Data management** workspace. By default, the data import and export process creates a staging table for each entity in the target database. Staging tables let you verify, clean up, or convert data before you move it.

> **NOTE**
> This topic assumes that you are familiar with data entities.

Data import/export process

Here are the steps to import or export data.

1. Create an import or export job where you complete the following tasks:
   - Define the project category.
   - Identify the entities to import or export.
   - Set the data format for the job.
   - Sequence the entities, so that they are processed in logical groups and in an order that makes sense.
   - Determine whether to use staging tables.
2. Validate that the source data and target data are mapped correctly.
3. Verify the security for your import or export job.
4. Run the import or export job.
5. Validate that the job ran as expected by reviewing the job history.
6. Clean up the staging tables.

The remaining sections of this topic provide more details about each step of the process.

> **NOTE**
> In order to refresh the Data import/export form to see the latest progress, use the form refresh icon. Browser level refresh is not recommended because it will interrupt any import/export jobs that are not run in batch.

Create an import or export job

A data import or export job can be run one time or many times.

Define the project category

We recommend that you take the time to select an appropriate project category for your import or export job. Project categories can help you manage related jobs.

Identify the entities to import or export

You can add specific entities to an import or export job or select a template to apply. Templates fill a job with a list of entities. The **Apply template** option is available after you give the job a name and save the job.

Set the data format for the job

When you select an entity, you must select the format of the data that will be exported or imported. You define formats by using the **Data sources setup** tile. A source data format is a combination of **Type**, **File format**, **Row delimiter** and **Column delimiter**. There are also other attributes, but these are the key ones to understand. The following table lists the valid combinations.

FILE FORMAT	ROW/COLUMN DELIMITER	XML STYLE
Excel	Excel	-NA-
XML	-NA-	XML-Element XML-Attribute
Delimited, fixed width	Comma, semicolon, tab, vertical bar, colon	-NA-

Sequence the entities

Entities can be sequenced in a data template, or in import and export jobs. When you run a job that contains more than one data entity, you must make sure that the data entities are correctly sequenced. You sequence entities primarily so that you can address any functional dependencies among entities. If entities don't have any functional dependencies, they can be scheduled for parallel import or export.

Execution units, levels, and sequences

The execution unit, level in the execution unit, and sequence of an entity help control the order that the data is exported or imported in.

- Entities in different execution units are processed in parallel.
- In each execution unit, entities are processed in parallel if they have the same level.
- In each level, entities are processed according to their sequence number in that level.
- After one level has been processed, the next level is processed.

Resequencing

You might want to resequence your entities in the following situations:

- If only one data job is used for all your changes, you can use resequencing options to optimize the execution time for the full job. In these cases, you can use the execution unit to represent the module, the level to represent the feature area in the module, and the sequence to represent the entity. By using this approach, you can work across modules in parallel, but you can still work in sequence in a module. To help guarantee that parallel operations succeed, you must consider all dependencies.
- If multiple data jobs are used (for example, one job for each module), you can use sequencing to affect the level and sequence of entities for optimal execution.
- If there are no dependencies at all, you can sequence entities at different execution units for maximum optimization.

The **Resequencing** menu is available when multiple entities are selected. You can resequence based on execution unit, level, or sequence options. You can set an increment to resequence the entities that have been selected. The unit, level, and/or sequence number that is selected for each entity is updated by the specified increment.

Sorting

Use can use the **Sort by** option to view the entity list in sequential order.

Truncating

For import projects, you can choose to truncate records in the entities prior to import. This is useful if your records must be imported into a clean set of tables. This setting is off by default.

Validate that the source data and target data are mapped correctly

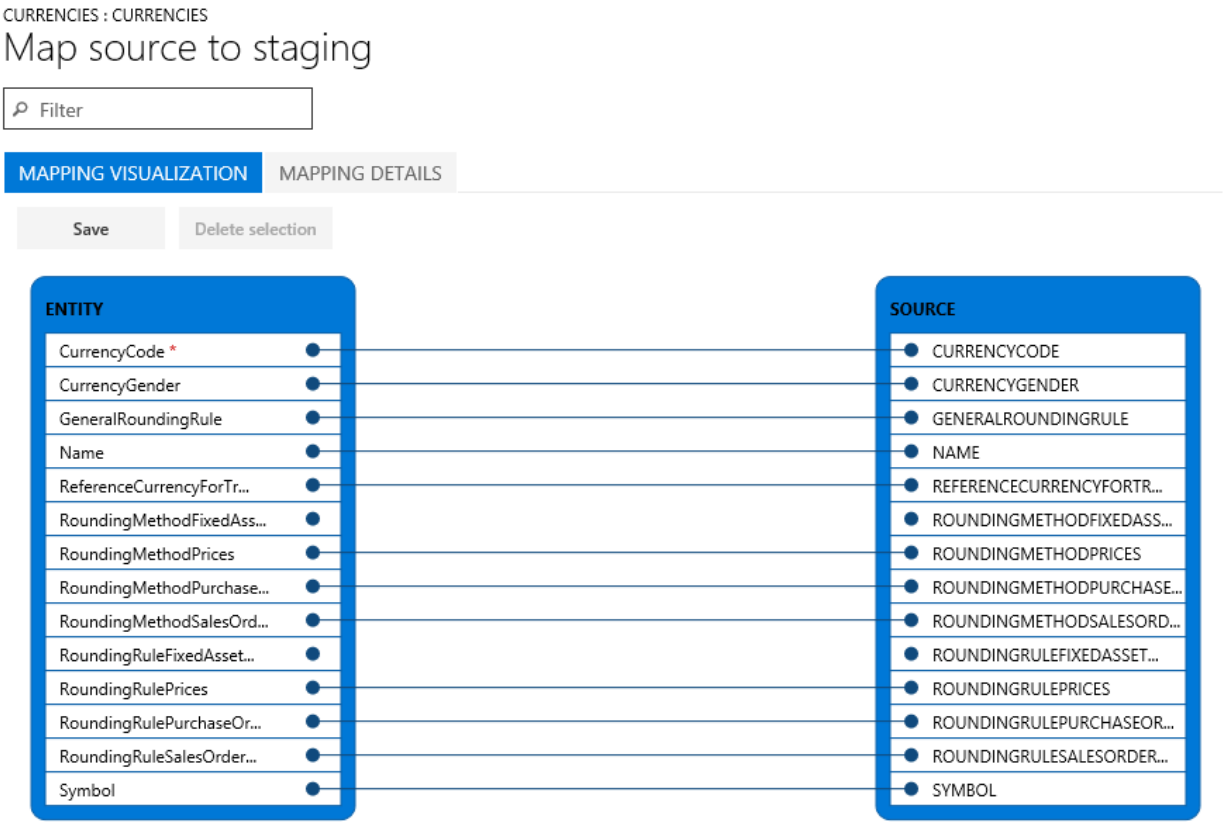
Mapping is a function that applies to both import and export jobs.

- In the context of an import job, mapping describes which columns in the source file become the columns in the staging table. Therefore, the system can determine which column data in the source file must be copied into which column of the staging table.
- In the context of an export job, mapping describes which columns of the staging table (that is, the source) become the columns in the target file.

If the column names in the staging table and the file match, the system automatically establishes the mapping, based on the names. However, if the names differ, columns aren't mapped automatically. In these cases, you must complete the mapping by selecting the **View map** option on the entity in the data job.

There are two mapping views: **Mapping visualization**, which is the default view, and **Mapping details**. A red asterisk (*) identifies any required fields in the entity. These fields must be mapped before you can work with the entity. You can unmap other fields as you require when you work with the entity. To unmap a field, select the field in either the **Entity** column or the **Source** column, and then select **Delete selection**. Select **Save** to save your changes, and then close the page to return to the project. You can use the same process to edit the field mapping from source to staging after you import.

You can generate a mapping on the page by selecting **Generate source mapping**. A generated mapping behaves like an automatic mapping. Therefore, you must manually map any unmapped fields.



Verify the security for your import or export job

Access to the **Data management** workspace can be restricted, so that non-administrator users can access only specific data jobs. Access to a data job implies full access to the execution history of that job and access to the staging tables. Therefore, you must make sure that appropriate access controls are in place when you create a data job.

Secure a job by roles and users

Use the **Applicable roles** menu to restrict the job to one or more security roles. Only users in those roles will have access to the job.

You can also restrict a job to specific users. When you secure a job by users instead of roles, there is more control if multiple users are assigned to a role.

Secure a job by legal entity

Data jobs are global in nature. Therefore, if a data job was created and used in a legal entity, the job will be visible in other legal entities in the system. This default behavior might be preferred in some application scenarios. For example, an organization that imports invoices by using data entities might provide a centralized invoice processing team that is responsible for managing invoice errors for all divisions in the organization. In this scenario, it's useful for the centralized invoice processing team to have access to invoice import jobs from all legal entities. Therefore, the default behavior meets the requirement from a legal entity perspective.

However, an organization might want to have invoice processing teams per legal entity. In this case, a team in a legal entity should have access only to the invoice import job in its own legal entity. To meet this requirement, you can configure legal entity–based access control on the data jobs by using the **Applicable legal entities** menu inside the data job. After the configuration is done, users can see only jobs that are available in the legal entity that they are currently signed in to. To see jobs from another legal entity, users must switch to that legal entity.

A job can be secured by roles, users, and legal entity at the same time.

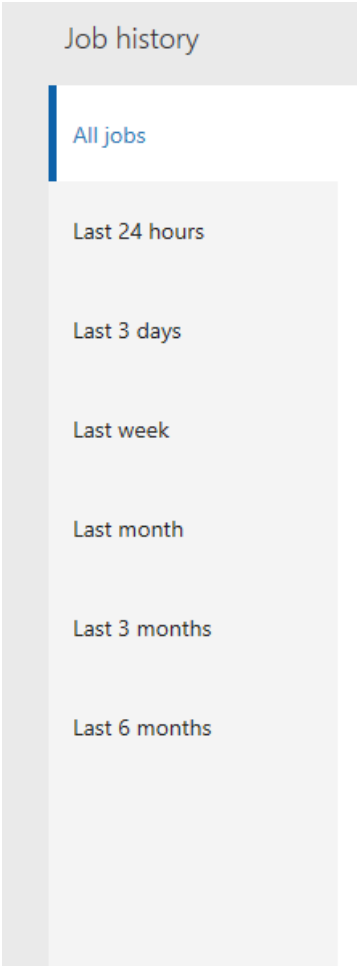
Run the import or export job

You can run a job one time by selecting the **Import** or **Export** button after you define the job. To set up a recurring job, select **Create recurring data job**.

> **NOTE**
>
> An import or an export job can be run by selecting the **Import** or **Export** button. This will schedule a batch job to run only once. The job may not execute immediately if batch service is throttling due to the load on the batch service. The jobs can also be run synchronously by selecting **Import now** or **Export now**. This starts the job immediately and is useful if the batch does not start due to throttling. The jobs can also be scheduled to execute at a later time. This can be done by choosing the **Run in batch** option. Batch resources are subject to throttling, so the batch job might not start immediately. Using a batch is the recommended option because it will also help with large volumes of data that need to be imported or exported. Batch jobs can be scheduled to run on a specific batch group, which allows more control from a load balancing perspective.

Validate that the job ran as expected

The job history is available for troubleshooting and investigation on both import and export jobs. Historical job runs are organized by time ranges.



Each job run provides the following details:

- Execution details
- Execution log

Execution details show the state of each data entity that the job processed. Therefore, you can quickly find the following information:

- Which entities were processed.
- For an entity, how many records were successfully processed, and how many failed.
- The staging records for each entity.

You can download the staging data in a file for export jobs, or you can download it as a package for import and export jobs.

From the execution details, you can also open the execution log.

Parallel imports

To speed up the import of data, parallel processing of importing a file can be enabled if the entity supports parallel imports. To configure the parallel import for an entity, the following steps must be followed.

1. Go to **System administration** > **Workspaces** > **Data management**.
2. In the **Import / Export** section, select the **Framework parameters** tile to open the **Data import/export framework parameters** page.
3. On the **Entity settings** tab, select **Configure entity execution parameters** to open the **Entity import execution parameters** page.
4. Set the following fields to configure parallel import for an entity:

- In the **Entity** field, select the entity.
- In the **Import threshold record count** field, enter the threshold record count for import. This determines the record count to be processed by a thread. If a file has 10K records, a record count of 2500 with a task count of 4 will mean, each thread will process 2500 records.
- In the **Import task count** field, enter the count of import tasks. This must not exceed the max batch threads allocated for batch processing in **System administration > Server configuration**.

Clean up the staging tables

Starting in Platform update 29, this functionality has been deprecated. This is replaced by a new version of job history clean-up functionality explained below.

You can clean up staging tables by using the **Staging clean up** feature in the **Data management** workspace. You can use the following options to select which records should be deleted from which staging table:

- **Entity** – If only an entity is provided, all records from that entity's staging table are deleted. Select this option to clean up all the data for the entity across all data projects and all jobs.
- **Job ID** – If only a job ID is provided, all records for all entities in the selected job are deleted from the appropriate staging tables.
- **Data projects** – If only a data project is selected, all records for all entities and across all jobs for the selected data project are deleted.

You can also combine the options to further restrict the record set that is deleted.

Job history clean up (available in Platform update 29 and later)

The job history clean-up functionality in data management must be used to schedule a periodic cleanup of the execution history. This functionality replaces the previous staging table clean-up functionality, which is now deprecated. The following tables will be cleaned up by the clean-up process.

- All staging tables
- DMFSTAGINGVALIDATIONLOG
- DMFSTAGINGEXECUTIONERRORS
- DMFSTAGINGLOGDETAIL
- DMFSTAGINGLOG
- DMFDEFINITIONGROUPEXECUTIONHISTORY
- DMFEXECUTION
- DMFDEFINITIONGROUPEXECUTION

The **Execution history cleanup** feature must be enabled in feature management and then can be accessed from **Data management > Job history cleanup**.

Scheduling parameters

When scheduling the clean-up process, the following parameters must be specified to define the clean-up criteria.

- **Number of days to retain history** – This setting is used to control the amount of execution history to be preserved. This is specified in number of days. When the clean-up job is scheduled as a recurring batch job, this setting will act like a continuously moving window thereby, always leaving the history for the specified number of days intact while deleting the rest. The default is 7 days.

- **Number of hours to execute the job** – Depending on the amount of history to be cleaned up, the total execution time for the clean-up job can vary from a few minutes to a few hours. This parameter must be set to the number of hours that the job will execute. After the clean-up job has executed for the specified number of hours, the job will exit and will resume the clean up the next time it is run based on the recurrence schedule.

  A maximum execution time can be specified by setting a max limit on the number of hours the job must run using this setting. The clean-up logic goes through one job execution ID at a time in a chronologically arranged sequence, with oldest being first for the cleanup of related execution history. It will stop picking up new execution ID's for cleanup when the remaining execution duration is within the last 10% of the specified duration. In some cases, it will be expected that the clean-up job will continue beyond the specified max time. This will largely depend on the number of records to be deleted for the current execution ID that was started before the 10% threshold was reached. The cleanup that was started must be completed to ensure data integrity, which means that cleanup will continue despite exceeding the specified limit. When this is complete, new execution ID's are not picked up and the clean-up job completes. The remaining execution history that was not cleaned up due to lack of enough execution time, will be picked up the next time the clean-up job is scheduled. The default and minimum value for this setting is set to 2 hours.

- **Recurring batch** – The clean-up job can be run as a one-time, manual execution, or it can be also scheduled for recurring execution in batch. The batch can be scheduled using the **Run in background** settings, which is the standard batch set up.

> **NOTE**
> If records in the staging tables are not cleaned up completely, ensure that the cleanup job is scheduled to run in recurrence. As explained above, in any clean up execution the job will only clean up as many execution ID's as is possible within the provided maximum hours. In order to continue cleanup of any remaining staging records, the job must be scheduled to run periodically.

Job history clean up and archival (available for preview in Platform update 39 or version 10.0.15)

The job history clean up and archival functionality replaces the previous versions of the clean up functionality. This section will explain these new capabilities.

One of the main changes to the clean up functionality is the use of system batch job for cleaning up the history. The use of system batch job allows Finance and Operations apps to have the clean up batch job automatically scheduled and running as soon as the system is ready. It is no longer required to schedule the batch job manually. In this default execution mode, the batch job will execute every hour starting at 12 midnight and will retain the execution history for the most recent 7 days. The purged history is archived for future retrieval.

> **NOTE**
> Because this functionality is in preview, the system batch job will not delete any execution history until it is enabled via the flight DMFEnableExecutionHistoryCleanupSystemJob. When the feature is generally available in a future release, this flight will not be required and the system batch job will start to purge and archive after the system is ready, based on the defined schedule as explained above.

> **NOTE**
> In a future release, the previous versions of the clean up functionality will be removed from Finance and Operations apps.

The second change in the clean up process is the archival of the purged execution history. The clean up job will archive the deleted records to the blob storage that DIXF uses for regular integrations. The archived file will be in the DIXF package format and will be available for 7 days in the blob during which time it can be downloaded. The default longevity of 7 days for the archived file can be changed to a maximum of 90 days in the parameters.

Changing the default settings

This functionality is currently in preview and must be explicitly turned on by enabling the flight DMFEnableExecutionHistoryCleanupSystemJob. The staging clean up feature must also be turned on in feature management.

To change the default setting for the longevity of the archived file, go to the data management workspace and select **Job history cleanup**. Set **Days to retain package in blob** to a value between 7 and 90 (inclusive). This will take effect on the archives that are created after this change was made.

Downloading the archived package

This functionality is currently in preview and must be explicitly turned on by enabling the flight DMFEnableExecutionHistoryCleanupSystemJob. The staging clean up feature must also be turned on in feature management.

To download the archived execution history, go to the data management workspace and select **Job history cleanup**. Select **Package backup history** to open the history form. This form shows the list of all archived packages. An archive can be selected and downloaded by selecting **Download package**. The downloaded package will be in the DIXF package format and contain the following files:

- The entity staging table file
- DMFDEFINITIONGROUPEXECUTION
- DMFDEFINITIONGROUPEXECUTIONHISTORY
- DMFEXECUTION
- DMFSTAGINGEXECUTIONERRORS
- DMFSTAGINGLOG
- DMFSTAGINGLOGDETAILS
- DMFSTAGINGVALIDATIONLOG

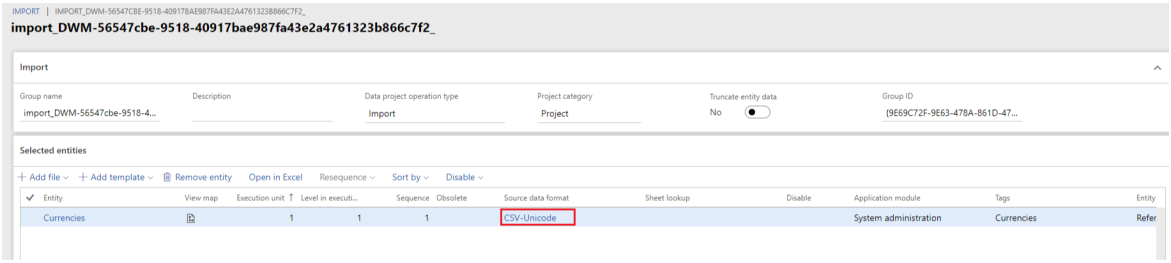
Synchronize date and time in import jobs

2/18/2021 • 2 minutes to read • Edit Online

It's important to set the time zone for your import job to Coordinated Universal Time (UTC). You might see unexpected dates and times in your imported data if you use a different setting. Without the correct setting, the import process converts the UTC date to the local format, and then system settings converts it again.

This dual conversion causes dates to change between applications. For example, the dual conversion could cause an employee's start date to be different between Dynamics 365 Human Resources and Dynamics 365 Finance due to differences in local time zones. Setting the import job to UTC resolves this problem.

1. In Dynamics 365 Finance and Operations, select **Data management**.
2. Select **Import projects**, and then select the project.
3. Under **Source date format**, select **CSV-Unicode**.



4. Change **Timezone** to **Coordinated Universal Timezone**, and change **Language** to **En-US**.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Importing vouchers by using the General journal entity

2/18/2021 • 5 minutes to read • Edit Online

This topic provides tips for importing data into the General journal by using the General journal entity.

You can use the General journal entity to import vouchers that have an account or offset account type of **Ledger**, **Customer**, **Vendor**, or **Bank**. The voucher can be entered as one line, using both the **Account** field and the **Offset account** field, or as a multi-line voucher, where only the **Account** field is used (and the **Offset account** is left blank on each line). The General journal entity doesn't support every account type. Instead, other entities exist for scenarios where different combinations of account types are required. For example, to import a project transaction, use the Project expense journal entity. Each entity is designed to support specific scenarios. This means additional fields may be available in entities for those scenarios. However, additional fields might not be available in entities for different scenarios.

Setup

Before you import by using the General journal entity, validate the following setup:

- **Number sequence setup for the journal batch number** – By default, when you import by using the General journal entity, the journal batch number uses the number sequence that is defined in the General ledger parameters. If you set the number sequence for the journal batch number to **Manual**, a default number isn't applied. This setup isn't supported.
- **Financial dimension configuration** – Every organization must define the order of financial dimensions when entities are used to import transactions. The order is defined for the **Ledger dimensions integration** format, at **General ledger** > **Chart of accounts** > **Dimensions** > **Financial dimension configuration for integrating applications** > **Select data entities**. The segments of the ledger account that is imported must have the same order. Otherwise, an error will occur during import.

General journal entity setup

Two settings in Data management affect how the default journal batch number or voucher number is applied:

- **Set-based processing** (on the data entity)
- **Auto-generated** (on the field mapping)

The following sections describe the effect of these settings. They also explain how the system generates batch numbers for journals and voucher numbers.

Journal batch number

- The **Set-based processing** setting on the General journal entity doesn't affect the way that journal batch numbers are generated.
- If the **Journal batch number** field is set to **Auto-generated**, a new journal batch number is created for every line that is imported. This behavior isn't recommended. The **Auto-generated** setting is found on the import project, under **View map**, on the **Mapping details** tab.
- If the **Journal batch number** field isn't set to **Auto-generated**, the journal batch number is created as follows:
    - If the journal batch number that is defined in the imported file matches an existing, unposted daily

journal, all lines that have a matching journal batch number are imported into the existing journal. Lines are never imported into a posted journal batch number. Instead, a new number is created.

- If the journal batch number that is defined in the imported file doesn't match an existing, unposted daily journal, all lines that have the same journal batch number are grouped under a new journal. For example, all lines that have a journal batch number of 1 are imported into a new journal, and all lines that have a journal batch number of 2 are imported into a second new journal. The journal batch number is created by using the number sequence that is defined in the General ledger parameters.

Voucher number

- When you use the **Set-based processing** setting on the General journal entity, the voucher number must be provided in the imported file. Every transaction in the General journal is assigned the voucher number that is provided in the imported file, even if the voucher isn't balanced. Note the following points if you want to use set-based processing, but you also want to use the number sequence that is defined for voucher numbers.
  - To enable this functionality, on the journal name that is used for imports, set **Number allocation at posting** to **Yes**.
  - A voucher number must still be defined in the imported file. However, this number is temporary and will be overwritten by the voucher number when the journal is posted. Be sure that the lines of the journal are grouped correctly by temporary voucher number. For example, during posting, three lines are found that have a temporary voucher number of 1. The temporary voucher number of all three lines is overwritten by the next number in the number sequence. If those three lines aren't a balanced entry, the voucher won't be posted. Next, if lines are found that have a temporary voucher number of 2, this number is overwritten by the next voucher number in the sequence, and so on.
- When you don't use the **Set-based processing** setting, you do not need to provide a voucher number in the imported file. The voucher numbers are created during import, based on the setup of the journal name (**One voucher only**, **In connection of balance**, and so on). For example, if the journal name is defined as **In connection of balance**, the first line receives a new default voucher number. The system then evaluates the line to determine whether the debits equal the credits. If an offset account exists on the line, the next line that is imported receives a new voucher number. If no offset account exists, the system evaluates whether the debits equal the credits as each new line is imported.
- If the **Voucher number** field is set to **Auto-generated**, the import won't succeed. The **Auto-generated** setting for the **Voucher number** field isn't supported.

By default, the General journal entity uses set-based processing. After you evaluate the business requirements for your organization, you can change the **Set-based processing** setting by clicking **Data entities** in the **Data management** workspace. Set-based processing is used to speed up the import process. If you don't use set-based processing, import of the General journal entity import will be slower.

Optimize data migration for Finance and Operations apps

2/18/2021 • 8 minutes to read • Edit Online

Data migration is a key success factor in almost every implementation. A primary concern of some customers is the speed that data can be migrated at, especially if there are vast amounts of data and a small cutover window. The Data migration framework is also used to move data as part of business requirements and operations.

The information in this topic represents a set of steps and actions that you can use to optimize the performance of data migration.

> **NOTE**
> Testing results in a Tier-1 environment should not be compared or extrapolated to performance in a Tier-2 or higher sandbox environment.

Not all standard entities have been optimized for data migration. Some entities have been optimized for integration with Open Data Protocol (OData), and if a required entity can't be optimized to meet the performance requirements, we recommend that you create a new optimized entity. A developer can accelerate this process by duplicating an existing entity.

Begin the optimization phase by using a subset of the data. For example, if you must import one million records, consider starting with 1,000 records, then increase the number to 10,000 records, and then increase it to 100,000 records.

After you've identified the entities that you will use, you should go through the following sections to explore opportunities for optimization.

Disable change tracking

You can enable and disable change tracking from the list of entities.

1. In the **Data management** workspace, select the **Data entities** tile.
2. On the **Target entities** page, select the entity in the grid, and then, on the Action Pane, on the **Change tracking** tab, select **Disable Change Tracking**.

Enable set-based processing

Follow these steps to verify that an entity supports set-based processing.

1. In the **Data management** workspace, select the **Data entities** tile.
2. On the **Target entities** page, find the entity in the grid, and review the value in the **Set-based processing** column.

For an example that shows how set-based processing can be enabled for the **General Journal** entity, see Best practices for importing vouchers by using the General journal entity. Not all entities support set-based processing. For example, if you try to enable support set-based processing for the **Customer definitions** (**CustCustomerBaseEntity**) entity and save your change, you will receive the following warning message:

```
Set operations not supported for 'Customer definitions' entity.
```

Here are some important considerations if you must create an entity that allows for set-based processing:

- The data sources can't be read-only.
- The **ValidTimeStateEnabled** property of the data entity view the must be set to **No**.
- All tables on the data sources must have **TableType** property set to **Regular**.
- The property **QueryType** on the used **Query** can't be set to **Union**.
- The main data source can't prevent data from being saved across companies. However, embedded data sources allow it.
- The main data source can't prevent data from being saved across partitions. However, embedded data sources allow it.

Create a data migration batch group

During cutover, run the data migration while there is little or no other activity. It can be helpful to configure and use a batch group where most or all compute nodes are assigned.

You can configure a batch group on the **Batch group** page (**System administration > Setup > Batch group**).

Enable priority-based batch scheduling

In Platform update 31, the new priority-based batch scheduling feature optimizes how batch jobs are run. If contention is identified in the batch framework, consider enabling priority-based batch scheduling.

Configure the maximum number of batch threads

To better use parallelism and multithreading, you can configure the maximum number of batch threads per instance of Application Object Server (AOS) by setting the **Maximum batch threads** field on the **Server configuration** page (**System administration > Setup > Server configuration**). Be careful about changing the value of this field. A value that is too high can have negative performance implications. Currently, the default value is **4**. You can change the value to **8** as you require. However, you should not set the field to a value that is more than 8 unless you do significant performance testing.

Import in batch mode

Whenever you run an import job, make sure that it's run in batch mode. Otherwise, a single thread will be used to run the job. In this case, the system won't be able to take full advantage of these optimization configurations.

Clean staging tables

We recommend that you clean up the staging tables. In Platform update 29 and later, you can achieve this optimization by scheduling the Job history cleanup job. To schedule this job, select the **Job history cleanup** tile in the **Data management** workspace.

> **NOTE**
> You must first turn on the **Execution history cleanup** feature in the **Feature management** workspace.

Update statistics

Before you run a data migration job that involves a large volume of data, consider updating the statistics across the associated tables. This optimization applies specifically to sandbox environments, because it's handled automatically in production environments. You can update statistics for a specific table from LCS. Alternatively, in

a sandbox environment, you can use the **sp_updatestats** stored procedure through direct SQL.

Clean the data

The time that is spent on validations and error reporting will increase the total time of the migration. Consider this fact when you import a high volume of invalid or inconsistent data. We recommend that you try to fix and reduce errors that are related to data quality. In this way, you help prevent unnecessary executions of validation and error handling.

Configurations to test during test runs of data migration

The following configurations can affect performance. Therefore, we recommend that you test changes by using different values that are suitable for your scenario.

Configure entity execution parameters

Follow these steps to change the execution parameters for all entities or specific entities.

1. In the **Data management** workspace, select the **Framework parameters** tile.
2. On the **Data import/export framework parameters** page, on the **Entity settings** tab, select **Configure entity execution parameters**.
3. On the **Entity import execution parameters** page, set the **Import threshold record count** and **Import task count** fields as appropriate for the desired entities.

Import threshold record count

This value determines the number of records to be processed per thread. By modifying the **Import threshold record count** you can control how you want to split the import into smaller tasks.

Import task count

This field defines the number of threads that will be used for the data migration job for a specific entity. For example, the **Maximum batch threads** field for each server is set to **8**, and four servers are assigned to the data migration batch group. In this case, the total maximum value of the **Import task count** field is **32** (= 8 × 4).

If a data entity doesn't support multithreading, you receive an error message when you try to configure the entity. Here is an example:

> Custom sequence is defined for the entity 'Customers V3', more than one task is not supported.

Validations

Validation logic for record insertions or updates might have been incorporated into the system, or there might be validation of individual fields. If the data migration is mature enough, the time that is spent on imports can be significantly reduced by disabling this validation, if it can be disabled.

Follow these steps to change the settings for each entity.

1. In the **Data management** workspace, select the **Data entities** tile.
2. On the **Target entities** page, select the entity in the grid, and then, on the Action Pane, select **Entity structure**.
3. On the **Entity structure** page, set the **Run business validations** and **Run business logic in insert or update method** fields as appropriate.

Run business validations

If this check box is selected, the system will run any logic that is written into the **validateWrite()** method on the table. It will also run any related event handlers.

Run business logic in insert or update method

If this check box is selected, the system will run any logic that is written into the **insert()** or **update()** method on the table. It will also run any related event handlers.

Call the validateField method on the target

Follow these steps to run field validation.

1. In the **Data management** workspace, select the **Data entities** tile.
2. On the **Target entities** page, select the entity in the grid, and then, on the Action Pane, select **Modify target mapping**.
3. On the **Map staging to target** page, select the **Call validate Field method** check box for the field that validation should be run for. The **validateField(FieldId p1)** method will then be called for that field.

Recommendations for optimizing data migration performance

Here are some general recommendations about the approach that you should use to optimize the performance of data migration:

- Break up large files into smaller chunks. This approach gives the SQL optimizer time to determine whether a new query plan will be optimal.
- Test performance in an appropriate Tier-2 or higher environment.
- Test performance in a mock cutover before go-live.

Testing of data migration performance is an iterative process. We recommend that you collect and compare information about each test to determine the optimal configuration for a specific entity. You should collect and verify some of the following settings:

- The batch group that is used
- The number of batch servers that are assigned to each batch group
- The maximum number of batch threads per batch server

Here is an example of the information that you might collect for each entity.

INFORMATION	DESCRIPTION
Entity	The name of the entity that is being tested
Number of records	The number of records that are being imported
Source format	The source format of the data that is being imported
Change tracking disabled	Yes/No
Set-based processing	Yes/No
Import threshold record count	The number of records
Import task count	The number of tasks
Run business validations	Yes/No
Run business logic in insert or update method	Yes/No
Call validate Field method	Yes/No (potential field list)

INFORMATION	DESCRIPTION
Required performance	The amount of time that the import must be completed within to achieve the cutover window
Actual performance	The actual amount of time that was required to import records

There are more areas where performance can be optimized. For example, you can analyze the specific queries and query plans. However, those processes are covered in other topics.

For more information about performance troubleshooting and optimization, see the following topics:

- Performance troubleshooting using tools in Lifecycle Services (LCS)
- Query cookbook

Design principles and best practices for data entities

2/18/2021 • 4 minutes to read • Edit Online

This article describes design principles for data entities. It also includes guidelines for the names of data entities, fields, relation roles, roles, and OData EntityTypes and EntitySets.

Entity design principles

A data entity should provide a holistic object that encapsulates the relevant business logic in a single consumable contract. The contract is then exposed through application interfaces (APIs), such as OData, import and export, integration, and the programming model. Each data entity should be designed to meet the following goals.

Encapsulation

- Each entity should provide an abstraction between the physical data model and the consumer of the entity. The entity should encapsulate the underlying tables that, together, can define an object that has a specific purpose in the business. Consumers of the entity might be form clients, services, and integration.
- Each entity should encapsulate multiple related tables to represent the domain object. In some situations, single table entities are acceptable.

A single public contract

- The public contract for an entity should be the same across all integration end points. For example, the customer entity must expose the same fields or API to both OData and import/export. This principle guarantees that the published schema for an entity is consistent, regardless of the mechanism for consumer interaction.
- When an entity is consumed, the business logic that is executed within the entity during CRUD operations must not vary based on the type of consumer.

Simplicity

- The consumer of an entity should be able to interact with the entity based on the accepted industry or domain definitions of the entity. The behavior details of the entity should be kept hidden and should be prevented from distorting the interaction.
- The consumer of an entity must be able to interact with the entity by using the natural key of the entity. The consumer must not be required to use any keys that are implementation-specific, such as any surrogate key that it generates.

Naming guidelines

Data entity names

DO	DON'T
Prefix the name with standard prefixes, because of the lack of namespaces.	Don't include underscores in names.
Add the postfix "Entity" to the name to prevent conflicts with tables and classes that have the same prefix.	Don't use abbreviations in conceptual names.

DO	DON'T
Give the entity a conceptual name that is aligned with the name in the en-us UI. For example, the conceptual name of the entity that exposes records from the InventTable table should be named **ReleasedProduct**, so that the full name of the entity will be **EcoResReleasedProductEntity**.	

Result: <prefix><ConceptualName>Entity

Data entity field names

DO	DON'T
Give the field name a conceptual name that is aligned with the name in the en-us UI. For example, use **ItemNumber** instead of **ItemID** as the field name to align with the UI, where the label is **Item number**.	Don't include prefixes in field names. For example, a field should not be named **InventLocationId**.
Add the postfix "ID," "Number," and so on, to the name of a field that is part of foreign keys to prevent collision with the navigation properties. For example, use **WarehouseID** instead of **Warehouse** as a field name, because **Warehouse** is the navigation method to the **Warehouse** entity.	Don't include country/region-specific postfixes in field names. For example, a field should not be named **InventoryProfileID_RU**.
	Don't include underscores in field names.
	Don't use abbreviations in field names.

Data entity relation role names

DO	DON'T
Use the plural form when you name roles that have a cardinality that is higher than 1. For example, the foreign key on Customer to Party should have the role name of Customers, because the cardinality from Party to Customer is 0...N.	Don't include prefixes in relation role names. For example, don't use the name **WMSWarehouseLocation**, even though the referenced entity includes the prefix "WMS."
Use the singular form when you name roles that have a cardinality of 0 (zero) or 1. For example, the foreign key on Worker to Person should have the role name of Worker, because the cardinality from Person to Worker is 0..1.	Don't add the postfix "Entity" to relation role names. For example, don't use the role name **WarehouseEntity** in a relationship, even though the referenced entity includes the name "Entity." Instead, use the name **Warehouse**.
Consider adding the role of the relationship as a prefix. For example, to clearly describe the role of the relationship, name a relationship **BuyingLegalEntity** instead of **LegalEntity**.	Don't add country/region-specific postfixes to relation role names. For example, don't use the role name **InventoryProfile_RU**, even though the relationship applies only in an RU country/region format.
	Don't include underscores in relation role names.

Data entity relation names

Do

- Give the relation name the same name as the related role name, in singular form. For example, the relation that describes the foreign key from Customer to Party should be named **Party**.

OData EntityType names

Do

- Give the EntityType a conceptual name. The name should be the same as the conceptual name of the data entity, but without the prefix and without the "Entity" postfix. For example, **EcoResReleasedProductEntity** becomes **ReleasedProduct**.
- Name the EntityType in singular form.

OData EntitySet (Entity Collection) names

Do

- Name the EntitySet in plural form. For example, the EntitySet for the **ReleasedProduct** EntityType is **ReleasedProducts**.

Number of columns in a data entity

Note that Microsoft Excel based import/export supports a maximum of 255 columns. If it is expected for an entity to be able to export/import more than 255 columns, then a non-Excel format must be planned or the entity should have less than or equal to 255 columns.

Build and consume data entities

2/18/2021 • 15 minutes to read • Edit Online

This tutorial shows how to build an entity and how to consume some out-of-band (OOB) entities in an integration scenario. You will also preview how these data entities will be consumed in various integrations scenarios, such as data import and export, integration, and OData services.

When you are ready to build your first entity for production, you will need to:

- Create your own package and model. For more information, see Models and packages.
- Create a new project and set the model property to the one that you just created.

Prerequisites

This tutorial requires that you access an environment by using Remote Desktop, and that you be provisioned as an administrator on the instance.

Throughout this tutorial, baseUrl refers to the base URL of the instance.

- In the cloud environment, the base URL is obtained from Microsoft Dynamics Lifecycle Services (LCS).
- On a local virtual machine (VM), the base URL is `https://usnconeboxax1aos.cloud.onebox.dynamics.com`.
- Download FMLab sample code to C:. For details, see FMLab sample code.

Key concepts

- Developing a data entity in Microsoft Visual Studio
- Enabling a data entity for data management and OData services
- Consuming a data entity in integration scenarios

Business problem

Fleet Management stores customer data in the FMCustomer table and address data in the FMAddressTable table. To access or update customer information, users must access multiple tables. Instead, you can create a business object that functionally represents customer information, and that you can use to build integration solutions.

Building the FMLabCustomer entity

In this section, you must create a data entity for FMLabCustomer in the Fleet Management model. This entity will be used to manage master data through import/export and integration services. The primary data source is FMCustomer, and the secondary data source is FMAddressTable.

Data entity

FMLabCustomerEntity

Data entity fields

NAME	MAPPING
CellPhone	FMCustomer.CellPhone

NAME	MAPPING
DriverLicense	FMCustomer.DriverLicense
Email	FMCustomer.Email
FirstName	FMCustomer.FirstName
LastName	FMCustomer.LastName
CustomerGroup	FMCustomer.CustGroup
AddressLine1	FMAddressTable.AddressLine1
AddressLine2	FMAddressTable.AddressLine2
City	FMAddressTable.City
State	FMAddressTable.State
ZipCode	FMAddressTable.ZipCode
Country	FMAddressTable.Country

Corresponding staging table

Staging tables are used in import/export scenarios to provide intermediary storage during file parsing and transformation. These tables are also used in connector integration scenarios. In many cases, staging table are mapped 1:1 to an entity. The staging table that corresponds to the **FMLabCustomerEntity** entity is named FMLabCustomerStaging.

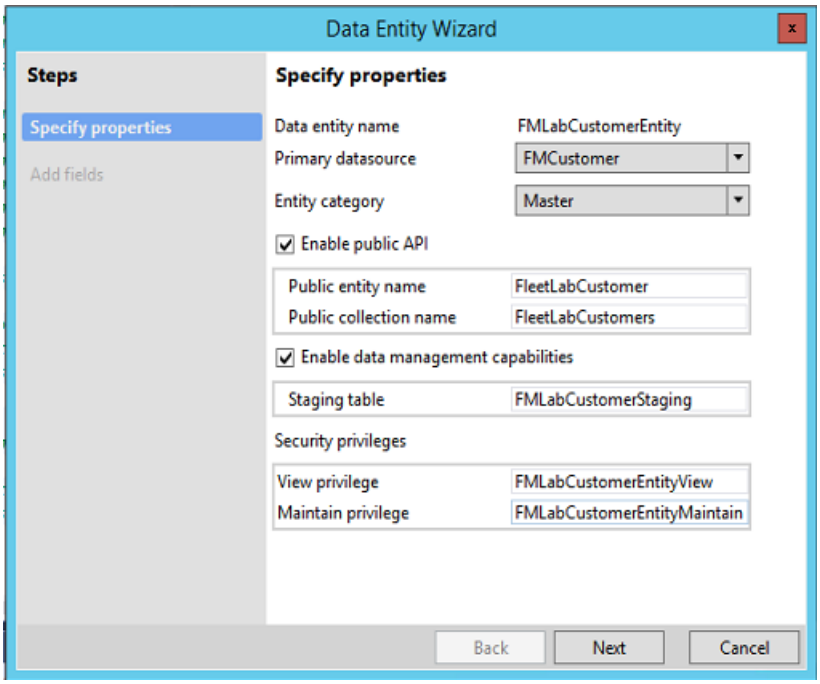
Create a new project

1. In Visual Studio click **File** > **New** > **Project**, and then select **Finance and Operations Project**.
2. Right-click the project, click **Properties**, and verify that the project is in the Fleet Management model. If it isn't, set the **Model** property to **Fleet Management**.

Add a new data entity to your project

1. Create a new entity that is named **FMLabCustomerEntity**. Right-click you project, and then click **Add** > **New item**. The **Add New Item** dialog box opens.
2. Select **Data Entity**, and then set the **Name** property to **FMLabCustomerEntity**.
3. Click **Add**.
4. In the **Data entity** wizard, specify the properties for the data entity that you're creating. Use the values that are shown in the following image.

> **NOTE**
>
> The name of an entity must not have '_' or any numeric digits (0...9). Using these characters may result in mapping errors later.

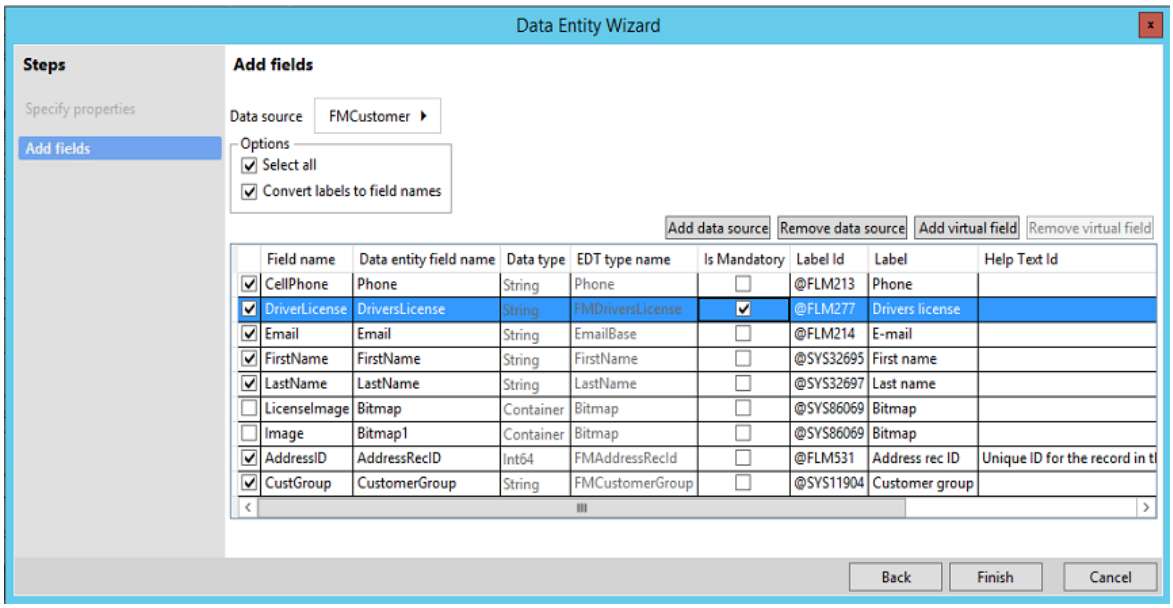


5. Click **Next**. For more information about the function of each property, see "Categories of entities" and "Building an entity" in Data entities overview.

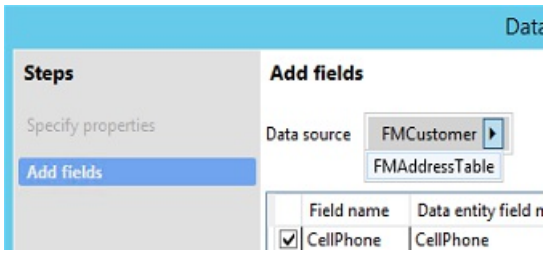
6. Add fields to the new entity from your data source, as shown in the following image. You can add fields from the primary data source, **FMCustomer**. For this entity, clear the check box for the **Image** and **LicenseImage** container types to simplify testing.

7. Rename the data entity fields to reflect public data contract standards, or click **Convert labels to field names** to generate names from the existing labels.

8. On the line for the **DriverLicense** field, select the **Is mandatory** check box. This field will be used as the natural key for the entity.

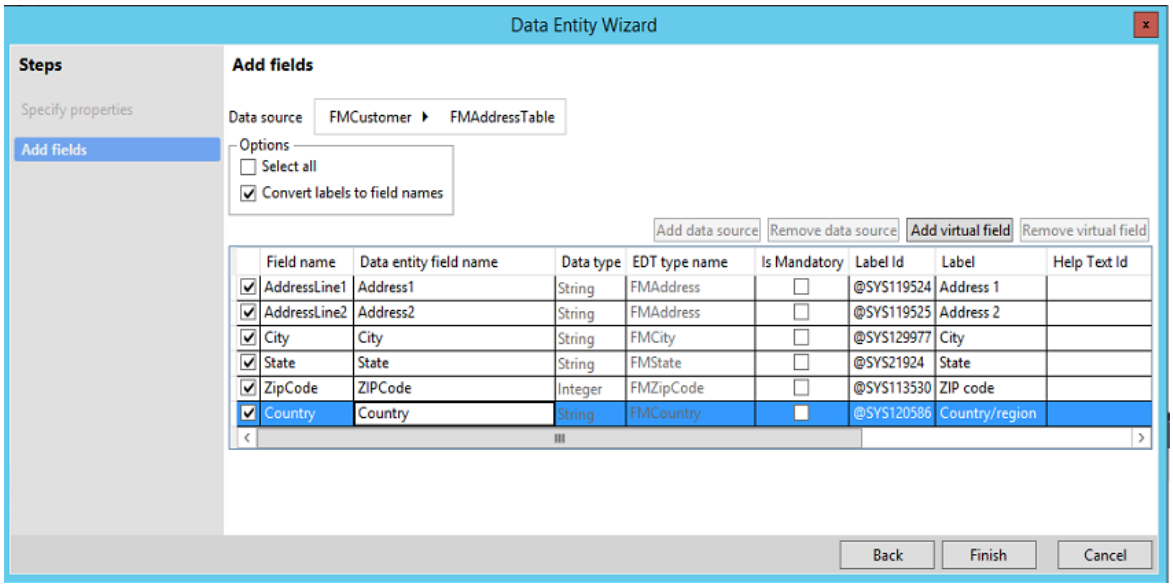


9. In the **Data source** field, select **PrimaryAddress**. Notice that the **PrimaryAddress** data source is automatically added because of automatic expansion or the surrogate foreign key replacement of **AddressID**.

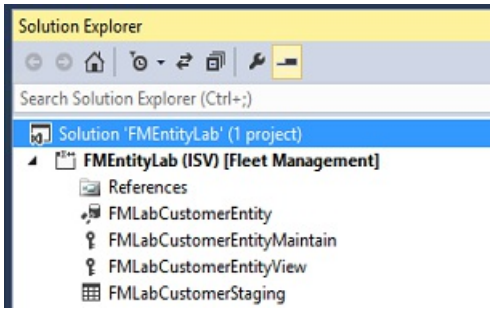


10. Select the fields from the **PrimaryAddress** data source that you want to be part of your entity. Additionally, rename the following fields to reflect proper public data contract naming:

    - PrimaryAddress _AddressLine1 > AddressLine1
    - PrimaryAddress _AddressLine2 > AddressLine2
    - PrimaryAddress _City > City
    - PrimaryAddress _State > State
    - PrimaryAddress _ZipCode > ZipCode
    - PrimaryAddress _Country > Country



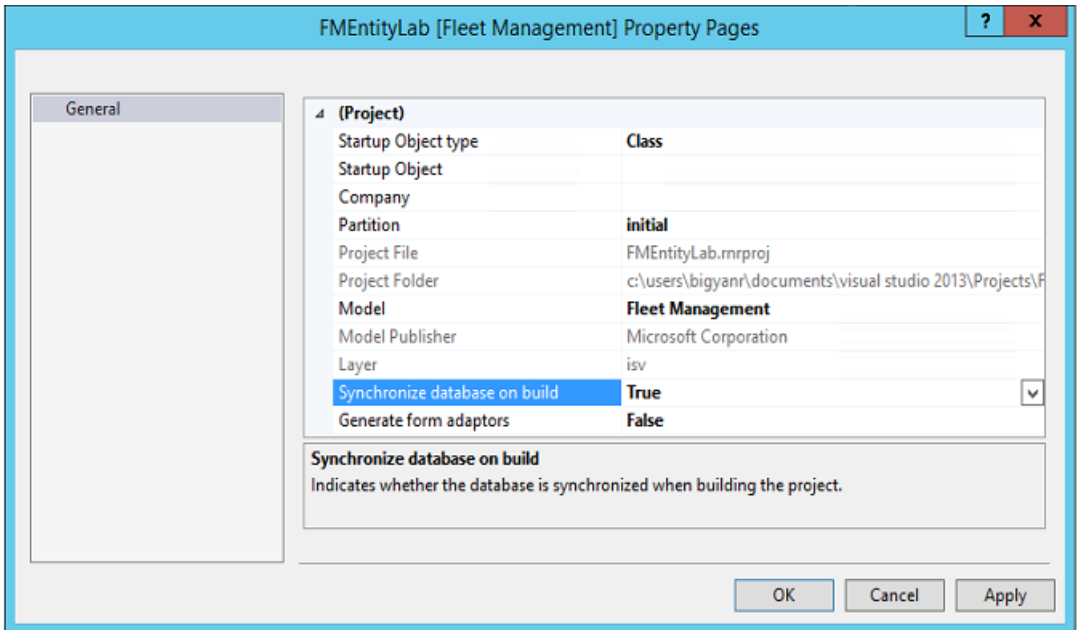
11. Click **Finish**. A data entity item and staging table are added to the project.



Build your project

1. In Solution Explorer, right-click your project, and then click **Properties**.
2. Change the value of the **Synchronize database on build** property to **True**, and then click **OK**. This property must be set only one time per project.

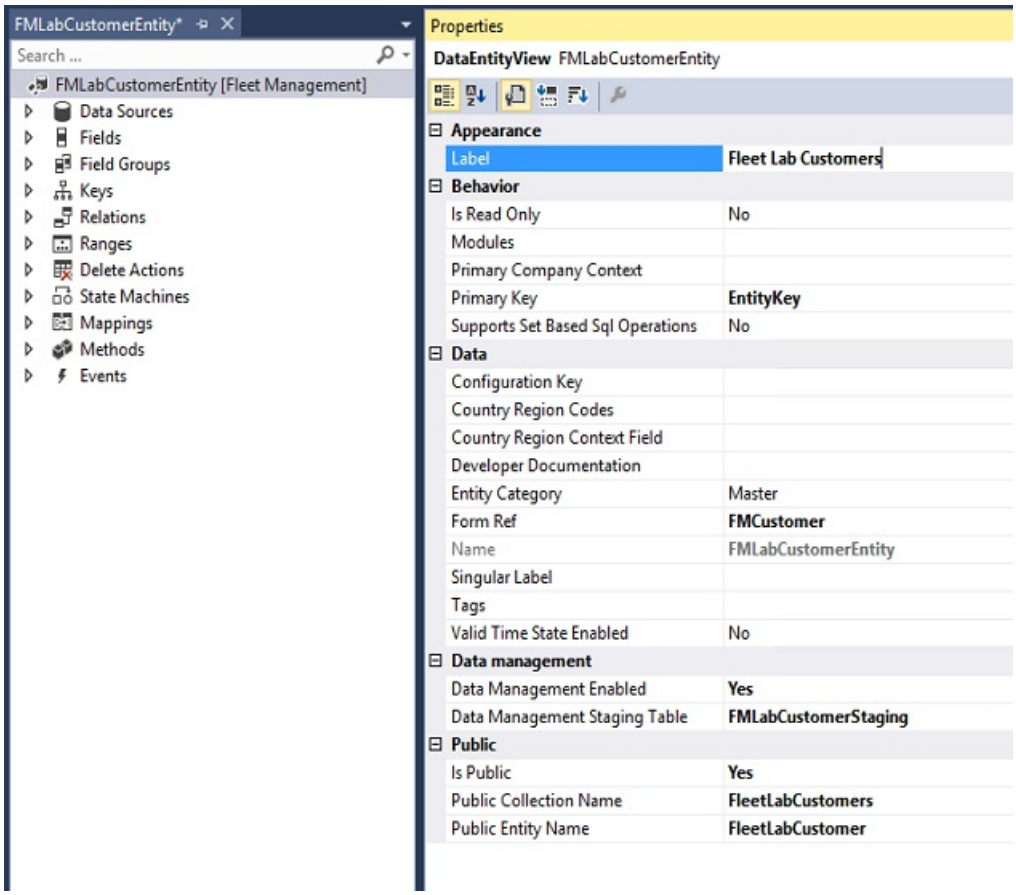
> **NOTE**
>
> Entities are created as views in Microsoft SQL Server, and staging tables are also added. Therefore, you must sync a database when you build entities.



3. On the Visual Studio toolbar, click **Build** > **Build Solution** to build the project.
4. Verify that the build doesn't contain any errors. At this point in the tutorial, warnings are allowed.

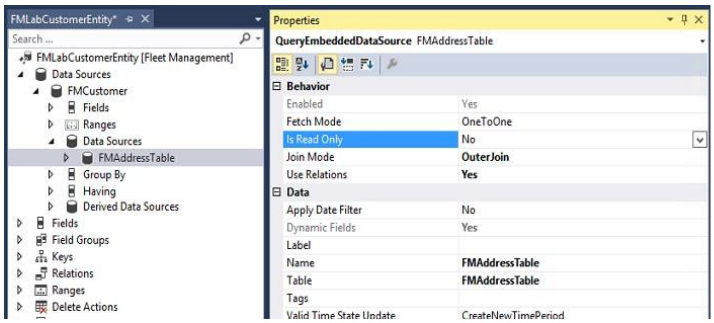
Visually validate and customize an entity

1. In Solution Explorer, right-click the **FMLabCustomerEntity** node, and then click **Open**. The designer for the entity opens in the middle pane.
2. Validate the properties of the **FMLabCustomerEntity** entity. Select the entity in Solution Explorer, and compare the **Properties** pane values to the following image.
3. Set the **Label** property to **Fleet Lab Customers**.

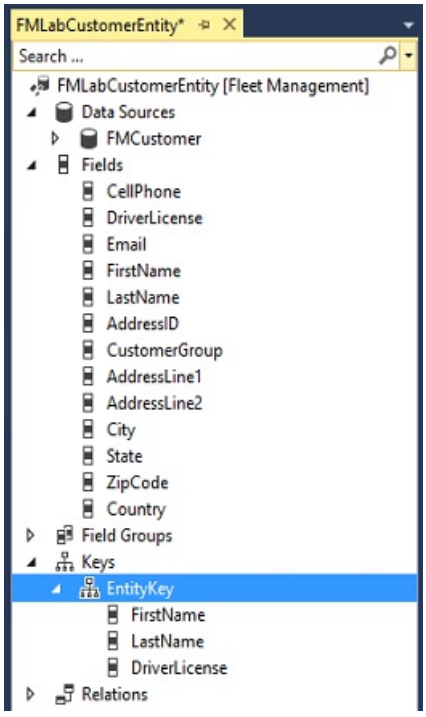


4. In the left pane, click **Data Sources** > **FMCustomer** > **Data Sources** > **FMAddressTable**.

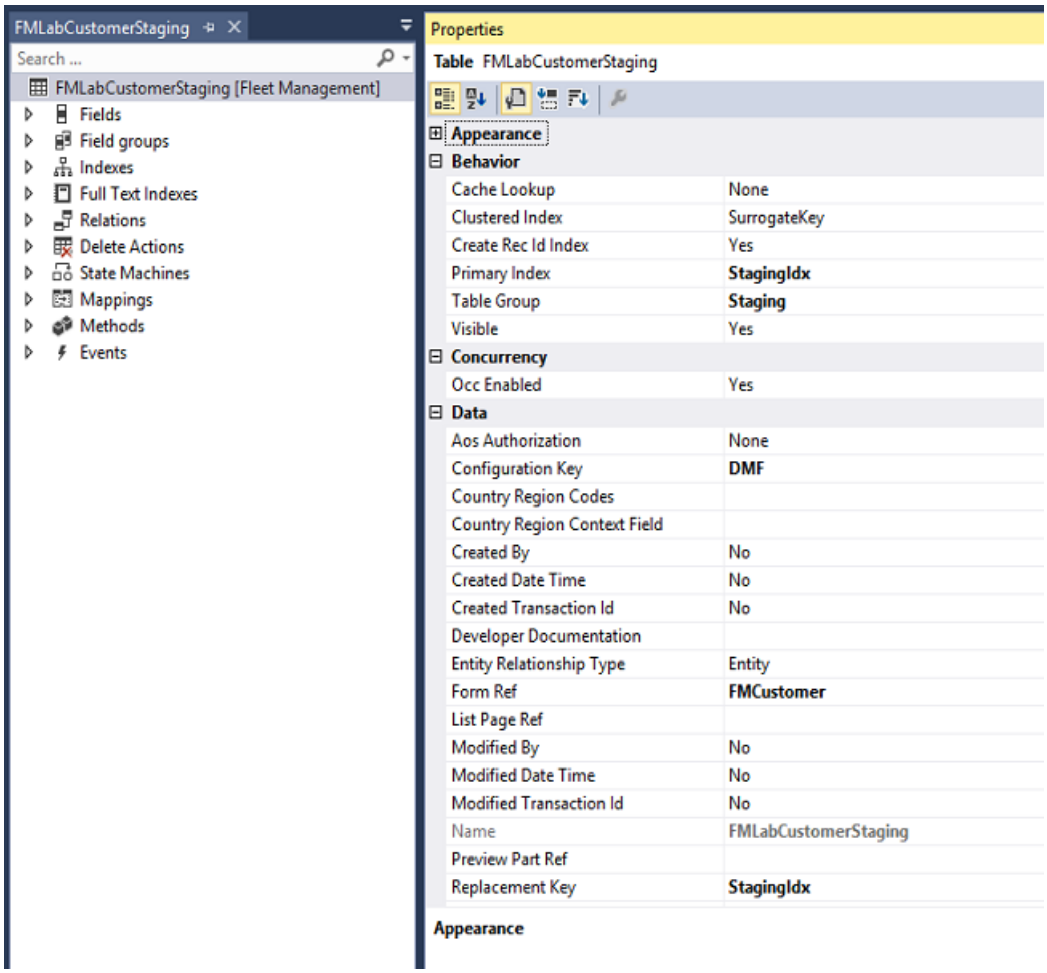
5. Change the **Is Read Only** property to **No**. This is a known issue. Eventually, the value will be set to **Yes** or **No** automatically, based on the type of join. The value should be **Yes** for composition scenarios, and **No** for associations (surrogate foreign key expansions). This property enables the data source to be read/write.



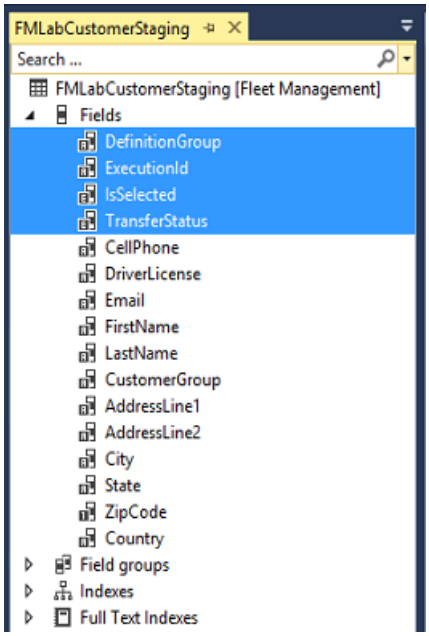
6. In the **FMLabCustomerEntity** designer, click **Keys** > **EntityKey**, and then expand the **Fields** node. Verify that the list of fields matches the following image.



7. To visually validate the staging table that will be used for import/export, open the **FMLabCustomerStaging** table in the designer, and then select the **FMLabCustomerStaging** node.



8. Click **FMLabCustomerStaging** > **Fields**. In the following image, the standard fields of the staging tables are selected. All entity staging tables have these standard fields. The image also shows the data fields that belong on this data entity.



9. In Solution Explorer, right-click your project, and then select **Rebuild** to rebuild and synchronize the project.

Testing data entities

Entities can be tested by using various methods in X++, through data import/export, or through integrations. In this section, we'll explore scenarios for validating entities.

Test the entity by using X++ code

One of the most common ways of interacting with data entities is through X++, by using a unit test or a runnable job to validate that the entities have been built. In this example, we will use a runnable job.

1. In Solution Explorer, click **Add** > **New item** > **Runnable class** to add a runnable class to your project.
2. Copy and paste the following code into the class to test your data entity.

```
public static void main(Args _args)
{
    FMLabCustomerEntity customer;
    str license = "License";
    Random r = new Random();
    int rand = r.nextInt();
    license = license + int2str(rand);

    //Create a new record in FM lab customer entity
    customer.clear();
    customer.FirstName = "Bob";
    customer.LastName = "Smith";
    customer.DriverLicense = license;
    customer.insert();

    info(strfmt("Tried to insert customer '%1 %2' with license %3",
        customer.FirstName, customer.LastName, customer.DriverLicense));

    //Display newly created record
    select forupdate customer where customer.DriverLicense==license;
    info(strfmt("Found newly created customer '%1 %2' with license %3",
        customer.FirstName, customer.LastName, customer.DriverLicense));

    //Now delete the record from the entity
    customer.delete();
    select customer where customer.DriverLicense==license ;
    info(strfmt("Deleted customer does not exist: license- %1", customer.DriverLicense));
}
```

3. Run the code in debugger to set it as a startup object.
4. To validate the entity, view the Infolog in the debugger window or in notifications on the website. You will see that three successful messages are logged. You will also see the actions that were taken.

Importing data by using entities

Data entities that have the **Data Management Enabled** property can be used to import and export data in various file formats. In this section, you will import data in a CSV file format for the **FMLabCustomer** entity.

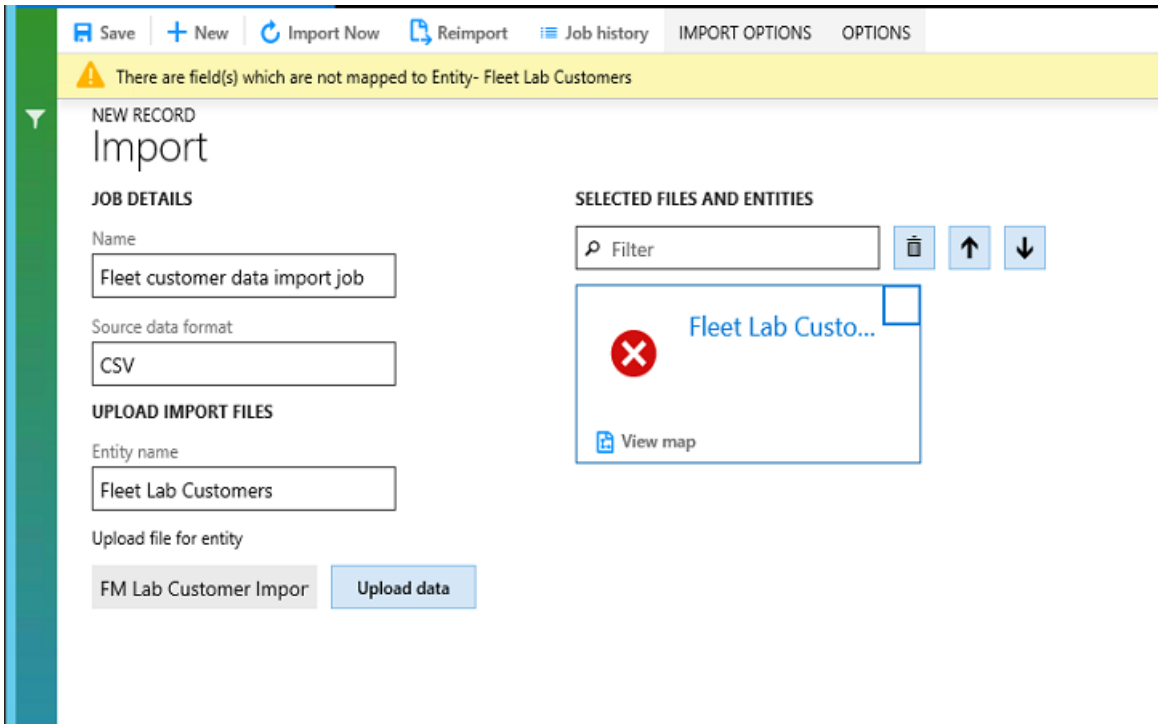
File import

After you create your data entity, you can validate import/export.

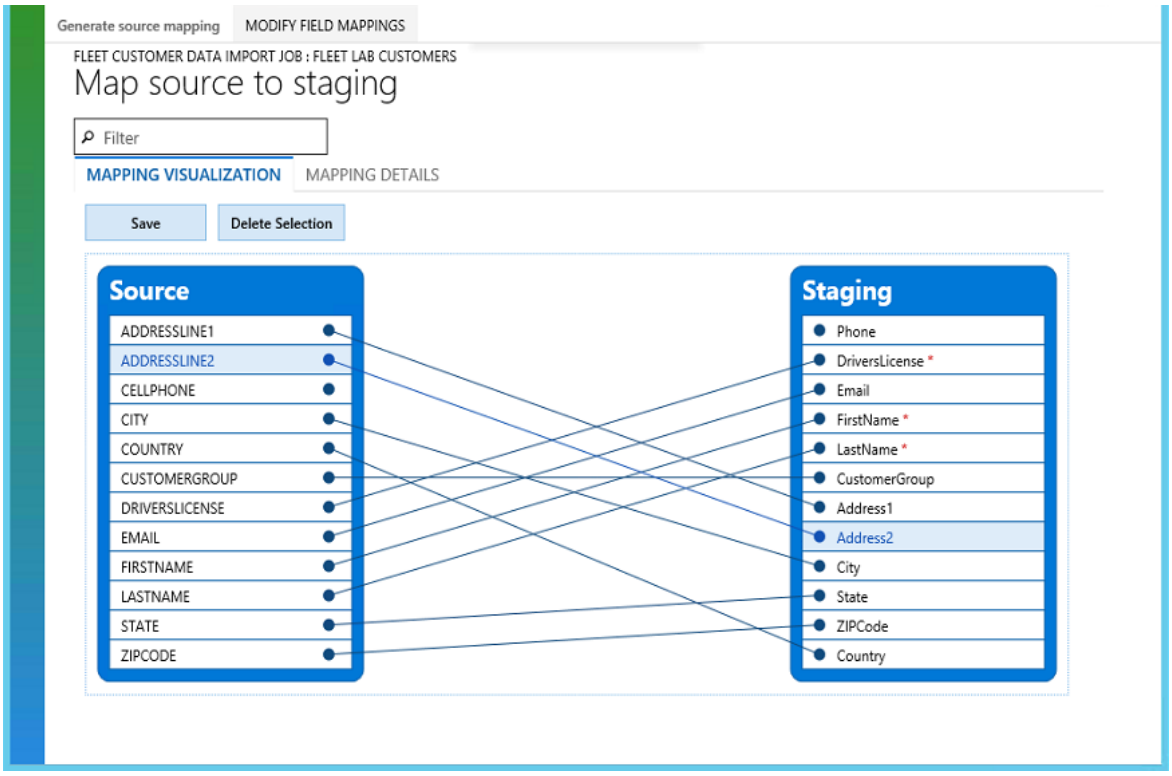
1. Create a sample CSV file that you can import. Copy the following text, and save it as **FM Lab Customer Import.csv**.

```
CELLPHONE,DRIVERSLICENSE,EMAIL,FIRSTNAME,LASTNAME,CUSTOMERGROUP,ADDRESSLINE1,ADDRESSLINE2,CITY,STATE,
ZIPCODE,COUNTRY(999) 555-0100,S615-3939-2349,chris.spencer@adatum.com,Chris,Spencer,adv\_mem\_1,444
Main Street,,Orlando,FL,77899,US(188) 555-0101,S615-3939-
2350,Ichiro.lannin@blueyonderairlines.com,Ichiro,Lannin,non\_mem\_1,12 Long Street,,New York
City,NY,99087,US(777) 555-0102,S615-3939-2351,josh.smith@fourthcoffee.com,Josh,Smith,adv\_mem\_1,9606
122th Avenue,,Sydney,TX,99874,US(456) 555-0103,S615-3939-
2352,Vince@fabrikam.us,Vince,Ahmed,non\_mem\_1,123 Microsoft Way,Unit 87,Seattle,WA,90001,US(345)
555-0104,S615-3939-2353,tony.parker@lucernepublishing.com,Tony,Parker,non\_mem\_1,12012 11th PLNE,Apt
160,San Francisco,CA,75645,US(312) 555-0105,S615-3939-
2354,Julia@fineartschool.net,Julia,Natarajan,exec\_mem\_1,449 Long Street,Apt
160,Bruxelles,ID,34213,US
```

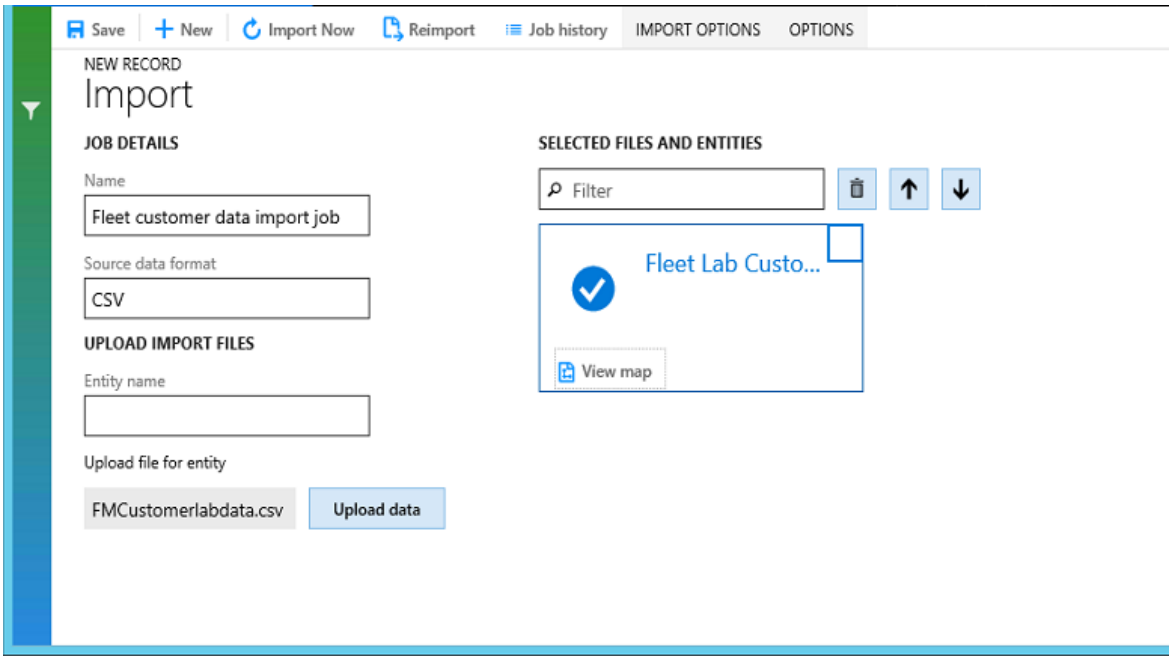
2. Click **User Dashboard** > **Data management**.
3. In the **Data Management** workspace, click the **Import** tile.
4. On the **Import** page, enter the import details, as shown in the following image.



5. Click the **Upload data** button next to the **Upload file for entity** field, and select the CSV file that you created.
6. After the file is uploaded, you will notice that the entity is added to the middle section. You will also receive an error that states that the mapping isn't valid. A few fields aren't mapped correctly between the source file and the target entity.
7. In the entities list, click **View map**.
8. **AddressLine1** and **AddressLine2** are two fields in the source that aren't mapped to target fields. In the visual mapper, or details view, map these fields as follows, and then click **Save**:
   - AddressLine1 – Address1
   - AddressLine2 – Address2



9. Click the **Back** button in your browser to go back to the **Import job** page. The check mark in the entities list indicates that the entity is now ready for import.



10. Click **Import Now**. After the import is completed, the job status page opens.

Consuming entities by using OData

In this section, you will learn how to expose and consume an entity for OData. Before you begin, verify that the Fleet demo data is loaded from the client: [baseURL]/?f=FMSetup

Review the FleetRental entity and add a navigation property for OData

You will review the existing **FleetRental** entity and then create a relationship from one data entity to another. This relationship will be used as a navigation property for OData entities.

1. In Solution Explorer, verify that you're in the **FMEntityLab** project.
2. In Application Explorer, search for **FMRentalEntity**, right-click it, and then select **Add to Project**.

3. In Application Explorer, search for **FMCustomerEntity**, right-click it, and then select **Add to project**.
4. In Solution Explorer, right-click **FMRentalEntity**, and then select **Open**.
5. In the view designer, select the root node, **FMRentalEntity**, and review the following properties.

PROPERTY	VALUE	DESCRIPTION
IsPublic	Yes	When this property is set to **Yes**, the entity is visible by using the OData application programming interface (API).
Public Entity Name	FleetRental	The name that will be used in the OData APIs for **EntityType**.
Public Collection Name	FleetRentals	The name that will be used for the OData collection entity.

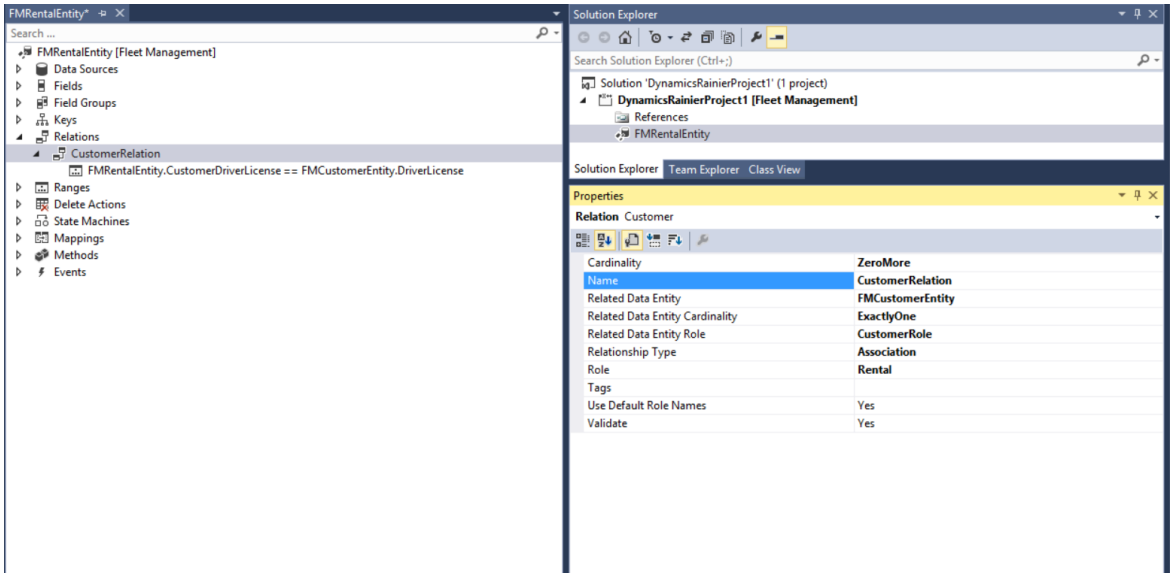
6. In the view designer, expand the **Relations** node.
7. Select **Customer Relation**, and then click **Delete**.
8. Right-click **Relations**, and then select **New** > **Relation**.
9. Select **Relation1**, and set the following properties.

PROPERTY	VALUE
Cardinality	ZeroMore
Name	CustomerRelation
Related Data Entity	FMCustomerEntity
Related Data Entity Cardinality	ExactlyOne
Related Data Entity Role	CustomerRole
Relationship Type	Association
Role	Rental

10. In the view designer, right-click the **CustomerRelation** node, and then select **New** > **Normal**.
11. Right-click the new node under **CustomerRelation**, and then select **Properties**.
12. Set the following properties.

PROPERTY	VALUE
Field	**CustomerDriverLicense**This is the foreign key field on **FMRentalEntity**.
Related Field	**DriverLicense**This is the unique key on **FMCustomerEntity**.

The following image shows the relation in Visual Studio.



13. On the **BUILD** menu, click **Build Solution** to save your changes and build the project. You can view the build progress in the **Output** window.

14. To update the OData endpoint with the changes, you must run an **iisreset** command. Open a **Command Prompt** window as an administrator, and enter **iisreset**.

You've now created a navigation property between **FMRentalEntity** and **FMCustomerEntity**.

Use standard OData syntax to retrieve data

In this section, you will use some of the standard OData syntax to navigate and query the OData entities that are exposed in the Fleet Management model. First, follow these steps to enable Internet Explorer to view JSON formatted data.

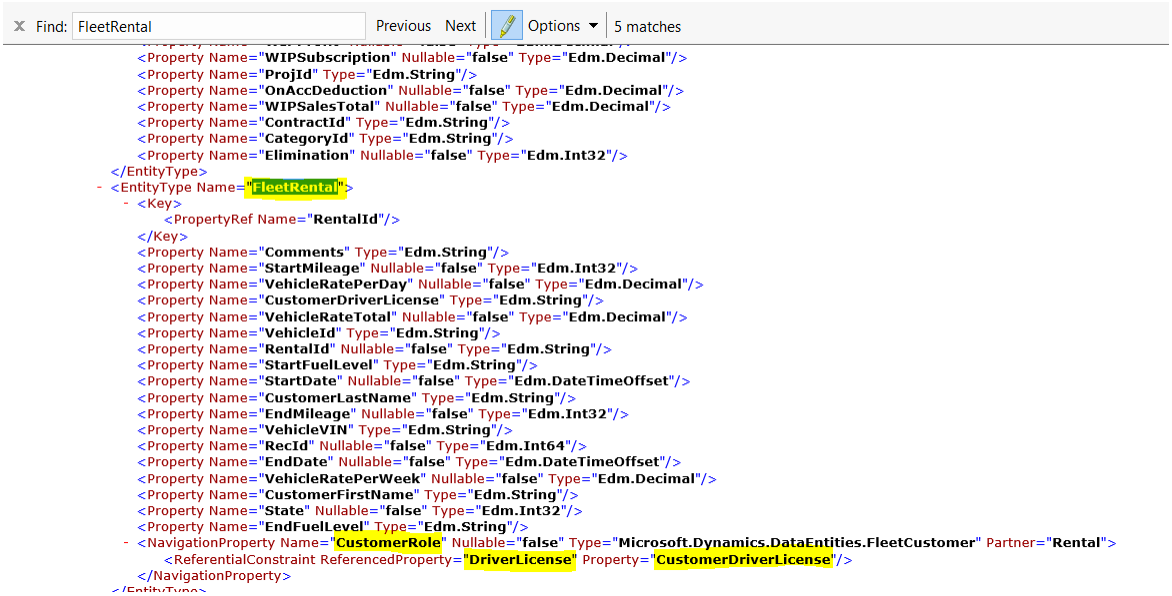
1. Close all Internet Explorer windows.
2. Go to C:\FMLab, and select and double-click the json-ie.reg file.
3. In the **Registry Editor** dialog box, click **Yes**.
4. Click **OK**.

You can now use Internet Explorer to explore some OData URIs.

1. Start Internet Explorer, and enter the following URL in the address bar: [baseURL]/data/$metadata You will see all the metadata that is associated with OData entities.

> **NOTE**
>
> The metadata might take a few minutes to appear the first time that you access it. In the XML, you can see all of the properties and navigation properties associated with the OData entities.

2. In the browser, find **FleetRental**. The following image shows the metadata of the **FleetRental** entity, together with the new relationship, **NavigationProperty**.



3. To view all the customers in the Fleet Management application in JSON format, enter the following URL into the address bar of your browser: [baseURL]/data/FleetCustomer

> **NOTE**
>
> Entity names are case-sensitive.

4. If you don't want to retrieve all properties for the customers, you can retrieve just selected properties. For example, to retrieve only **FirstName** and **LastName**, enter the following URL: [baseURL]/data/FleetCustomers?$filter=FirstName.LastName

5. You can also apply filters. For example, to filter on all customers where **FirstName**=**Phil**, enter the following URL: [baseUrl]/data/FleetCustomers?$filter=FirstName%20eq%20'Phil'

> **NOTE**
>
> These URLs won't work if you copy and paste them. You must manually enter them in the address bar.

6. To retrieve all the **Rental** records, together with all details of the customers, enter the following URL: [baseURL]/data/FleetRentals?$expand=CustomerRole The following example shows a **Rental** record, together with details of the linked customer, in JSON format.

```
"@odata.context":"https://testax32aos.cloud.test.dynamics.com/en/data/$metadata#FleetRentals","value"
:
{
    {
        "@odata.etag":"W/"JzEsNTYzNzE0NDU3NjsxLDU2MzcxNDQ1NzY7MTc4NjA2OTg1Niw1NjM3MTQ0NjA1Jw==",
        "Comments":"","StartMileage":0,"VehicleRatePerDay":40,"CustomerDriverLicense":
        "S468-3184-6541","VehicleRateTotal":280,"VehicleId":"Litware_LitwareFour_1",
        "RentalId":"000001",
        "StartFuelLevel":"Full","StartDate":"2010-04-09T00:00:00Z","CustomerLastName":"Spencer",
        "EndMileage":0,"VehicleVIN":"2J4FY19P0NJ710529",
        "RecId":5637144576,"EndDate":"2010-04-16T00:00:00Z",
        "VehicleRatePerWeek":270,"CustomerFirstName":"Phil","State":3,
        "EndFuelLevel":"","CustomerRole":

        {"@odata.etag":"W/"JzEsNTYzNzE0NDU3NjsxLDU2MzcxNDQ1NzYn"",
        "CellPhone":"(999) 555-0100",
        "DriverLicense":"S468-3184-6541","AddressLine2":"",
        "State":"FL","Country":"US","FirstName":"Phil",
        "Email":"phil.spencer@adatum.com","CustomerGroup":
        "adv_mem_1","AddressLine1":"167 BBN Way","City":"Orlando",
        "ZipCode":77899,"RecId":5637144576,"LastName":"Spencer"
    }
}
```


Add an action to OData entity

Actions provide a way to inject behaviors into the data model. In Dynamics 'AX 7,' you add actions by adding a method to the data entity and then decorating the method with specific attributes. In this section, we'll walk through the steps for adding an action.

1. In Solution Explorer, right-click **FMRentalEntity**, and then select **View code**.
2. Copy the following code lines, and paste them into the **Code** window.

```
public class FMRentalEntity extends common
{
    [SysODataActionAttribute("ReturnRental", true)]
    public str ReturnRental()
    {
        //do something
        return "Rental was successfully returned. Thanks for your business";
    }
}
```

3. On the **BUILD** menu, click **Rebuild Solution** to save your changes and build the project. You can view the build progress in the **Output** window.
4. To update the OData endpoint with the changes, you must run an **iisreset** command. Open a **Command Prompt** window as an administrator, and enter **iisreset**.

   The action that you just added can be invoked through code, as you will see in the next section.

Consume the OData API from an external console application

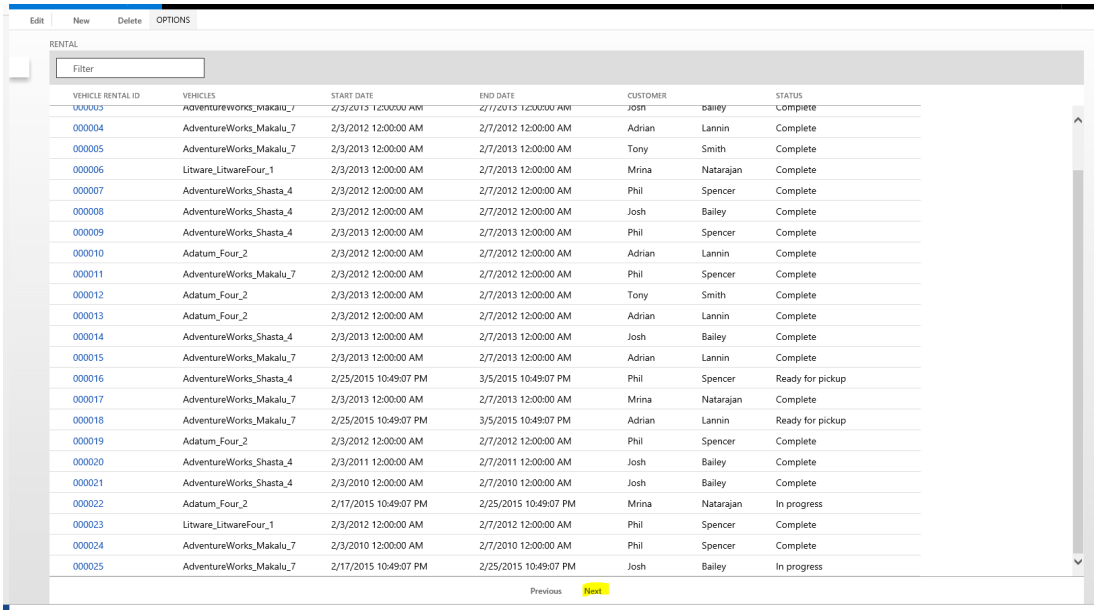
In this section, you will use a console application to consume the OData endpoints that are exposed in the Fleet Management application. The console application first creates a new customer and then creates a new reservation for that customer. This tutorial shows how easy it is to use OData together with standard .NET Windows Communication Foundation (WCF) data service libraries to integrate with Dynamics AX.

1. Start a new instance of Visual Studio.
2. On the **File** menu, click **Open** > **Project/Solution**.

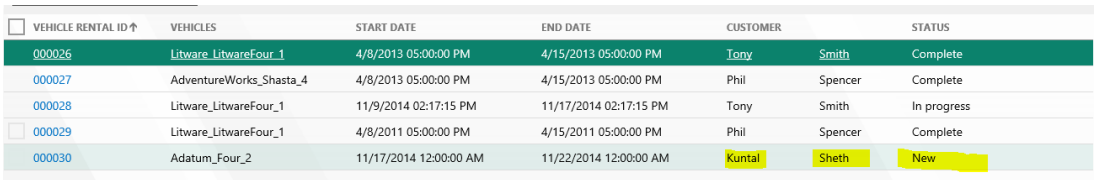
3. In the **Open Project** dialog box, browse to **C:\FMLab\Odata4ConsoleApplication**, and then select **Odata4ConsoleApplication.csproj**.
4. Click **Open**. The **Odata4ConsoleApplication** project appears in Solution Explorer.
5. In Solution Explorer, double-click **OdataProxyGenerator.tt**.
6. In the code editor, replace the following string with your organization's URL.

```
<baseURL> public const string MetadataDocumentUri = "<baseURL>/data/"
```

7. Save the OdataProxyGenerator.tt file.
8. In the **Save of Read-only file** dialog box, click **Overwrite**. The proxy class for the OData metadata endpoint is generated. This operation might take a few minutes.
9. In Solution Explorer, double-click **Program.cs**.
10. Replace the value of the **dynamicsBaseUri** variable with your organization's URL.
11. Verify that there is a final closing slash (/) in the URL, and then click **Save**.
12. In the **Save of Read-only file** dialog box, click **Overwrite**.
13. Press F5 to run the application, and then follow the instructions in the output console window. The application might prompt you for your Dynamics AX credentials. After the application has run, the new customer and the corresponding reservation are created.
14. Follow these steps to verify that the new reservation appears on the **Rental** page:
    a. Start Internet Explorer, and enter the following URL in the address bar: [baseURL]/?mi=FMRental The **FMRental** page shows the list of rentals.



    b. At the bottom of the list, click **Next** to view the next page. On this page, you can see that the reservation was created for the new customer that you added.



This completes the walkthrough, where you've seen an external client interacting with the Fleet Management model by using OData endpoint.

Casing rules in data entities

XML format

During an export, the entity name and the field names are exported in uppercase. If there is a need to apply a transformation, the transformation must use uppercase in all references.

During an import, data management accepts input file in any casing. However, care must be taken to have the same format for a given attribute/element in the file. When applying a transformation, ensure that the transformation is using the same casing rules in all references as in the incoming file.

Excel format

During an export, column names will be exported in uppercase. Imports are not case sensitive.

CSV format

During an export, column names will be exported in uppercase. Imports are not case sensitive.

Tips and tricks

Max join limits

During entity development, ensure that the overall structure of the entity does not exceed the max join limit of 26. This is the default limit. Increasing the join limit is not recommended because it can have unintended consequences. If this limit is exceeded, the entity will most likely fail to process records and will result in the following SQL error. We also recommend managing the total number of columns in the entity to avoid this error.

```
Cannot create a row of size xxx which is greater than the allowable maximum row size of 8060
```

Exporting container fields

If an entity has container fields and these fields need to be exported, the entity must implement **getFieldsToBeConvertedToFile** to allow each container field to export its data value to a separate file. This allows for container fields to be exportable and at the same time, prevents making the entity export file (core entity data without the container fields) unreadable. If **getFieldsToBeConvertedToFile** is not implemented, then these fields will not be exported but the rest of the entity data will export as usual.

Additional resources

Develop entities for data migration

Behavioral properties on data entities

2/18/2021 • 3 minutes to read • Edit Online

Every data entity has properties that let you override the same property values on the tables or views that are the data sources of that entity. Your choices affect the behavior of the entity. In the following table, the first column lists the properties that are discussed in this topic. The top row lists the levels where the property is found in the entity designer. The levels are listed in order of increasing granularity: the data source level is more granular than the entity level but less granular than the field level.

	ENTITY LEVEL	DATA SOURCE LEVEL	FIELD LEVEL
ReadOnly	Applies	Applies	.
AllowEdit	.	.	Applies
AllowEditOnCreate	.	.	Applies
Mandatory	.	.	Applies

Entity level

In the designer for your data entity, when you click the name at the root node, the **Properties** pane includes the **Is Read Only** property. The following table describes the behavioral differences between the **Yes** and **No** values of this property.

GROUP	PROPERTY NAME	DISPLAY NAME	VALUES	DEFAULT	DESCRIPTION

GROUP	PROPERTY NAME	DISPLAY NAME	VALUES	DEFAULT	DESCRIPTION
Behavior	IsReadOnly	Is Read Only	No, Yes	No	• **No:** Data modification operations (CUD) *are* allowed, *unless* an individual data source node in the entity's designer is set to **IsReadOnly** = **Yes**. • **Yes:** Only read operations are allowed, regardless of the **IsReadOnly** settings on the individual data source nodes in the entity's designer.

You would set **IsReadOnly** to **Yes** for entities that are consumed mainly for export.

Data source level

If a data entity has three data sources, you might want to allow processes to use the entity to modify the data in one of the data sources but not in the other two. A read-only data source can be used for lookup purposes. You can use the entity designer to achieve this extra degree of granular control. Under the entity's **Metadata** > **Data Sources** node, you can select an entity node and then set the **IsReadOnly** property value for that one data source. The following table describes the interaction between the **IsReadOnly** settings at the data source level and the entity level.

GROUP	PROPERTY NAME	DISPLAY NAME	VALUES	DEFAULT	DESCRIPTION

GROUP	PROPERTY NAME	DISPLAY NAME	VALUES	DEFAULT	DESCRIPTION
Behavior	IsReadOnly	Is Read Only	No, Yes	No	• **No:** Data modification operations (CUD) *are* allowed on the data source, *unless* **IsReadOnly** is set to **Yes** at the entity level. • **Yes:** Only operations are allowed, regardless of the **IsReadOnly** setting on the entity.

Field level

At the field level, the **AllowEdit** and **AllowEditOnCreate** properties are available instead of an **IsReadOnly** property. The two **Allow** properties include **Auto** as a third available value. The **Auto** value inherits the value that is on the field in the underlying table.

> **NOTE**
> The **Auto** value isn't available for unmapped fields, such as computed or virtual fields.

GROUP	PROPERTY NAME	DISPLAY NAME	VALUE	DEFAULT	DESCRIPTION
Behavior	AllowEditOnCreate	Allow edit on create	Auto, No, Yes	Auto	• **Auto:** The property is inherited from the underlying table field. > [!NOTE] The **Auto** value

GROUP	PROPERTY NAME	DISPLAY NAME	VALUE	DEFAULT	DESCRIPTION
					isn't available for unmapped fields, such as computed or virtual fields. - **No**: Users aren't allowed to modify the data for this field in a new record. - **Yes**: Users are allowed to modify the data for this field for a new record. This behavior is enforced for all consumers – X++, OData, and so on. [!IMPORTANT] The **No** and **Yes** values do *not* override the setting on the field in the underlying table.

GROUP	PROPERTY NAME	DISPLAY NAME	VALUE	DEFAULT	DESCRIPTION
Behavior	AllowEdit	Allow edit	Auto, No, Yes	Auto	The behavior is the same as the behavior for **AllowEditOnCreate**, but it applies to updates to *existing* records instead of new records that are being created. This behavior is enforced for all consumers – X++, OData, and so on.
Behavior	Mandatory	Mandatory	Auto, No, Yes	Auto	**Auto:** The property is inherited from the underlying table field. This behavior is enforced for all consumers – X++, OData, and so on. [!IMPORTANT] The **No** and **Yes** values do *not* override the setting on the field in the underlying table.

Computed columns and virtual fields in data entities

2/18/2021 • 6 minutes to read • Edit Online

This article provides information about computed and virtual fields, which are the two types of unmapped fields that a data entity can have. The article includes information about the properties of unmapped fields, and examples that show how to create, use, and test them.

The sample code is targeted towards creating or modifying an entity that is a part of solution that you own. Extending an existing entity requires slight modifications.

Overview

A data entity can have additional *unmapped* fields beyond those that are directly mapped to fields of the data sources. There are mechanisms for generating values for unmapped fields:

- Custom X++ code
- SQL executed by Microsoft SQL Server

The two types of unmapped fields are computed and virtual. Unmapped fields always support read actions, but the feature specification might not require any development effort to support write actions.

Computed field

- Value is generated by an SQL view computed column.
- During read, data is computed by SQL and is fetched directly from the view.
- For writes, custom X++ code must parse the input value and then write the parsed values to the regular fields of the data entity. The values are stored in the regular fields of the data sources of the entity.
- Computed fields are used mostly for reads.
- If possible, it's a good idea to use computed columns instead of virtual fields, because they are computed at the SQL Server level, whereas, virtual fields are computed row by row in X++.

Virtual field

- Is a non-persisted field.
- Is controlled by custom X++ code.
- Read and write happens through custom X++ code.
- Virtual fields are typically used for intake values that are calculated by using X++ code and can't be replaced by computed columns.

Properties of unmapped fields

CATEGORY	NAME	TYPE	DEFAULT VALUE	BEHAVIOR

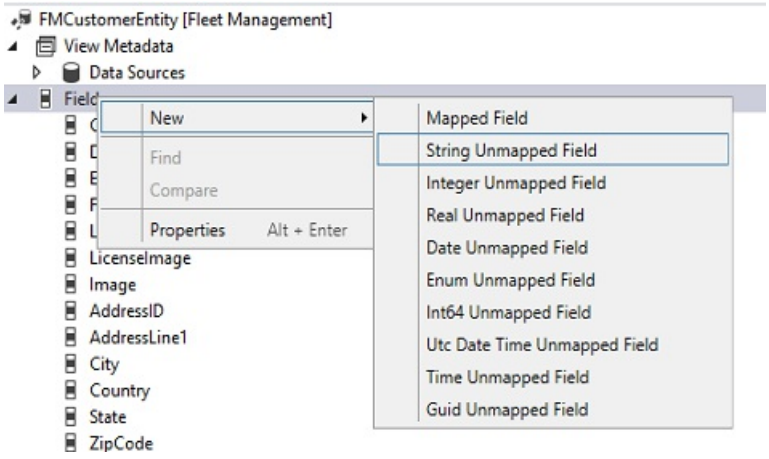
CATEGORY	NAME	TYPE	DEFAULT VALUE	BEHAVIOR
Data	IsComputedField	NoYes	Yes	• **Yes** – The field is synchronized as a SQL view computed column. Requires an X++ method to compute the SQL definition string for the column. The virtual column definition is static and is used when the entity is synchronized. After that, the X++ method is not called at run time. • **No** – The field is a true virtual field, where inbound and outbound values are fully controlled through custom code.
Data	ComputedFieldMethod	String		A static **DataEntity** method in X++ to build the SQL expression that will generate the field definition. This property is disabled and irrelevant if the property **IsComputedField** is set to **No**. The method is required if the property **IsComputedField** is set to **Yes**.
Data	ExtendedDataType	String		

Example: Create a computed field

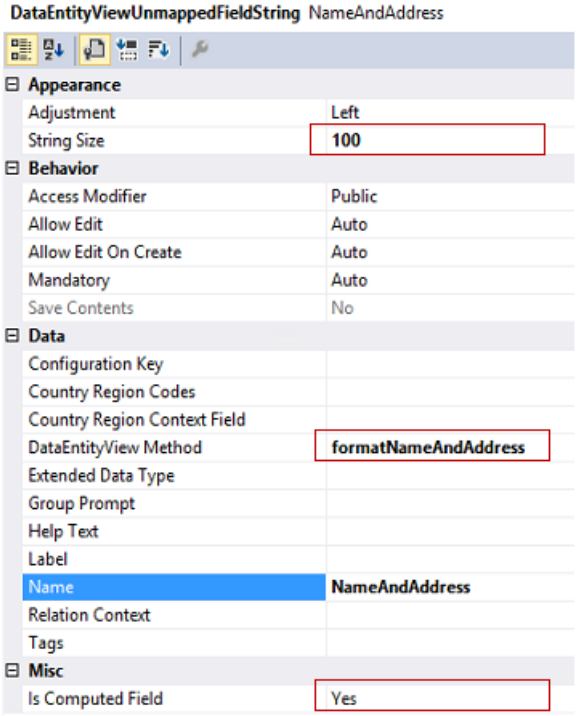
In this example, you add a computed field to the **FMCustomerEntity** entity. For reads, the field combines the name and address of the customer into a nice format. For writes, your X++ code parses the combined value into

its separate name and address values, and then the code updates the regular name and address fields.

1. In Microsoft Visual Studio, right-click your project, and add the existing **FMCustomerEntity**.
2. In Solution Explorer, right-click the **FMCustomerEntity** node, and then click **Open**.
3. In the designer for **FMCustomerEntity**, right-click the **Fields** node, and then click **New** > **String Unmapped Field**.



4. Rename the new field **NameAndAddress**.
5. Update properties of the **NameAndAddress** unmapped field, as shown in the following screenshot.



6. Go to **FMCustomerEntity** > **Methods**. Right-click the **Methods** node, and then click **New**. Ensure that the method name matches the **DataEntityView Method** property value of the unmapped computed field.
7. Paste the following X++ code into the method. The method returns the combined and formatted **NameAndAddress** value.

> **NOTE**
>
> The **server** keyword is required.

```
private static server str formatNameAndAddress()   // X++
{
    DataEntityName      dataEntityName= tablestr(FMCustomerEntity);
    List                fieldList = new List(types::String);
    ////Format name and address to look like following
    ////John Smith, 123 Main St, Redmond, WA 98052
    fieldList.addEnd(SysComputedColumn::returnField(DataEntityName, identifierstr(FMCustomer),
fieldstr(FMCustomer, FirstName)));
    fieldList.addEnd(SysComputedColumn::returnLiteral(" "));
    fieldList.addEnd(SysComputedColumn::returnField(DataEntityName, identifierstr(FMCustomer),
fieldstr(FMCustomer, LastName)));
    fieldList.addEnd(SysComputedColumn::returnLiteral("; "));
    fieldList.addEnd(SysComputedColumn::returnField(DataEntityName, identifierstr(FMAddressTable),
fieldstr(FMAddressTable, AddressLine1)));
    fieldList.addEnd(SysComputedColumn::returnLiteral(", "));
    fieldList.addEnd(SysComputedColumn::returnField(DataEntityName, identifierstr(FMAddressTable),
fieldstr(FMAddressTable, City)));
    fieldList.addEnd(SysComputedColumn::returnLiteral(", "));
    fieldList.addEnd(SysComputedColumn::returnField(DataEntityName, identifierstr(FMAddressTable),
fieldstr(FMAddressTable, State)));
    fieldList.addEnd(SysComputedColumn::returnLiteral(", "));
    fieldList.addEnd(SysComputedColumn::cast(
        SysComputedColumn::returnField(DataEntityName, identifierstr(FMAddressTable),
fieldstr(FMAddressTable, ZipCode)), "NVARCHAR"));
    return SysComputedColumn::addList(fieldList);
}

T-SQL for the computed column.

( Cast (( ( T1.firstname ) + ( N' ' ) + ( T1.lastname ) + ( N'; ' ) +
    ( T5.addressline1 )
    + ( N', ' ) + ( T5.city ) + ( N', ' ) + ( T5.state ) + (
    N', '
    ) +
    ( Cast(T5.zipcode AS NVARCHAR) ) ) AS NVARCHAR(100))
)
AS
NAMEANDADDRESS
```

TIP

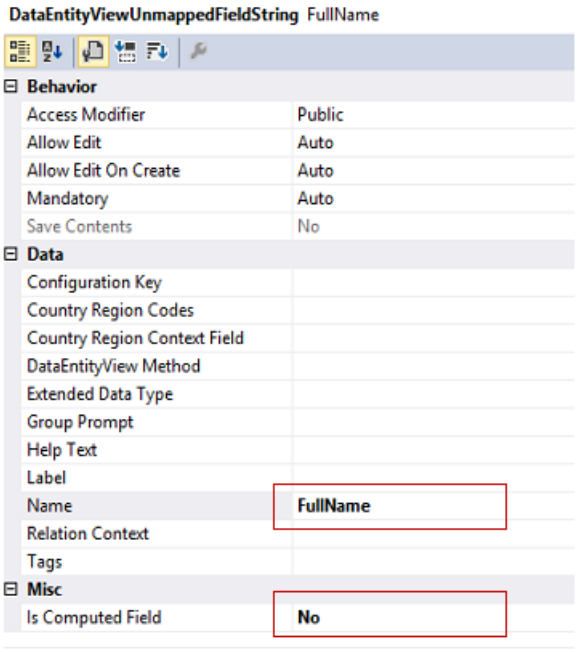
If you receive error in data entity synchronization because of computed columns, it's easier to come up with the SQL definition in Microsoft SQL Server Management Studio (SSMS) before using it in X++.

8. Rebuild the project.
9. Synchronize the database. Don't forget this step. You can do this by going to **Dynamics 365 > Synchronize database > Synchronize**.

Example: Create a virtual field

In this example, you add a virtual field to the **FMCustomerEntity** entity. This field displays the full name as a combination of the last name and first name. X++ code generates the combined value.

1. In the designer for the **FMCustomerEntity** entity, right-click the **Fields** node, and then click **New > String Unmapped Field**.
2. In the properties pane for the unmapped field, set the **Name** property to **FullName**.
3. Set the **Is Computed Field** property to **No**. Notice that you leave the **DataEntityView Method** empty.



4. In the **FMCustomerEntity** designer, right-click the **Methods** node, and then click **Override > postLoad**. Your X++ code in this method will generate the values for the virtual field.

5. Paste the following X++ code in for the **postLoad** override. Notice that the **postLoad** method returns **void**.

```
public void postLoad()
{
    super();
    //Populate virtual field once entity has been loaded from database
    //Format full name - "Doe, John"
    this.FullName = this.LastName + ", " + this.FirstName;
}
```

6. Compile your project.

Example: Use a virtual field to receive and parse an inbound field

Imagine that an external system sends the name of a person as a compound value that combines the last and first names in one field that comes into our system. However, our system stores the last and first names separately. For this scenario, you can use the **FullName** virtual field that you created. In this example, the major addition is an override of the **mapEntityToDataSource** method. When **update** is called, **mapEntityToDataSource** methods are invoked for each data source.

1. In the designer for the **FMCustomerEntity**, right-click the **Methods** node, and then click **Override > mapEntityToDataSource**.

2. Paste the following X++ code in for the **mapEntityToDataSource** method.

```
public void mapEntityToDataSource(DataEntityRuntimeContext entityCtx,
DataEntityDataSourceRuntimeContext dataSourceCtx)
{
    super(entityCtx, dataSourceCtx);
    //Check if desired data source context is available
    if (dataSourceCtx.name() == "FMCustomer")
    {
        FMCustomer dsCustomer = dataSourceCtx.getBuffer();
        //Find position of "," to parse full name format "Doe, John"
        int commaPosition = strfind(this.FullName, ",",0,strlen(this.FullName));
        //Update FirstName and LastName in the data source buffer to update
        dsCustomer.LastName = substr(this.FullName,0,commaPosition-1);
        dsCustomer.FirstName = substr(this.FullName, commaPosition+1, strlen(this.FullName));
    }
}
```

Test the computed and virtual fields

The following **main** method tests your computed and virtual fields. Both fields are tested in a read action, and the virtual field is tested in an update action.

1. For this example, ensure that you have the data set named **Fleet Management (migrated)**. The data set is available from the dashboard in the browser. Click the menu icon in the upper-right corner, click the **APP LINKS** menu, and then scroll to find the data set named **Fleet Management (migrated)**.
2. Paste the following X++ code into the startup object of your project. Run your project.

```
public static void main(Args _args)   // X++
{
    FMCustomerEntity customer;
    //Using transactions to avoid committing updates to database
    ttsbegin;
    //SELECT single customer entity record from the database
    select customer
        where customer.Email == "phil.spencer@adatum.com";
    //Read full name (Virtual Field)
    info(customer.FullName);
    //Read formatted NameAndAddress(computed Field)
    info(customer.NameAndAddress);
    //UPDATE full name (virtual field)
    customer.FullName = "Doe, John";
    customer.update();
    //Reselect data from database to get updated information
    select customer
        where customer.Email == "phil.spencer@adatum.com";
    //Read full name (virtual field)
    info(customer.FullName);
    ttsabort;
}
```

Note on computed column generation failures

If the X++ method that generates the SQL for a computed column throws an exception, DbSync catches the exception, **sets the value of that column to** `NULL`, and logs a *warning*.

Developers are advised to check configuration keys manually in computed column methods to avoid hitting a `NULL` value, if the generation failed.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Cross-company behavior of data entities

2/18/2021 • 7 minutes to read • Edit Online

This topic provides information about how data entities interact with the cross-company concept. To understand this aspect of data entities, you must understand how tables and views apply the cross-company concept. Therefore, this topic begins with a brief review of tables and views, and then explains how data entities are related.

Review of tables and views for cross-company

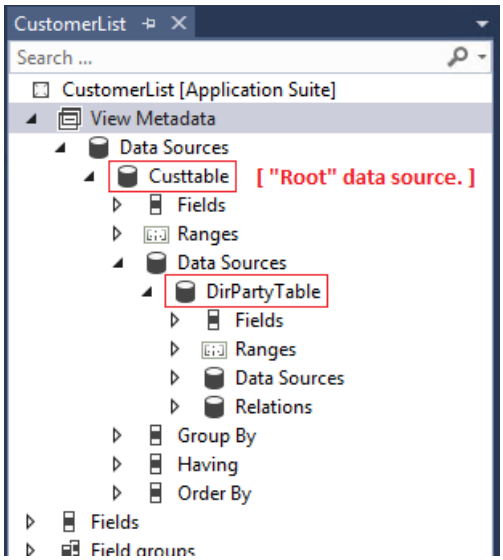
Each table has a **SaveDataPerCompany** property, and each view has a **AllowCrossCompany** property. The following table describes these two properties.

	TABLE	VIEW
Property name	SaveDataPerCompany	AllowCrossCompany
Relevant CRUD mode	CUD	R
Timing of effect	Run time, Design time	Run time, mostly. At design time, this setting causes the view to have **dataAreaId** in its list of selected fields. However, the filter for a specific **dataAreaId** value is added later, at run time.
Meaning of value = Yes	At design time, the system automatically *adds* a **dataAreaId** field to the table, even though the field isn't displayed in the Application Object Tree (AOT). Every record in the table is tagged with the company (or legal entity) that it belongs to. The system automatically *adds* a filter to the SQL **Where** clause to limit the returned set of rows to one **dataAreaId** value.	At run time, the system does *not* automatically add a filter for **dataAreaId** on the **Where** clause of the SQL **Select** statement that it sends to the underlying Microsoft SQL Server system. Therefore, SQL **Select** statements from the view can return a set of records that contains records for *multiple* companies.
Meaning of value = No	The system does *not* add a **dataAreaId** field to the table. The table is said to be a shared table, because none of its records contain any formal company-specific data.	The system automatically *adds* a filter to the SQL **Where** clause to limit the returned set of rows to one **dataAreaId** value. However, the **AllowCrossCompany** property is ignored if the *root* data source of the view is a shared table.

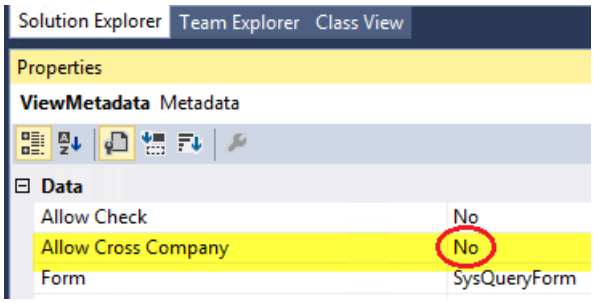
Comparisons within AllowCrossCompany = No

In the following screenshot, the **CustomerList** view has two data sources:

- **Root** – CustTable, which has its **SaveDataPerCompany** property set to **Yes**.
- **Non-root** – DirPartyTable, which has its **SaveDataPerCompany** property set to **No**.



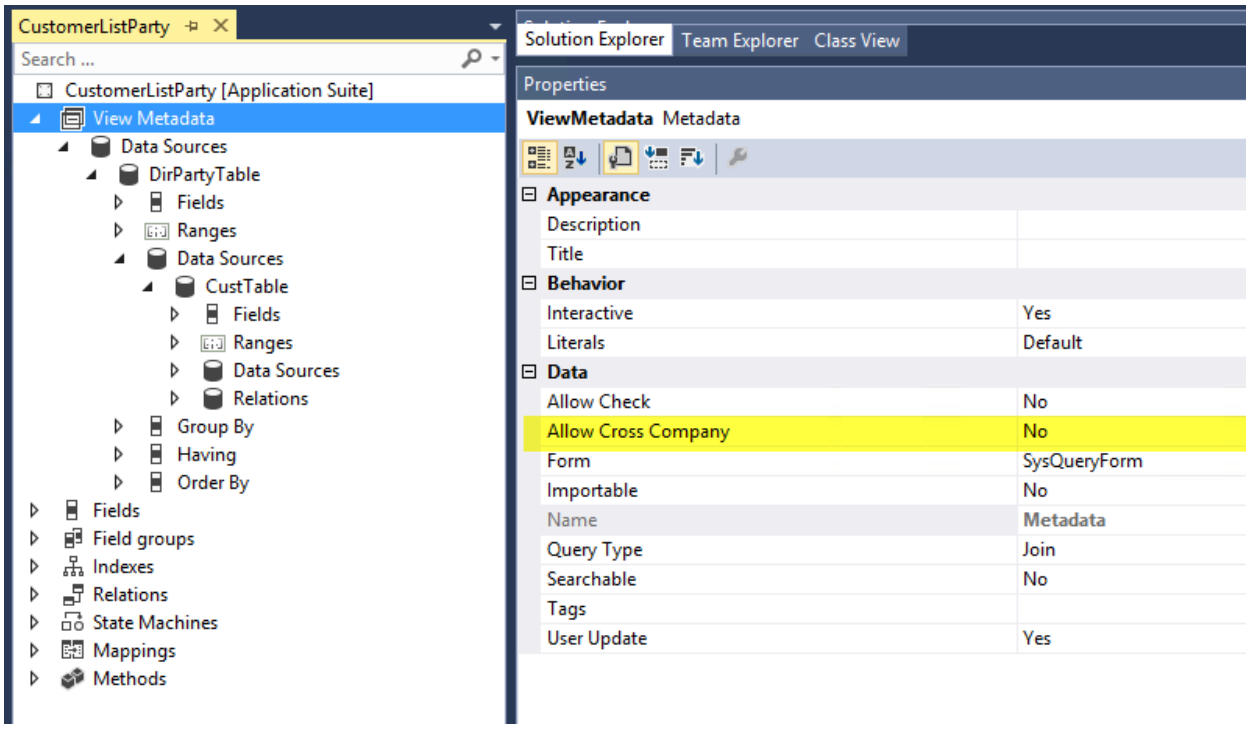
The **CustomerList** view has its **AllowCrossCompany** property set to **No**, as shown in the following screenshot.



Given the preceding information about the **CustomerList** view, the system creates the view in the underlying SQL Server system by generating and then running the following SQL **Create View** statement.

```
CREATE VIEW [dbo].[CUSTOMERLIST]
AS
       SELECT T1.accountnum AS ACCOUNTNUM,
              T1.dataareaid AS DATAAREAID,  -- AllowCrossCompany =No caused this line.
              T1.partition  AS PARTITION,
              T1.recid      AS RECID,
              T2.partition  AS PARTITION#2,
              T2.name       AS NAME
       FROM   custtable T1
              CROSS JOIN dirpartytable T2
       WHERE ( T1.party = T2.recid
              AND ( T1.partition = T2.partition ) )
```

Making DirPartyTable the root data source



By swapping the positions of the two data source tables in the **CustomerList** view, you make the DirPartyTable table the root data source.

```
CREATE VIEW [dbo].[CUSTOMERLISTPARTY]
AS
       SELECT T1.name       AS NAME,
              T1.partition  AS PARTITION,
              T1.recid      AS RECID,
              T2.partition  AS PARTITION#2,
              T2.accountnum AS ACCOUNTNUM
       FROM   dirpartytable T1
              CROSS JOIN custtable T2
       WHERE  ( T2.party = T1.recid
              AND ( T2.partition = T1.partition ) )
go
```

In this case, the SQL **Create View** statement is the same, except for the following two differences:

- The **FROM** clause mentions DirPartyTable first and CustTable second.
- The **SELECT** column list does *not* include the line for **dataAreaId** (because DirPartyTable has its **SaveDataPerCompany** property set to **No**.)

Limitations of tables and views

In some cases, the cross-company control features of tables and views aren't as granular control as you might require. Here are the limitations:

- Company or legal entity fields other than the system **dataAreaId** field can’t be recognized or treated automatically in the that way **dataAreaId** can.
- The cross-company behavior for views is too restricted to the properties of the root data source, even when non-root data sources have a **dataAreaId** field.

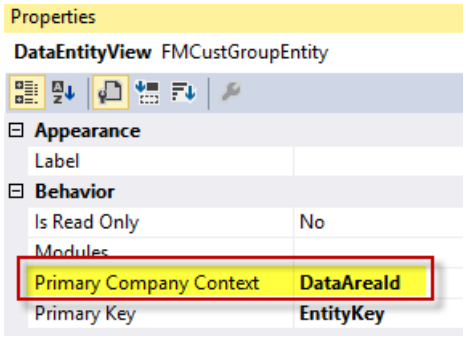
For example, this might happen if the legal entity information is in **LegalEntityRecId**, or if shared tables don't have a **dataAreaId** column.

Design time: Setting the PrimaryCompanyContext property

Data entities help you overcome the limitations of tables and view where cross-company functionality is concerned. Data entities have a **PrimaryCompanyContext** property, where you can specify the entity field to use for company identification. This property provides flexibility and granular control in the following ways:

- The field can be from any data source of the entity and isn't limited to fields of the root data source.
- The field can be any field that is extended from the **DataAreaId** extended data type (EDT), and isn't limited to an underlying system **dataAreaId** field.
- You can use the **PrimaryCompanyContext** property even when the entity has only shared tables as its data sources, if this makes sense for your specific situation.

The following screenshot shows the value set for the **PrimaryCompanyContext** property on the **FMCustGroupEntity** entity.



When the **PrimaryCompanyContext** value is set to a non-empty value, the entity can't behave as a shared entity. The **dataAreaId** field is added to the SQL **Create View** statement.

```
CREATE VIEW [dbo].[FMCUSTGROUPENTITY]
AS
       SELECT T1.custgroup   AS GROUPNAME,
              T1.description AS DESCRIPTION,
              T1.dataareaid  AS DATAAREAID,   -- dataAreaId is added.
              T1.recversion  AS RECVERSION,
              T1.partition   AS PARTITION,
              T1.recid       AS RECID
       FROM   fmcustgroup T1
```

Run time: The behavior of data entities for cross company

In the context of X++ code, the cross-company behavior of data entities resembles the behavior of tables. If the **PrimaryCompanyContext** property for an entity has no value and is empty, the entity behaves like a shared table.

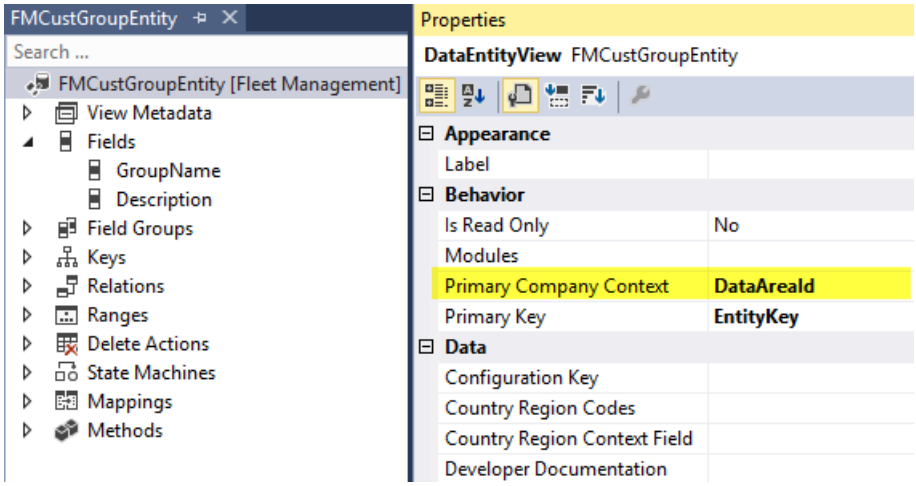
X++ when PrimaryCompanyContext is set

The following table describes the behavior of a data entity under CRUD access when the **PrimaryCompanyContext** property is set to a field value. Both X++ and OData accesses are described.

	X++	ODATA
Read (R)	By default, results are *always* filtered by **dataAreaId** = current company, and cross-company data can be fetched by using the **cross company** option.	Results are *not* filtered by **dataAreaId**. The consumer must filter explicitly.

	X++	OData
Write (CUD)	CUD access to the data entity always occurs in the context of the current company. If cross-company CUD access to the entity is required, use the **changeCompany** keyword.	CUD access to the entity can be accomplished by the consumer for any company by setting the value of the **PrimaryCompanyContext(myDataAreaId)** field. The framework handles the necessary **ChangeCompany** action.

The following X++ code example accesses **FMCustGroupEntity**, which has its **PrimaryCompanyContext** property set to **dataAreaId**.



```
class CrossCompanyXPlusPlusTest   // X++
{
    /// <summary>
    /// Runs the class with the specified arguments.
    /// </summary>
    /// <param name = "_args">The specified arguments.</param>
    public static void main(Args _args)
    {
        FMCustGroupEntity customerGroup;

        // Reads
        // Returns record(s) from current company
        while select GroupName from customerGroup
        {
            info(strfmt("%1-%2",customerGroup.GroupName,customerGroup.DataAreaId));
        }

        // Returns record(s) from crosscompany
        while select crosscompany GroupName from customerGroup
        {
            info(strfmt("%1-%2",customerGroup.GroupName,customerGroup.DataAreaId));
        }



        // Writes
        // create record in current company
        customerGroup.clear();
        customerGroup.GroupName = "CG test";
        customerGroup.insert();

        select firstonly GroupName from customerGroup where customerGroup.GroupName == "CG test";
        info(strfmt("%1-%2",customerGroup.GroupName,customerGroup.DataAreaId));

        // update record in current company
        ttsbegin;

        select firstonly forupdate customerGroup where customerGroup.GroupName == "CG test";
        if (customerGroup)
        {
            customerGroup.Description = "CG test";
            customerGroup.update();
        }
        ttscommit;

        select firstonly GroupName,Description from customerGroup
               where customerGroup.GroupName == "CG test";
        info(strfmt("%1-%2",customerGroup.Description,customerGroup.DataAreaId));


        // Writes
        // create record in CEE company
        changecompany('cee')
        {
            customerGroup = null;
            customerGroup.GroupName = "Cee test";
            customerGroup.insert();

            select firstonly GroupName from customerGroup
                   where customerGroup.GroupName == "Cee test";
            info(strfmt("%1-%2",customerGroup.GroupName,customerGroup.DataAreaId));

            // update record in CEE company
            ttsbegin;

            select firstonly forupdate customerGroup
                   where customerGroup.GroupName == "Cee test";
            if (customerGroup)
            {
                customerGroup.Description = "Cee test";
                customerGroup.update();
            }
            ttscommit;

            select firstonly GroupName,Description from customerGroup
                   where customerGroup.GroupName == "Cee test";
            info(strfmt("%1-%2",customerGroup.Description,customerGroup.DataAreaId));

        }
    }
}
```

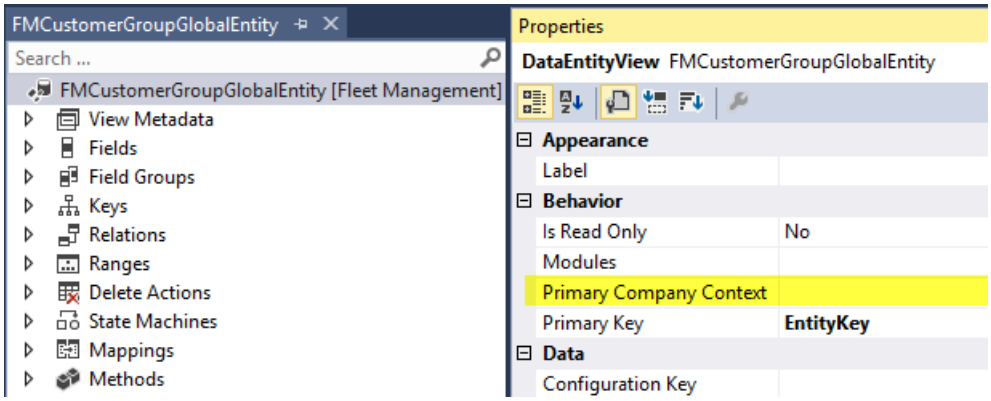
X++ when PrimaryCompanyContext is empty

When the **PrimaryCompanyContext** property is set on the data entity, a **dataAreaId** field is created in the view schema and mapped to the **PrimaryCompanyContext** field. The following table describes the behavior of a data entity under CRUD access when the **PrimaryCompanyContext** property is empty. Both X++ and OData accesses are described.

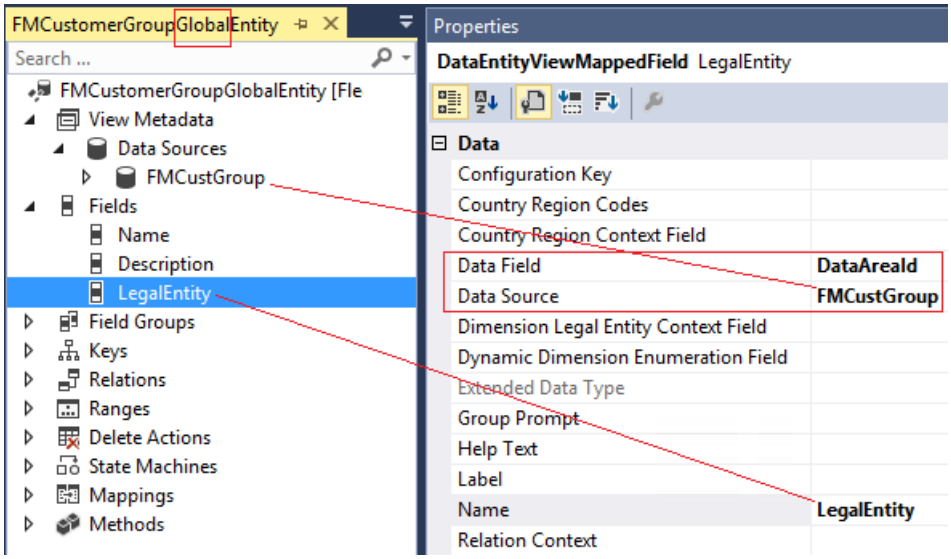
	X++	ODATA

	X++	ODATA
Read (R)	Results aren't filtered, because no system **dataAreaId** field is created on the view schema.	(The same as for R with X++)
Write (CUD)	There is no primary company context to set. Therefore, CUD access to the entity is always in the context of the current company.	(The same as for CUD with X++)

In the current example, the **FMCustomerGroupGlobalEntity** entity has no value assigned to its **PrimaryCompanyContext** property.



However, a **dataAreaId** field from the FMCustGroup table is mapped to the **FMCustomerGroupGlobalEntity** entity as a regular field that is named **LegalEntity**. In this example, the FMCustGroup table is the root data source for **FMCustomerGroupGlobalEntity**. However, we are using this **dataAreaId** field in an informal way that bypasses the automatic mechanisms of the system. All these details are shown in the following screenshot of the **LegalEntity** field.



NOTE

Although the terms *legal entity* and *data entity* both use the word *entity*, don't confuse them. Legal entities and data entities are two entirely different concepts. When the **PrimaryCompanyContext** property is empty, the SQL **Create View** statement usually contains no mention of a system **dataAreaId** column. However, in the current example, **dataAreaId** is "half-mentioned" because of the **LegalEntity** regular field on the data entity. This field is shown in the following SQL statement.

```
CREATE VIEW [dbo].[FMCUSTOMERGROUPGLOBALENTITY]
AS
       SELECT T1.custgroup   AS NAME,
              T1.description AS DESCRIPTION,
              T1.dataareaid  AS LEGALENTITY,   -- dataAreadId is named LegalEntity.
              T1.recversion  AS RECVERSION,
              T1.partition   AS PARTITION,
              T1.recid       AS RECID
       FROM   fmcustgroup T1
```

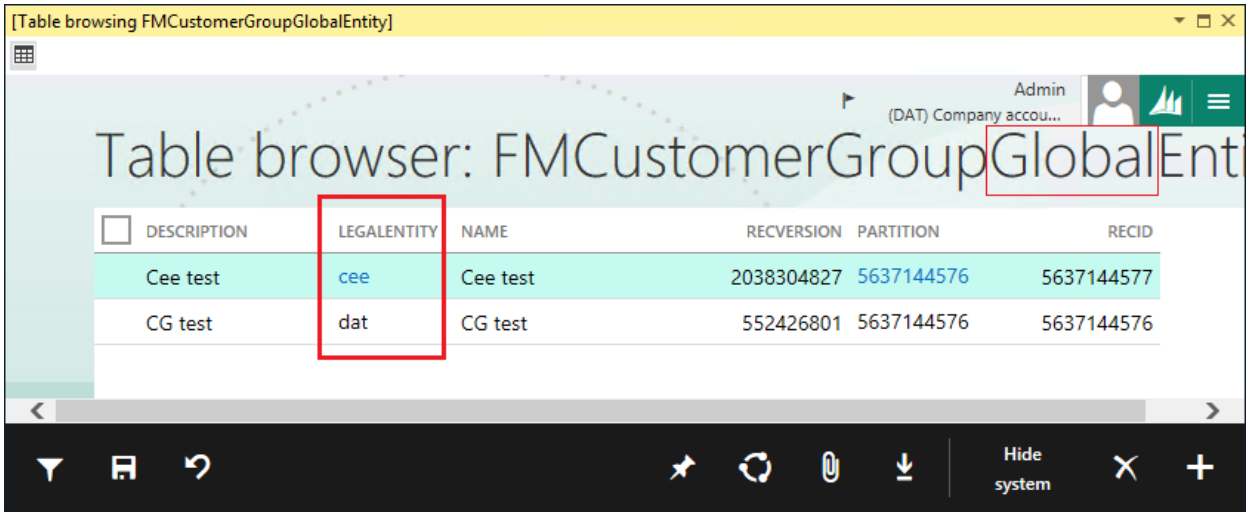
Purpose of this example

This example has two purposes:

- Show shared data by default, even though the backing table might be company-specific.
- Enable the consumer of the data entity to filter on or apply **dataAreaId** if this is required, by using the regular field that is named **LegalEntity**.

Test data

The following screenshot of the **Table browser** page shows the test data that is in the **FMCustomerGroupGlobalEntity** entity before the X++ test code is run.



X++ code

Here's how the X++ test code works with the shared entity:

- It accesses the data entity in shared mode for reads.
- It accesses the data entity with one specific company when a new record is created.

```
class GlobalCrossCompanyXPlusPlusTest
{
    /// <summary>
    /// Runs the class with the specified arguments.
    /// </summary>
    /// <param name = "_args">The specified arguments.</param>
    public static void main(Args _args)
    {
        FMCustomerGroupGlobalEntity customerGroup;

        // Reads
        // Returns record(s) for global entity from all legal entities
        while select Name,LegalEntity from customerGroup
        {
            info(strfmt("%1-%2",customerGroup.Name,customerGroup.LegalEntity));
        }


        // Writes
        // create record in company based on Legal entity
        // In this case, even without ChangeCompany, data source insert would happen
        // in company identified by Legal entity automatically. However, there is a
        // known bug around this area.
        changecompany('ceu')
        {
            customerGroup.clear();
            customerGroup.Name = "CEU test-1";
            customerGroup.LegalEntity = 'CEU';
            customerGroup.insert();
        }

        select firstonly Name,LegalEntity from customerGroup
                where customerGroup.Name == "CEU test-1";
        info(strfmt("%1-%2",customerGroup.Name,customerGroup.LegalEntity));
    }
}
```

Country/region codes and configuration keys

2/18/2021 • 2 minutes to read • Edit Online

This article provides scenarios that are applicable from an implementation perspective for both configuration keys and country/region.

Customer table schema

FIELD NAME	FIELD LABEL	COUNTRY CONTEXT
CustNum	Customer number	
CustName	Customer name	
EinvoiceEANNum	EAN	DK
FiscalCode	Fiscal code	IT

Sample data

CUSTNUM	CUSTNAME	EINVOICEEANNUM{DK}	FISCALCODE{IT}	DATAAREAID
1	Contoso Denmark	AA	{Empty}	DK
2	Contoso Italy	{Empty}	DD	IT

Sample entity

FIELD NAME	COUNTRY CONTEXT
CustomerNumber	
CustomerName	
EAN	DK
FiscalCode	IT

Scenario – Field level only

Isaac, a developer, builds a customer entity that has fields that contain regional settings. The entity is consumed through OData.

For read operations: The consumer of the entity uses this information to complete an effective regional mapping. The consumer ignores the fields that aren't required for that region. For example, consumers in Denmark (DK) are concerned with reading the values of the **EAN** field and core fields only.

For write operations: The consumer of the entity uses this information to identify only the fields that are required to populate data. The consumer expects validation to occur for regional fields and associated core fields.

Behaviors – Fields only

SCENARIO	DESCRIPTION
Design	Entities automatically inherit localization properties from underlying fields.
Design	Developers can't override or set localization properties on entity fields. These properties should be inherited only from tables. Only override on unmapped fields.
Read behavior (OData metadata)	The consumer of an entity from OData will have metadata or annotations to specify which fields are localized.
Read behavior (Data management)	In import/export fields, metadata displays country/region values, so that this information is obvious to the end user.
Read behavior	During cross-company read operations, data from localized fields is displayed only if the context matches. Note that this should already be implemented through the table/view.
Read behavior (Performance)	During company-specific read operations, localized fields are dropped from the query when the context doesn't match.
Write	During write operations to localized fields, hard errors occur if the fields don't match the context.
(Shared table)	If the data source or fields contain a country/region that is a shared (global) table, all operations are ignored, just as if no keys are applied.

Behavior – Data source

The behavior of configuration keys and a country/region that are applied at the data source resembles the behavior of fields. These values are inferred from the data source, just as if they are applied to the field level. Here's an example.

```
Entity E1

    |_ Data Source 1 (DS1)

        Field1

        Field2

    |_Data Source 2 (DS2) UK

        Field3

            Field4
```

Evaluation at the entity E1 level

```
Entity E1

|_F1

|_F2

|_F3 (UK inferred)

|_F4 (UK inferred)
```

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Super types and sub types

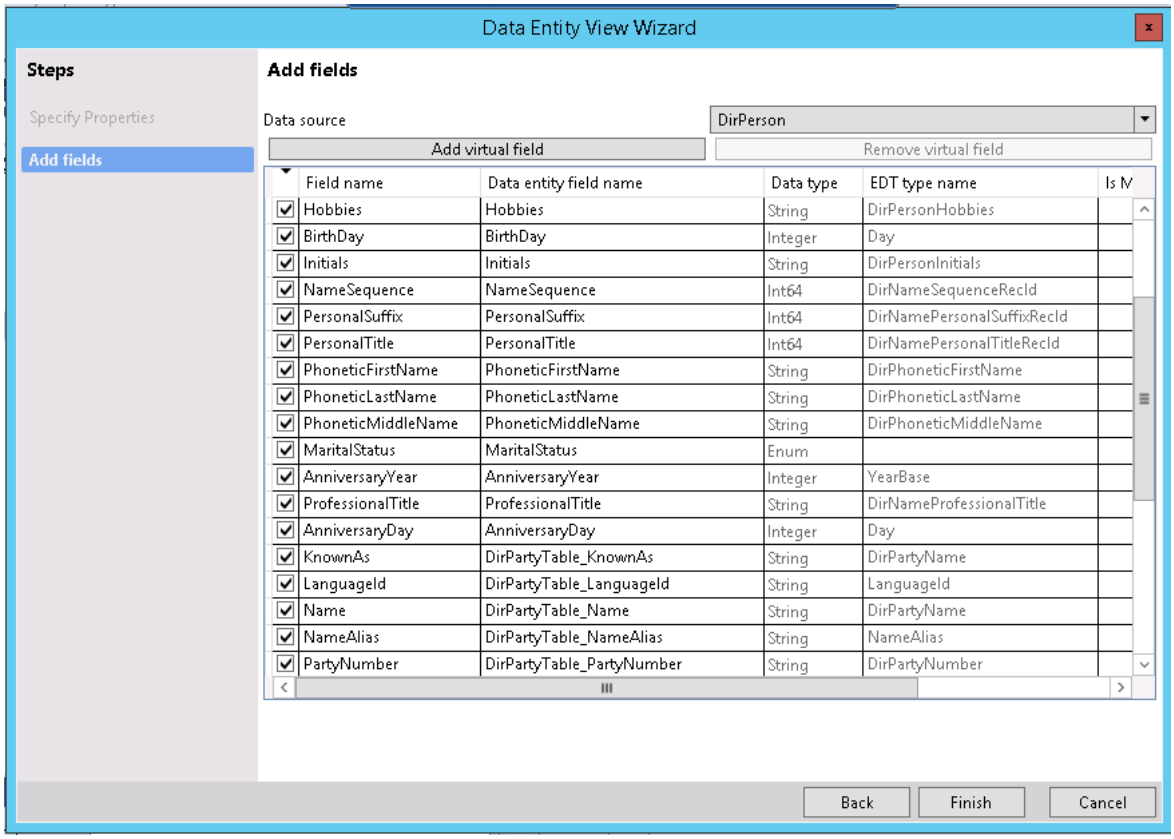
2/18/2021 • 2 minutes to read • Edit Online

Describes support for inheritance patterns in data entities.

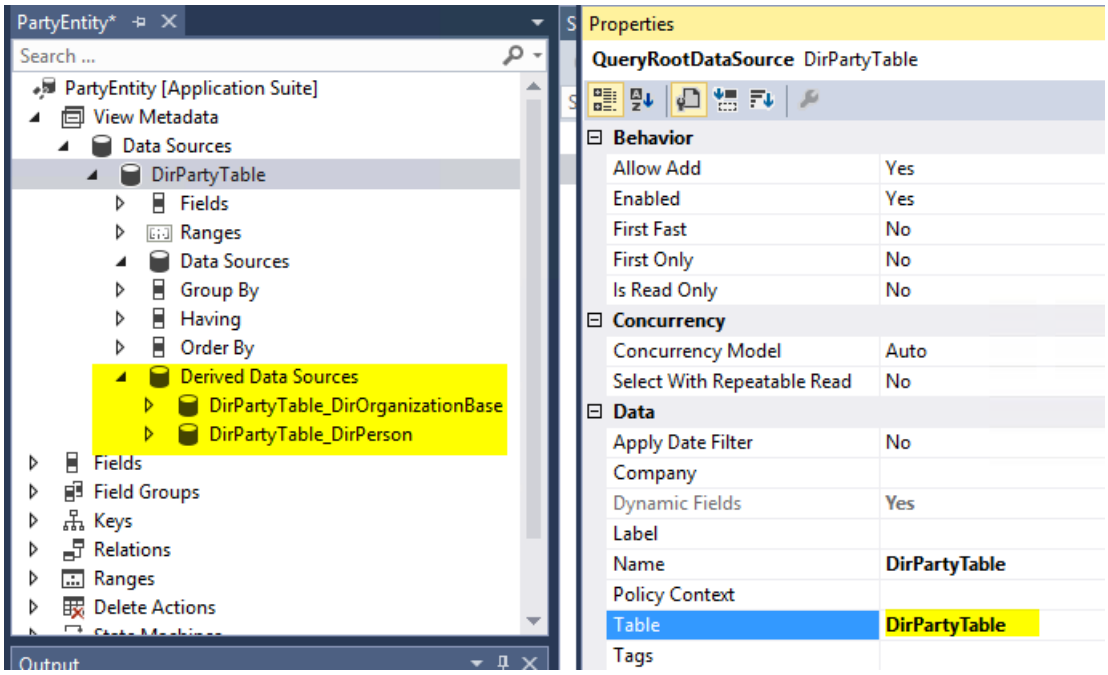
Patterns

There are several ways to create entities for tables that involve inheritance:

- **Leaf/concrete type as data source:** If a concrete type is used as a data source, fields are displayed for both the base type and the current type. For example, in the following screen shots, if DirPerson is the data source, data source fields from both DirPerson and DirPartytable appear.



- **Abstract type/non-leaf as data source:** If a non-leaf type is used as a data source, fields are displayed for both the base type and the current type, but fields from any derived types aren't displayed. Fields from derived types must be added from derived data sources, as shown in the following screen shot.

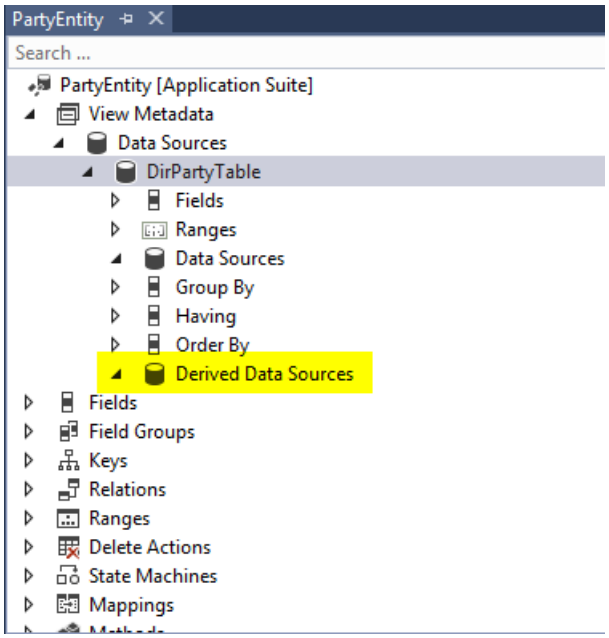


Data Entity View wizard

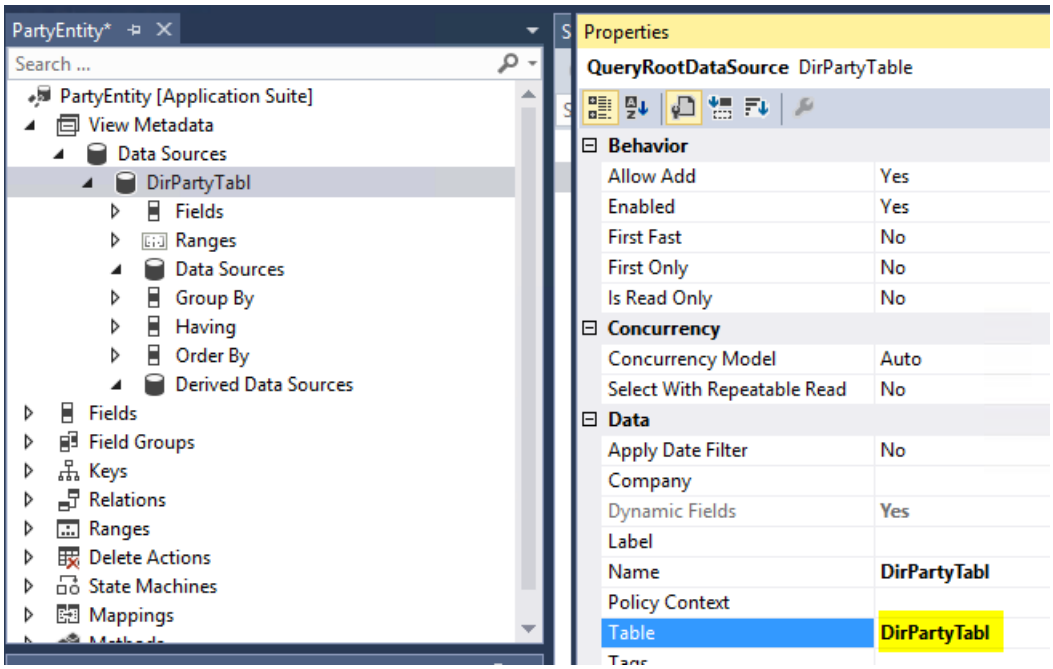
You can use the **Data Entity View** wizard to create data entities where the primary data source (and additional data sources) can be tables that are involved in inheritance.

> **NOTE**
>
> Currently, the wizard doesn't support derived data sources. It shows only fields from the current type or the base type. After you create an entity, you can manually modify it to display derived data sources.

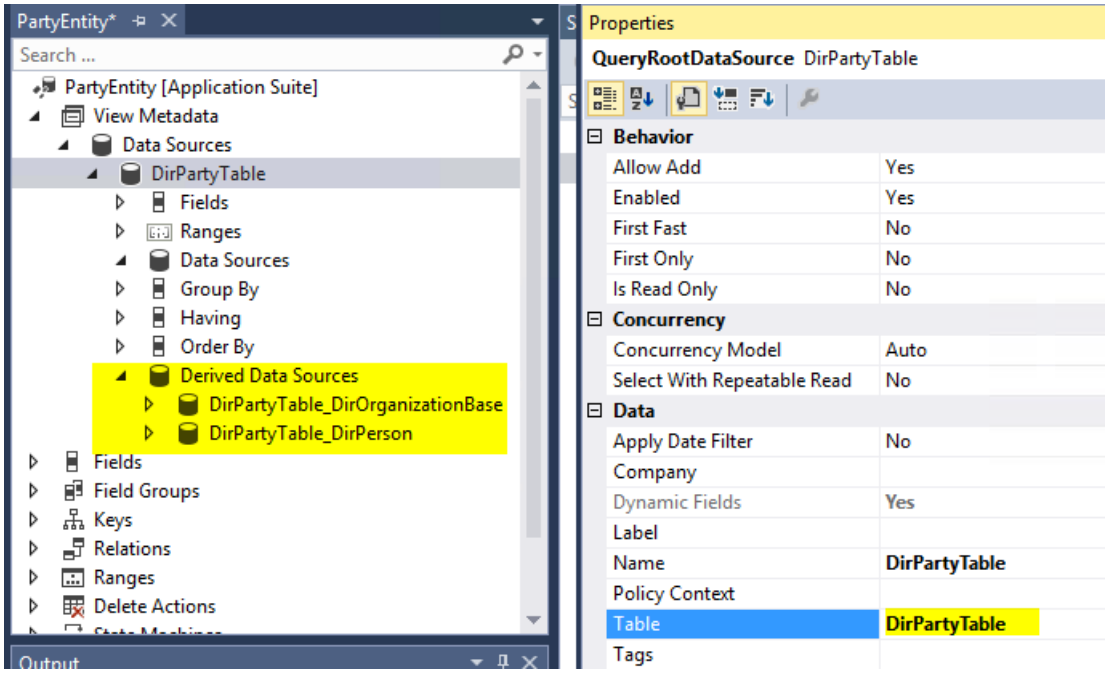
The following screen shots show a data entity that was created by using the wizard, where DirPartyTable is the primary data source.



1. Update the data source table to **DirPartyTabl**.



2. Update the data source table to **DirPartyTable**.

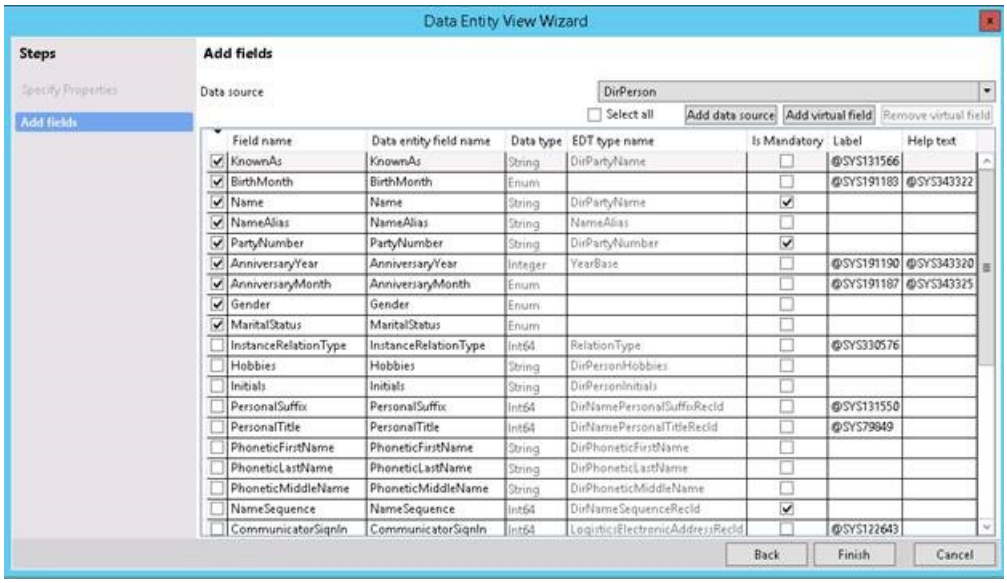


Run time

There is run-time behavior for entities that related to inheritance.

Creating entities for specified types

In this example, we create separate **Person** and **Organization** entities. The primary data source for the **Person** entity is DirPerson, and the primary data source for the **Organization** entity is DirOrganization. This approach, which is reflected in the following screen shots, doesn't require that you write any special run-time code.

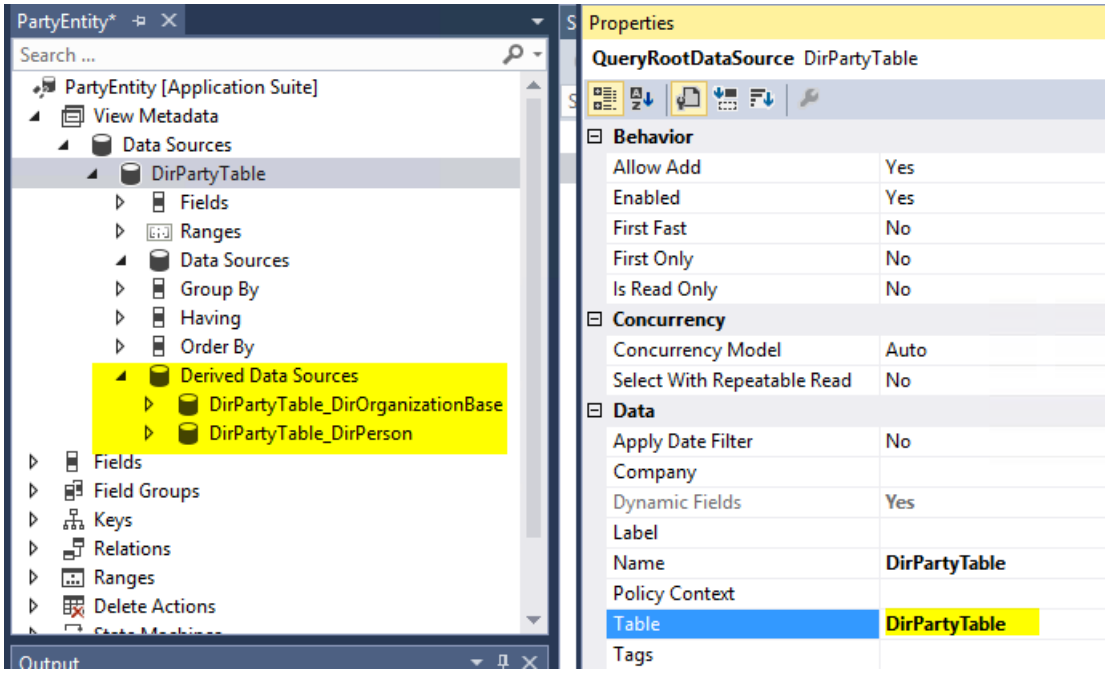

Data Entity View Wizard
Steps
Specify Properties
Add fields
Add fields
Data source
DirPerson
Select all
Add data source
Add virtual field
Remove virtual field
Field name
Data entity field name
Data type
EDT type name
Is Mandatory
Label
Help text
KnownAs KnownAs String DirPartyName @SYS131566
BirthMonth BirthMonth Enum @SYS191183 @SYS343322
Name Name String DirPartyName
NameAlias NameAlias String NameAlias
PartyNumber PartyNumber String DirPartyNumber
AnniversaryYear AnniversaryYear Integer YearBase @SYS191190 @SYS343320
AnniversaryMonth AnniversaryMonth Enum @SYS191187 @SYS343325
Gender Gender Enum
MaritalStatus MaritalStatus Enum
InstanceRelationType InstanceRelationType Int64 RelationType @SYS330576
Hobbies Hobbies String DirPersonHobbies
Initials Initials String DirPersonInitials
PersonalSuffix PersonalSuffix Int64 DirNamePersonalSuffixRecId @SYS131550
PersonalTitle PersonalTitle Int64 DirNamePersonalTitleRecId @SYS79849
PhoneticFirstName PhoneticFirstName String DirPhoneticFirstName
PhoneticLastName PhoneticLastName String DirPhoneticLastName
PhoneticMiddleName PhoneticMiddleName String DirPhoneticMiddleName
NameSequence NameSequence Int64 DirNameSequenceRecId
CommunicatorSignIn CommunicatorSignIn Int64 LogisticsElectronicAddressRecId @SYS122643
Back
Finish
Cancel

Creating entities for generalized types

In this example, we create a single entity, **Party**, that can be used for both **Person** and **Organization**. The primary data source is DirPartyTable, and derived data sources are DirPerson and DirOrganization. The new entity contains the following kinds of fields:

- **Common attributes** – Attributes that aren't specific to **Person** or **Organization**, such as **Name**. These fields are mapped to DirPartyTable.
- **Person-specific attributes** – **Gender**, **Marital Status**, and so on. These fields are mapped to derived data source DirPartyTable_DirPerson.
- **Organization-specific attributes** – **OrgNumber**, **ABC**, and so on. These fields are mapped to derived

data source DirPartyTable_DirOrganization.



Mapping fields from base and multiple derived types in a single data entity is a design-time task. However, at run time, we must specify when each derived type should be created. This can be based on fields such as **InstanceRelationType**, or a computed column can be created to use **String** to represent different types. In the **Party** entity example, a **PartyType** computed column can be created to represent the **Person** and **Organization** derived types. The following code snippet illustrates this approach.

```
    private static server str PartyType()
    {
        str type = ' CASE '+ SysComputedColumn::returnField(tablestr(PartyEntity),
            tablestr(DirPartyTable),fieldstr(DirPartyTable,instancerelationtype))  +
         ' WHEN ' + int2str(tablenum(DirOrganization)) +" THEN 'Organization' "+
        ' WHEN ' + int2str(tablenum(DirPerson)) +" THEN 'Person' "+
        " ELSE 'Party' " +
        ' END ';
        return type;
    }
```

In this example, the **Party** type is computed by using the **InstanceRelationType** column on DirPartyTable. This approach works for reading data. However, to do **Create** or **Update** operations, you must write code where you override the **initializeEntityDataSource** method on the data entity, based on type, and set a correct instance of the derived type for the data source run-time context buffer.

```
    /// <summary>
    ///
    /// </summary>
    /// <param name = "entityCtx"></param>
    /// <param name = "dataSourceCtx"></param>
    public void initializeEntityDataSource(DataEntityRuntimeContext entityCtx,
DataEntityDataSourceRuntimeContext dataSourceCtx)
    {
        DirPerson           person;
        DirOrganization     org;
        DirPartyTable       party;

        if (dataSourceCtx.name() == tablestr(DirPartyTable) &&
            (dataSourceCtx.getDatabaseOperation() == DataEntityDatabaseOperation::Insert ||
             dataSourceCtx.getDatabaseOperation() == DataEntityDatabaseOperation::Update)
           )
        {
if ( dataSourceCtx.getDatabaseOperation() == DataEntityDatabaseOperation::Update)
            {
                party = dataSourceCtx.getBuffer();
                if (this.PartyType == 'Person')
                {
                    person = party as DirPerson;
                    dataSourceCtx.setBuffer(person);
                }
                else
                {
                    org = party as DirOrganization;
                    dataSourceCtx.setBuffer(org);
                }

            }
            else
            {
                if (this.PartyType == 'Person')
                {
dataSourceCtx.setBuffer(new DictTable(tablenum(DirPerson)).makeRecord());
                }
                else
                {
dataSourceCtx.setBuffer(new DictTable(tablenum(DirOrganization)).makeRecord());
                }
            }

        }
        super(entityCtx, dataSourceCtx);
    }
```

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Data entity wizard rules

2/18/2021 • 3 minutes to read • Edit Online

This topic provides information about the natural key expansion of surrogate foreign key fields and the expansion of child/parent relations.

Natural key expansion of surrogate foreign keys

A surrogate foreign key field's extended data type must be **RefRecId** or a derivative. The natural key expansion of a surrogate foreign key field uses the rules in the following list. These rules are listed in the order of evaluation.

1. **Replacement key** – The replacement key fields
2. **Primary key** – The primary index key fields
3. **Alternate key** – The first unique alternate key
4. **Auto-identification key** – The auto-identification fields

Surrogate foreign key fields that are nested in the natural key are recursively expanded. Recurring nested surrogates are limited to the first occurrence. If you select the `is mapped` property of a surrogate foreign key (that is, if you set the property to **true**), the related data source is automatically added to the entity, and the `is mapped` property of each field in the related data source's natural key is selected. In addition, any nested surrogate foreign key data sources are recursively added to the entity. If you clear the `is mapped` property of a surrogate foreign key (that is, if you set the property to **false**), the related data source are automatically removed and unmapped from the entity and any nested surrogate foreign key data sources. The effect of selecting and clearing the **is mapped** property of a surrogate foreign key field differs from the effect of using the **Add data source** and **Remove data source** buttons. If you add a surrogate foreign key data source, the **is mapped** property of the parent data source surrogate foreign key field isn't automatically set to **true**. If you are removing a surrogate foreign key data source, the **is mapped** property of the parent data source surrogate foreign key field isn't automatically set to **false**. By default, the **is mapped** property of the root data source's surrogate foreign key field is set to **true**. Therefore, by default, surrogate foreign key relations are expanded to one level.

Expansion of parent/child relations

Parent/child relations are composition/extension relations that are stored as a relation on the child table. The parent table doesn't detect the relation. A parent/child relation can be either a surrogate foreign key relation or a natural foreign key. Parent/child relations use the following rules:

- The collection of child tables is obtained by querying the cross-reference database for "type reference" relations to the parent table.
- The child table must belong to the same table group as the parent table.
- The relation between the child and parent **relationship type** property must be **association**, **composition**, **link**, or **aggregation**.
- The relation between the child and parent **cardinality** property must be **exactly one** or **zero or one**.
- The relation between the child and parent **related table cardinality** property must be **exactly one** or **zero or one**.

By default, parent/child relations aren't expanded. You must add them by using the **Add data source** button. To change the default behavior for a project in Microsoft Visual Studio, set the **Expand parent/child** option to

true.

Using label text as field names

The following rules enable label text to be used as the field name when the option is selected (**true**):

- All whitespace is removed from the label text.
- The label text is truncated to 77 characters, and then a unique three-digit numeric identifier (000 through 999) is added.
- The label text is passed to the **NamedElement.IsValid** method. If the name is valid, it can be used as the field name. Otherwise, the label text isn't used, and the field name remains unchanged.

The Is mandatory field

The default value of data entity fields is **auto**. This value is used unless it's explicitly changed. For relations, the related field's **Is mandatory** property is set based on the value of the field's **Is mandatory** value.

- If the field's **Is mandatory** property and the related field's **Is mandatory** are both **true**, both fields are explicitly set to **true**.
- If the field's **Is mandatory** property and the related field's **Is mandatory** are both **false**, both fields are explicitly set to **false**.

If the field's **Is mandatory** property and the related field's **Is mandatory** property differ in value, both fields remain unchanged. In this case, the default value of **auto** is used.

Behavioral properties on data entities

2/18/2021 • 3 minutes to read • Edit Online

Every data entity has properties that let you override the same property values on the tables or views that are the data sources of that entity. Your choices affect the behavior of the entity. In the following table, the first column lists the properties that are discussed in this topic. The top row lists the levels where the property is found in the entity designer. The levels are listed in order of increasing granularity: the data source level is more granular than the entity level but less granular than the field level.

	ENTITY LEVEL	DATA SOURCE LEVEL	FIELD LEVEL
ReadOnly	Applies	Applies	.
AllowEdit	.	.	Applies
AllowEditOnCreate	.	.	Applies
Mandatory	.	.	Applies

Entity level

In the designer for your data entity, when you click the name at the root node, the **Properties** pane includes the **Is Read Only** property. The following table describes the behavioral differences between the **Yes** and **No** values of this property.

GROUP	PROPERTY NAME	DISPLAY NAME	VALUES	DEFAULT	DESCRIPTION

GROUP	PROPERTY NAME	DISPLAY NAME	VALUES	DEFAULT	DESCRIPTION
Behavior	IsReadOnly	Is Read Only	No, Yes	No	• **No:** Data modification operations (CUD) *are* allowed, *unless* an individual data source node in the entity's designer is set to **IsReadOnly** = **Yes**. • **Yes:** Only read operations are allowed, regardless of the **IsReadOnly** settings on the individual data source nodes in the entity's designer.

You would set **IsReadOnly** to **Yes** for entities that are consumed mainly for export.

Data source level

If a data entity has three data sources, you might want to allow processes to use the entity to modify the data in one of the data sources but not in the other two. A read-only data source can be used for lookup purposes. You can use the entity designer to achieve this extra degree of granular control. Under the entity's **Metadata** > **Data Sources** node, you can select an entity node and then set the **IsReadOnly** property value for that one data source. The following table describes the interaction between the **IsReadOnly** settings at the data source level and the entity level.

GROUP	PROPERTY NAME	DISPLAY NAME	VALUES	DEFAULT	DESCRIPTION

GROUP	PROPERTY NAME	DISPLAY NAME	VALUES	DEFAULT	DESCRIPTION
Behavior	IsReadOnly	Is Read Only	No, Yes	No	• **No:** Data modification operations (CUD) *are* allowed on the data source, *unless* **IsReadOnly** is set to **Yes** at the entity level. • **Yes:** Only operations are allowed, regardless of the **IsReadOnly** setting on the entity.

Field level

At the field level, the **AllowEdit** and **AllowEditOnCreate** properties are available instead of an **IsReadOnly** property. The two **Allow** properties include **Auto** as a third available value. The **Auto** value inherits the value that is on the field in the underlying table.

> **NOTE**
> The **Auto** value isn't available for unmapped fields, such as computed or virtual fields.

GROUP	PROPERTY NAME	DISPLAY NAME	VALUE	DEFAULT	DESCRIPTION
Behavior	AllowEditOnCreate	Allow edit on create	Auto, No, Yes	Auto	• **Auto:** The property is inherited from the underlying table field. > [!NOTE] The **Auto** value

GROUP	PROPERTY NAME	DISPLAY NAME	VALUE	DEFAULT	DESCRIPTION
					isn't available for unmapped fields, such as computed or virtual fields. - **No:** Users aren't allowed to modify the data for this field in a new record. - **Yes:** Users are allowed to modify the data for this field for a new record. This behavior is enforced for all consumers – X++, OData, and so on. [!IMPORTANT] The **No** and **Yes** values do *not* override the setting on the field in the underlying table.

GROUP	PROPERTY NAME	DISPLAY NAME	VALUE	DEFAULT	DESCRIPTION
Behavior	AllowEdit	Allow edit	Auto, No, Yes	Auto	The behavior is the same as the behavior for **AllowEditOnCreate**, but it applies to updates to *existing* records instead of new records that are being created. This behavior is enforced for all consumers – X++, OData, and so on.
Behavior	Mandatory	Mandatory	Auto, No, Yes	Auto	**Auto:** The property is inherited from the underlying table field. This behavior is enforced for all consumers – X++, OData, and so on. [!IMPORTANT] The **No** and **Yes** values do *not* override the setting on the field in the underlying table.

Validations, default values, and unmapped fields

2/18/2021 • 7 minutes to read • Edit Online

This topic describes how data entity values are validated, how default values can be provided, and how to use fields that are not mapped to data source values, but instead contain virtual or computed data (unmapped fields).

Validations

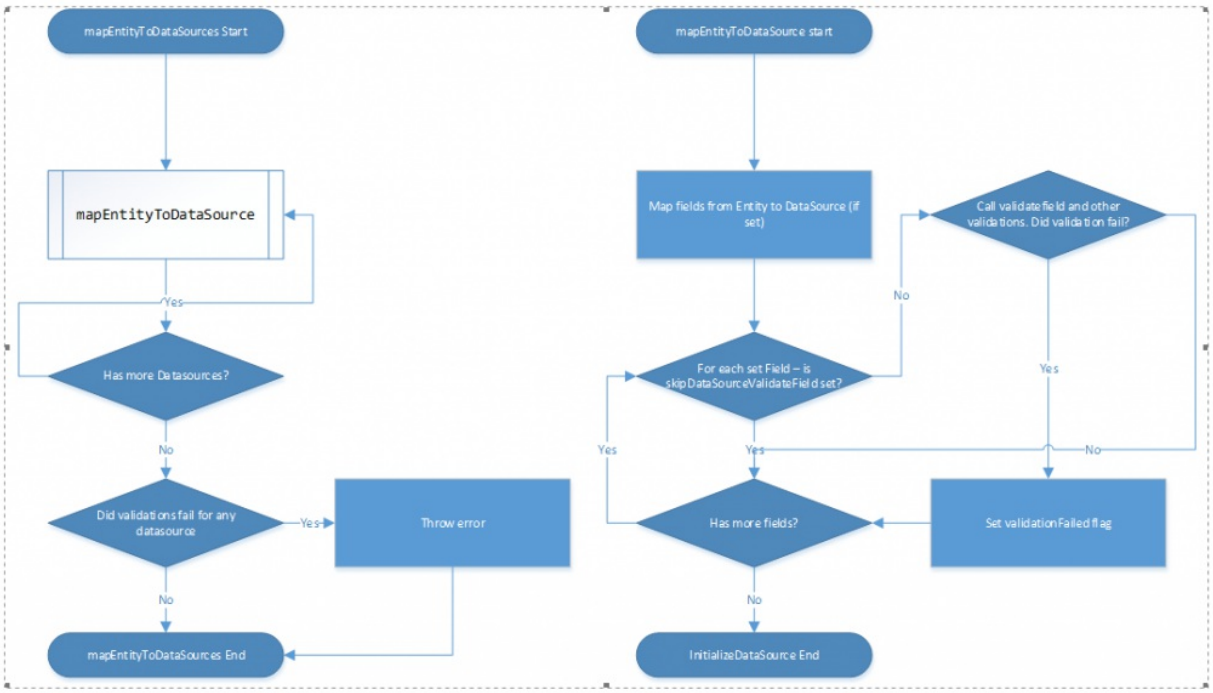
Validations can be defined on the tables that back up entities, at both the field level and the record level. Validations can also be defined at the data entity level.

Table (data source) vs. entity validation

Entities are backed by tables (data sources), and validations are defined for these tables at both the field level (**Table.validateField()**) and the record level (**Table.validateWrite()**). The validations are respected by data entities that are built by using those tables. Although these validations are intrinsic to the tables that back a data entity, validations can also be defined at the data entity level. Like table-based validations, entity-based validations can be written at the field level (**DataEntity.validateField()**) or the record level (**DataEntity.validateWrite()**).

Table-based validation behavior

Table validations are fired automatically as a part of the CUD operations. **Table.ValidateField, AllowEdit, AllowEditOnCreate** Field-level validations are fired automatically when you perform inserts or updates on the data entity. This is true for all paths (X++, OData, and so on). These validations occur during the mapping process, when fields are mapped from an entity to individual data sources.

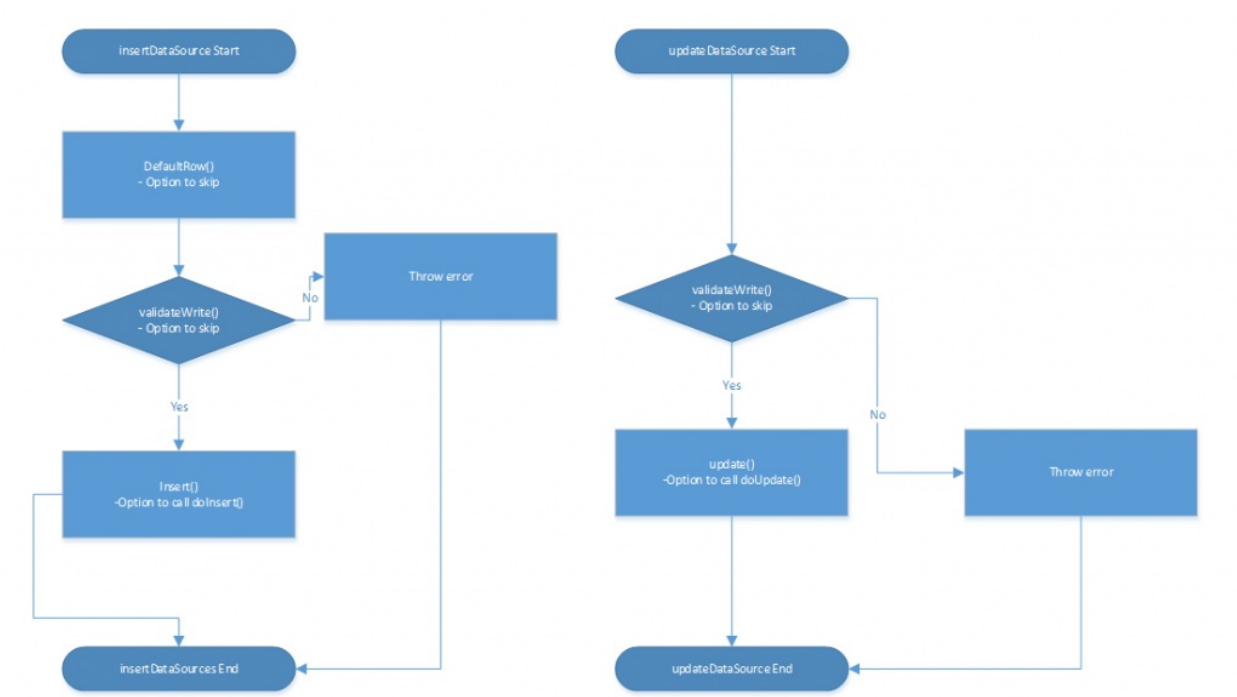


After the field values from the data entity are copied to mapped data source fields, field validations are run on the set fields. Validations include table-level **validateField**, which validates **AllowEdit** and **AllowEditOnCreate**. If a validation fails because of an error, validation for the remaining fields continues. Finally, validation checks whether any error occurred during the validation process for any of the data sources. If there was an error, the process errors out at this point, and table-level **validateWrite()** isn't called. To skip

validateField for a back-end table, a consumer can call **DataEntity.skipDataSourceValidateField(Int _DataEntityFieldId, Boolean _skip)**. Note that the field ID for this method is the field ID of the data-entity mapped field, not the back-end table field. By using the following API, you can skip validation for a particular field, regardless of the consumer.

```
/// <summary>
///
/// </summary>
/// <param name = "entityCtx"></param>
public void persistEntity(DataEntityRuntimeContext entityCtx)
{
    this.skipDataSourceValidateField(<FieldId>,true)
    super(entityCtx);
}
```

Table.ValidateWrite Record-level **ValidateWrite** validations that are defined in back-end tables are fired automatically when you perform data-entity inserts and updates. This is true for all paths (X++, OData, and so on). These validations occur just before the actual insert or update is applied to the data source. If the validation fails, an error is thrown, and the process stops for other data sources.



To skip **validateWrite** for all back-end tables for a data entity, a consumer can call **DataEntity.skipDataSourceValidateWrite(Boolean _skip)**. This method turns **validateWrite** on or off for all data sources. By using the following API, you can skip validation for a particular field, regardless of the consumer.

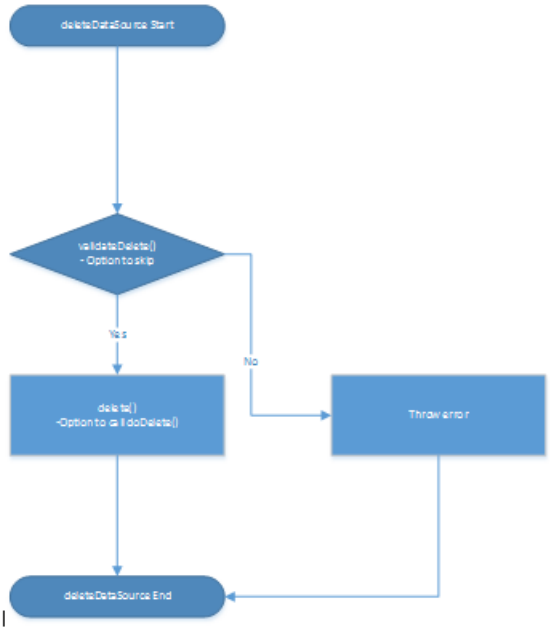
```
    /// <summary>
    ///
    /// </summary>
    /// <param name = "entityCtx"></param>
    /// <param name = "dataSourceCtx"></param>
    /// <returns></returns>
    public boolean saveEntityDataSource(DataEntityRuntimeContext entityCtx,
DataEntityDataSourceRuntimeContext dataSourceCtx)
    {
        boolean ret;

        if (entityCtx.getDatabaseOperation() == DataEntityDatabaseOperation::Insert && dataSourceCtx.name()
== tablestr(DirPartyTable))
        {

        dataSourceCtx.skipValidateWrite(true);
}
        ret = super(entityCtx, dataSourceCtx);

        return ret;
    }
```

Table.ValidateDelete Record-level **ValidateDelete** validations that are defined in back-end tables are fired automatically when you perform data entity deletes. This is true for all paths (X++, OData, and so on). These validations occur just before the delete is applied to the data source. If the validation fails, an error is thrown, and the process stops for other data sources.



To skip **validateDelete** for all back-end tables for a data entity, a consumer can call **DataEntity.skipDataSourceValidateDelete(Boolean _skip)**. This method turns **validateDelete** on or off for all data sources. By using the following API, you can skip validation for a particular data source, regardless of the consumer.

```
    /// <summary>
    ///
    /// </summary>
    /// <param name = "entityCtx"></param>
    /// <param name = "dataSourceCtx"></param>
    /// <returns></returns>
public boolean saveEntityDataSource(DataEntityRuntimeContext entityCtx,
DataEntityDataSourceRuntimeContext dataSourceCtx)
       {
        boolean ret;

        if (entityCtx.getDatabaseOperation() == DataEntityDatabaseOperation::Delete && dataSourceCtx.name()
== tablestr(DirPartyTable))
        {
            dataSourceCtx.skipValidateDelete(true);
        }
        ret = super(entityCtx, dataSourceCtx);

        return ret;
       }
```

Entity-based validation behavior

VALIDATION	TARGET	CALLER
DataEntity.ValidateField	• Data types • Mandatory relationships (both tables and extended data types [EDTs]) • Any custom validation • Doesn't call **validateField** for underlying mapped table fields	• Is called automatically from OData • Is called by the form engine when a field is modified • Isn't called automatically if an insert/update is fired from X++ code
DataEntity.ValidateWrite	• Mandatory columns • Relationships (both tables and EDTs) • Any custom validation • Doesn't call table-level **validateWrite** for underlying tables	• Is called automatically from OData • Is called by the form engine when a record is saved. • Isn't called automatically if an insert/update is fired from X++ code
DataEntity.ValidateDelete	• DeleteActions • Any custom validation • Doesn't call table-level **validateDelete** for underlying tables	• Is called automatically from OData. • Is called by the form engine when a record is deleted • Isn't called automatically if a delete is fired from X++ code

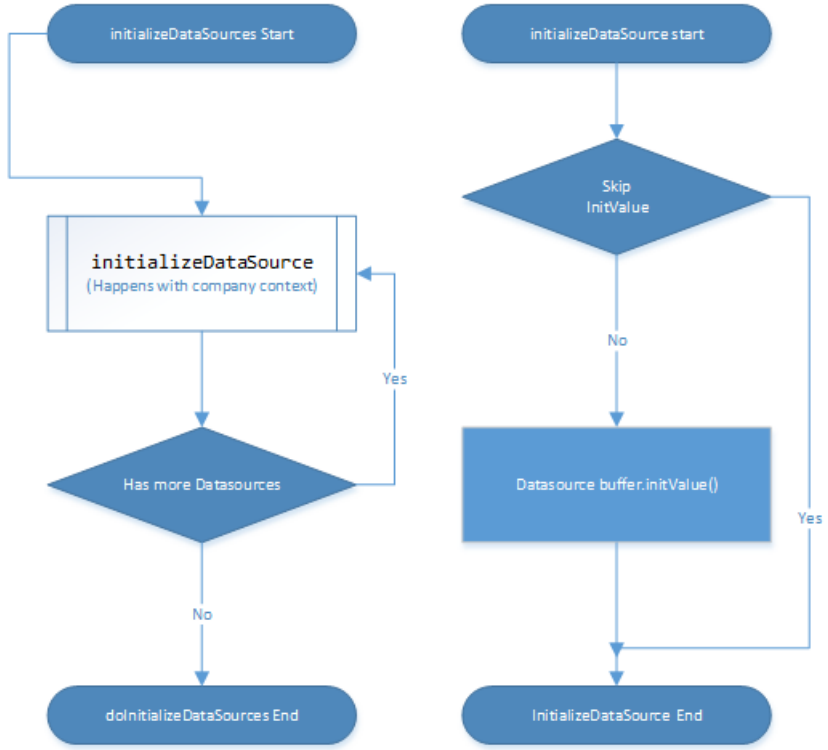
Defaults

Default values can be provided for initializations and rows.

Initializations

DataEntity.initValue: A data entity is initialized with default values and by using any custom logic that is present in entity-level **initValue**. This method isn't called automatically when an insert or update is performed on a data entity from X++. It must be called explicitly if it's required. The method is called automatically by the form engine when a new record is created. **DataEntity.initValue** doesn't call the **initValue** method for back-end tables that are used in the data entity. **Table.initValue:** Table-level **initValue**, as defined for back-end tables, is fired when you perform a data entity insert. This is true for all paths (X++, OData, and so on). **Table.initValue**

is run just before the entity is mapped to data source fields.



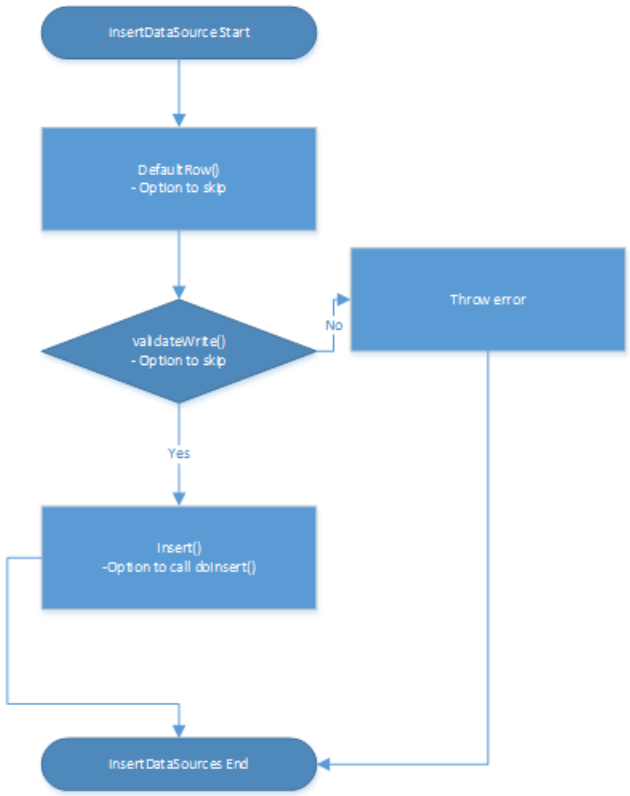
To skip entity-level **initValue** for all back-end tables for a data entity, a consumer can call **DataEntity.skipDataSourceInitValue(Boolean _skip)**. This method turns **initValue** on or off for all data sources. By using the following API, you can skip **initValue** for a particular field, regardless of the consumer.

```
    /// <summary>
    ///
    /// </summary>
    /// <param name = "uvCtx"></param>
    /// <param name = "uvdsCtx"></param>
public void initializeEntityDataSource(DataEntityRuntimeContext uvCtx, DataEntityDataSourceRuntimeContext
uvdsCtx)
    {

        if (uvdsCtx.name() == tablestr(DirPartyTable)
&& uvdsCtx.getDatabaseOperation() == DataEntityDatabaseOperation::Insert)
        {
            uvdsCtx.skipInitValue(true);
        }
        super(uvCtx, uvdsCtx);
    }
```

DefaultRow

DataEntity.DefaultRow: DataEntity.DefaultRow is used in conjunction with **defaultField** and **getDefaultingDependencies** to provide defaults. It isn't called automatically by X++ or the form engine. **Table.DefaultRow: Table.DefaultRow** is called automatically for each data source after mapping is completed, and before the insert and validation on the data source.



Unmapped fields

A data entity can have *unmapped* fields in addition those fields that are directly mapped to fields of the data sources. There are two mechanisms for generating values for unmapped fields:

- Custom X++ code
- SQL that is run by Microsoft SQL Server

The two types of unmapped fields are *virtual* and *computed*. Unmapped fields always support read actions, but the feature specification might not require any development effort to support write actions.

Virtual field

- A non-persisted field.
- Controlled by custom X++ code.
- Read and writes occur through custom X++ code.
- Typically used for intake values that are calculated by using X++ code and can't be replaced by computed columns.

Computed field

- The value is generated by an SQL view computed column.
- During reads, data is computed by SQL and fetched directly from the view.
- For writes, custom X++ code must parse the input value and then write the parsed values to the regular fields of the data entity. The values are stored in the regular fields of the data sources of the entity.
- Used mostly for reads.
- It's a good idea to use computed columns instead of virtual fields whenever you can, because computed columns are computed at the SQL Server level, whereas virtual fields are computed row by row in X++.

Properties of unmapped fields

CATEGORY	NAME	TYPE	DEFAULT VALUE	BEHAVIOR
Data	IsComputedField	NoYes	Yes	• **Yes:** The field is synchronized as a SQL view computed column. An X++ method is required to compute the SQL definition string for the column. The virtual column definition is static and is used when the entity is synchronized. After that, the X++ method isn't called at run time. • **No:** The field is a true virtual field, where inbound and outbound values are fully controlled through custom code.
Data	ComputedFieldMethod	String		A static **DataEntity** method in X++ is used to build the SQL expression that generates the field definition. This property is disabled and irrelevant if the **IsComputedField** property is set to **No**. The method is required if the **IsComputedField** property is set to **Yes**.
Data	ExtendedDataType	String		

Unmapped field comparison

	VIRTUAL FIELD	COMPUTED FIELD

	VIRTUAL FIELD	COMPUTED FIELD
Metadata properties	Is computed = No	• Is Computed = Yes • Computed Field Method = static method
Read	• X++ (override **postLoad**) • Row by row	• SQL computed column • Set-based read possible
Write	X++ (override **mapEntityToDataSource**)	X++ (override **mapEntityToDataSource**)
Advantages	• Unbound to the schema, keeps the public contract the same, but the implementation can change • Call X++ methods	Faster reads, large export can occur directly from the view

Examples

The following table provides a computed example if a **UnitOfMeasure** relationship exists, and displays that in an unmapped field.

VIRTUAL FIELD	COMPUTED FIELD
On postLoad()*//Check to see if record exists in UnitOfMeasureInternalCode.UnitOfMeasure//Set hasFixedInternalCode value based on the field*if(this.UnitOfMeasure)this.HasFixedInternalCodeVirtual = NoYes::Yes; else this.HasFixedInternalCodeVirtual = NoYes::No;	On computedFieldMethod()*//Desired SQL computed column statement(CASE WHEN T2.RECID IS NULL THEN 0 ELSE 1 END) AS INT)*

Security and data entities

2/18/2021 • 7 minutes to read • Edit Online

> **NOTE**
> Data entities do not support the Extensible Data Security (XDS) concepts.

Entry points

Data entities support entry point security. This support resembles the support that menu items and pages have. To give you flexibility when you define a security model, data entities allow for a separate security configuration for each integration mode. Currently, two entry points/integration modes are identified for a data entity.

ENTRY POINT	DESCRIPTION
Data services	The ability to use OData services (API) for the entity.
Data management	The ability to use asynchronous integration options for the entity, such as import/export and connector integration.

Target scenarios

Data entities can support multiple categories of scenarios. Each category might have to be secured separately.

- **Data management (file-based import/export, and so on)** – Typically, a data manager performs these scenarios. These scenarios might provide access to data that isn't usually accessible through the UI for the client. Therefore, you will often want to secure data management scenarios independently of access to the related page, so that a data manager can perform only import/export operations.
- **General integration via OData** – Many integration scenarios require that data entities be exposed as services, so that data can be accessed via OData (for example, from an online storefront or a Process Lifetime Management [PLM] system). Often, you will want to control access to data entities that are built for this purpose independently of page access. In other words, you will want to grant access to the service interface without granting access through the client UI.
- **Microsoft Office integration (Edit in Excel, and so on)** – Office integration scenarios also require that data entities be accessed via OData. However, from an end-user perspective, these scenarios can be viewed as a natural extension of the client, where, for example, Microsoft Excel is used to simplify some editing tasks. Therefore, there is usually no reason to secure the Microsoft Office integration independently of page access.

Privilege/duty mapping

Depending on the target scenarios for a data entity, you should create one or more new privileges, and extend existing duties. Alternatively, you can map the new privileges to duties that are created specifically for the target scenario. This approach helps guarantee that no user is granted more access than the user requires for the scenario.

The following table shows the privileges that you should create. It also explains how you should map these privileges to duties. If your data entity is intended to support more than one scenario, you should create the combined set of privileges and duty mappings.

TARGET SCENARIO	PRIVILEGES	DUTY MAPPING

The data entity is intended for data management.	Create the following new privileges:
	• *<DataEntity>***Import**
	  ○ Grant=Create
	  ○ IntegrationMode=Data Management
	• *<DataEntity>***Export**
	  ○ Grant=Read
	  ○ IntegrationMode=Data Management
		Extend the relevant data management duties with the new privileges. For more information, see the "Data administrator role" section later in this topic.

The data entity is intended for general integration via OData.	Create the following new privileges:
	• *<DataEntity>***View**
	  ○ Grant=Read
	  ○ IntegrationMode=Data Services
	• *<DataEntity>***Maintain**
	  ○ Grant=Delete
	  ○ IntegrationMode=Data Services
		Create new duties for the integration scenario, and map the relevant new privileges to these duties. For more information, see the "Duty naming guidelines" section later in this topic.

The data entity is intended for Microsoft Office integration.	Create the following new privileges:
	• *<DataEntity>***View**
	  ○ Grant=Read
	  ○ IntegrationMode=Data Services
	• *<DataEntity>***Maintain**
	  ○ Grant=Delete
	  ○ IntegrationMode=Data Services
		Extend the relevant existing duties that provide access to the related pages with the new privileges.

Because the approach that is described in the preceding table complies with the principle of least privilege, we recommend that you use it. Nevertheless, in some situations, you can use the following simpler approach. However, be aware that this approach might be less secure. It might also be slightly harder to maintain and extend.

TARGET SCENARIO	PRIVILEGES	DUTY MAPPING	POTENTIAL ISSUES

The data entity is intended for data management, general integration via OData, and Microsoft Office integration.	Create the following new privileges:
	• *<DataEntity>***View**
	  ○ Grant=Read
	  ○ IntegrationMode=All
	• *<DataEntity>***Maintain**
	  ○ Grant=Delete
	  ○ IntegrationMode=All
		1. Extend the relevant data management duties with the new privileges.
		2. Create new duties for the integration scenario, and map the relevant new privileges to these duties.
		3. Extend the relevant existing duties that provide access to the related page with the new privileges.
			When you use this approach, a data manager who is granted access to import/export data can also access the system from a web service. Likewise, a user who is granted access to the page that is associated with a data entity can also access the system from a web service. This user will be prevented from data import/export only if the user hasn't been granted the related data management duty.

Duty naming guidelines

When you create data entities for specific integration scenarios, you should also create separate duties. These duties grant the external application or service the required access to the data entities. The duties that you create should follow the same naming guidelines as the corresponding duties that provide access through the client UI. However, you should add a "using services" suffix.

DUTY TYPE	DUTY OBJECT NAME SUFFIX	DUTY NAME TEMPLATE
Enable	...IntegrationEnable	Enable ... using services
Record	...IntegrationMaintain	Maintain ... using services
Authorize	...IntegrationApprove (Release, Confirm, Journalize)	Approve (Release, Confirm, Journalize) ... using services
Inquire	...IntegrationInquire	Inquire into ... using services

Here are some examples of duty names that follow these guidelines:

- Maintain route master using service
- Inquire into case progress using services

Data administrator role

The **DataManagementApplicationAdministrator** security role enables an associated user to have full import/export capabilities in the **Data management** workspace. This security role has two security duties for each of the five data entity categories that are assigned to it. One duty is for importing data via data entities of the associated category, and one for exporting data via data entities of the associated category. Therefore, a total of 10 security duties are assigned to this security role:

- DataManagementApplicationDocumentEntitiesMaintain
- DataManagementApplicationDocumentEntitiesView
- DataManagementApplicationMasterEntitiesMaintain
- DataManagementApplicationMasterEntitiesView
- DataManagementApplicationParametersEntitiesMaintain
- DataManagementApplicationParametersEntitiesView
- DataManagementApplicationReferenceEntitiesMaintain
- DataManagementApplicationReferenceEntitiesView
- DataManagementApplicationTransactionEntitiesMaintain
- DataManagementApplicationTransactionEntitiesView

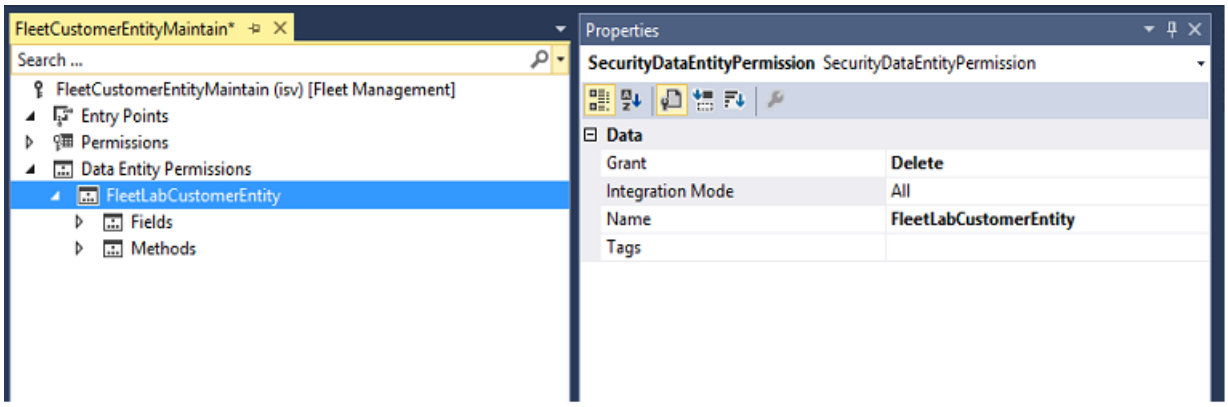
When you create data entities that can be used in the **Data management** workspace, you must extend these duties with the new security privileges, based on the **Entity Category** property that is specified on the data entity. (For information about how to extend duties with the new security privileges, see the "Privilege/duty mapping" section earlier in this topic.) You can also use the duties to create new roles for specific data management scenarios.

Modeling new entry point security in the Application Explorer

The pattern for modeling security resembles the pattern for modeling security with privileges on an entry point. To model security, follow these steps.

1. Create a new privilege.

2. Create new data entity permissions.
3. Set the **Name** to **Data Entity**.
4. Select the **Access Level**.
5. Select **Integration Mode** (All > Data Services > Data Management). This is specific to Object Type: Data Entity.
    - **All** – Applies same security settings to be applied to both OData and data import/export.
    - **Data Management** – Applies only to data import/export and connector integration.
    - **Data Services** – Only applies to OData Services.



Sensitive data

The Table Protection Framework (TPF) enables strict access control to data that is stored in Finance and Operations. This feature is exposed through the AOSAuthorization property on tables and table fields. If you mark a table or field by using AOSAuthorization, the security framework now requires that a user be granted explicit access to that resource. This requirement also applies when the table or field is accessed through data entities. This section describes the guidelines for granting TPF permissions for data entities.

Data management

For data entities that are targeted at data migration, you should assign TPF permissions to the corresponding import/export privileges. In this way, you help guarantee that all data can be imported and exported.

Integration by using OData

For data entities that are targeted at integration scenarios, the TPF permissions that you should assign depend on whether the TPF-protected field is essential for the data entity as a whole to work:

- **If the TPF-protected field is essential**: An essential field is a field that will always be read/written. In this case, TPF permissions should be granted to the same privileges that grant access to the data entity.
- **If the TPF-protected field isn't essential**: Examples of nonessential fields include the field for a worker's Social Security number and the field for a vendor's bank account number. In this case, TPF permissions for accessing the field should be granted through a separate privilege, and that privilege should be assigned directly to the roles that require access to the TPF-protected field. However, if the field is a mapped field on the entity, that access has probably already been granted to the role, if that role also has access to the field through pages in the client UI.

There are several advantages to granting explicit access to TPF-protected fields that aren't considered essential for the entity:

- You can more easily discover who has access to sensitive data.
- You help reduce the risk that someone will gain access to sensitive data by accident, because a role gains

access only if you assign both a duty and a privilege to it.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Test services by using third-party utilities

2/18/2021 • 7 minutes to read • Edit Online

At https://github.com/Microsoft/Dynamics-AX-Integration, Microsoft provides sample code for consuming services. However, there are many scenarios where the other endpoint in an integration might not use a Microsoft stack. Even when the other endpoint does use, for example, the Open Data Protocol (OData) client code that Microsoft makes available, you might find it useful to perform the following actions:

- Explore and analyze how an interaction's messages are constructed.
- Test the response of a service to a well-known request.
- Determine how exceptions will appear to the other endpoint.

Many frequently used tools that will help you perform these actions are available. This topic isn't an endorsement of any tool. Although it provides examples that use some frequently used software utilities, the principles should broadly apply to other, similar tools.

Prerequisites

Before you can test a service by using an external application, you must register the application in Microsoft Azure, and in Finance and Operations.

For details, see:

- Register an application with AAD
- Register your external application

Query OData by using Postman

Postman (https://www.getpostman.com/postman) is a tool that is often used to interact with RESTful services (such as OData) in scenarios that involve the development and testing of application programming interfaces (APIs). This procedure isn't an endorsement of Postman, and other similar tools are available. However, we are using Postman to illustrate the concepts and messages that are involved when you use OAuth to authenticate with Azure AD, and then make OData requests to and receive responses from the application.

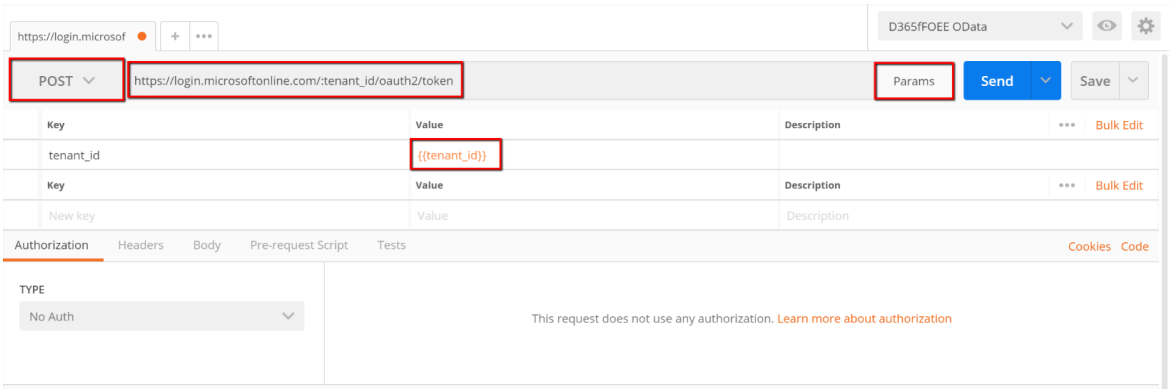
1. Start Postman.
2. In the upper-right corner, select the gear button, and then select **Manage environments** to create or update an environment.
3. Enter a name for the environment, and then select **Bulk Edit**.
4. Enter key-value pairs as shown in the following table. Enter one pair per line, and separate the key and value by using a colon (:).

KEY	VALUE
tenant_id	The Azure tenant ID that you looked up during the setup of prerequisites
client_id	The Azure AD application ID that you registered during the setup of prerequisites

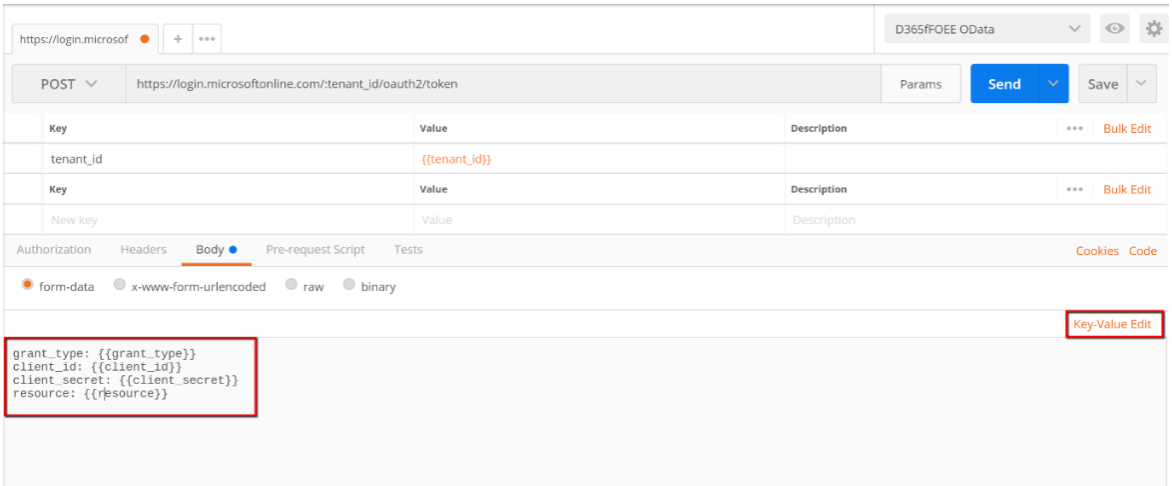
KEY	VALUE
client_secret	The secret key that you generated during application registration during the setup of prerequisites
grant_type	client_credentials
resource	The base URL of the instance without the trailing '/'

5. To verify that the key-value pairs can be parsed correctly, select **Key-Value Edit**, and review the results.
6. Close the environment page.
7. In the field to the left of the gear and eye buttons, select the new or updated environment.
8. To retrieve an Azure AD token, create a POST request that has a URL in the format `https://login.microsoftonline.com/[tenant ID]/oauth2/token`.

   You can use a URL parameter that refers to the **tenant_id** environment variable, such as `https://login.microsoftonline.com/:tenant_id/oauth2/token`.



9. On the **Body** tab, add body elements as request parameters that refer to the environment variables that you created earlier. Select **Bulk Edit**, enter the keys from the previous table, enter a colon (:), and then enter the key name again but enclose it in double braces ({{}}). Enter one request parameter per line. For example, enter **grant_type:{{grant_type}}**. Here is an example.

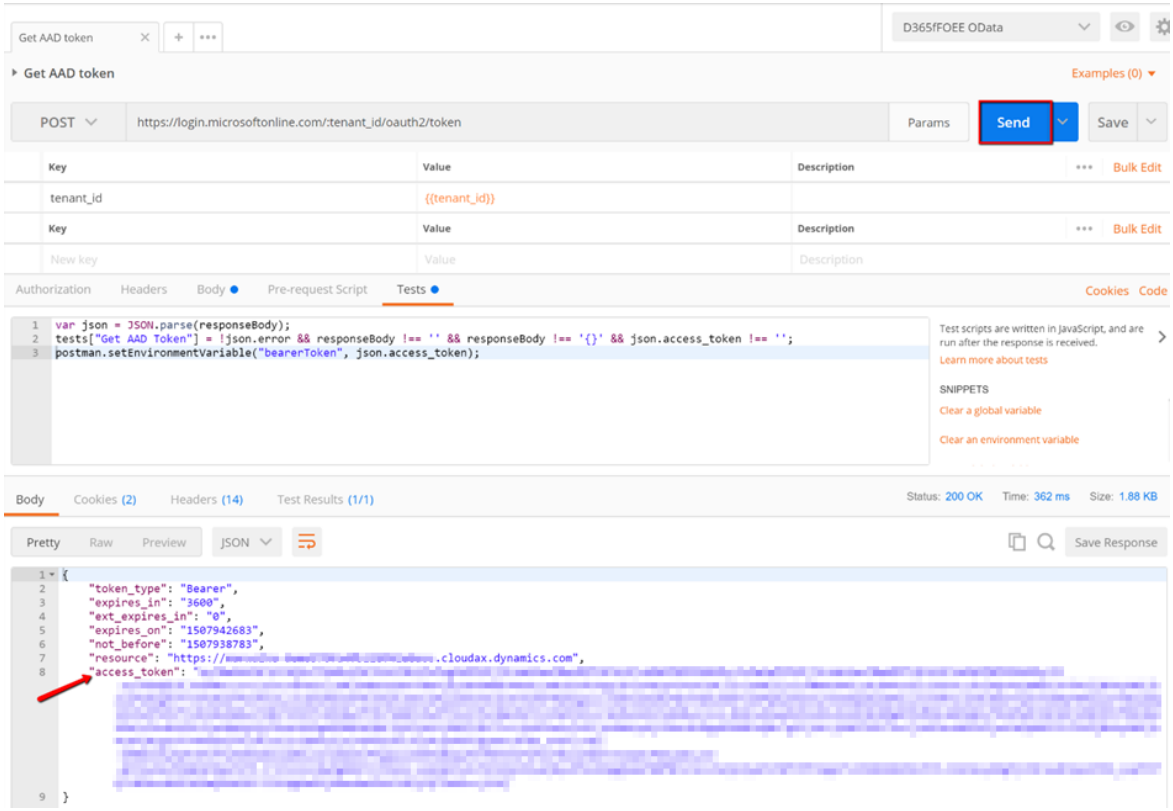


10. On the **Tests** tab, create a test that validates that the response is reasonable, and that stores the returned authorization token in an environment variable. Here is an example.

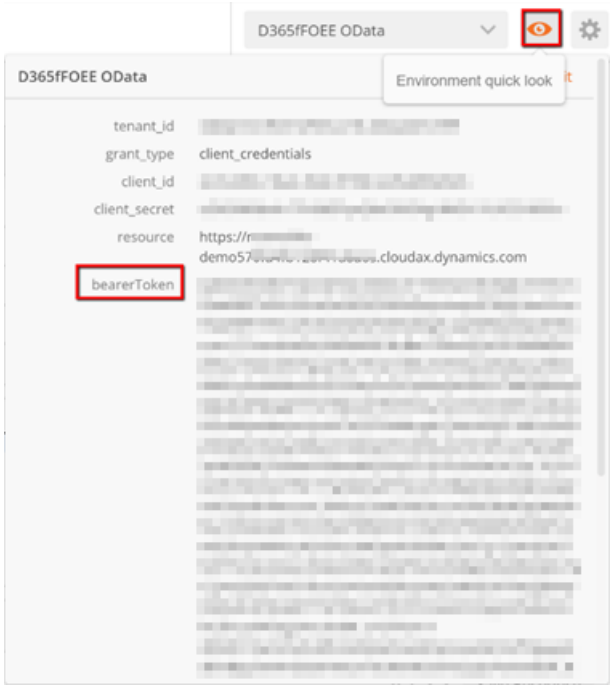
```
var json = JSON.parse(responseBody);
tests["Get Azure AD Token"] = !json.error && responseBody !== '' && responseBody !== '{}' &&
json.access_token !== '';
postman.setEnvironmentVariable("bearerToken", json.access_token);
```

11. Select **Save**, enter a name and collection for the request, and then select **Save** again.

12. Select **Send** to make the authorization request. The **Body** tab should now contain an Azure AD token together with other response details.



13. Because of the test code, the token is now in an environment variable. You can see that the token is an environment variable by selecting the **Environment quick look** button (the eye button).

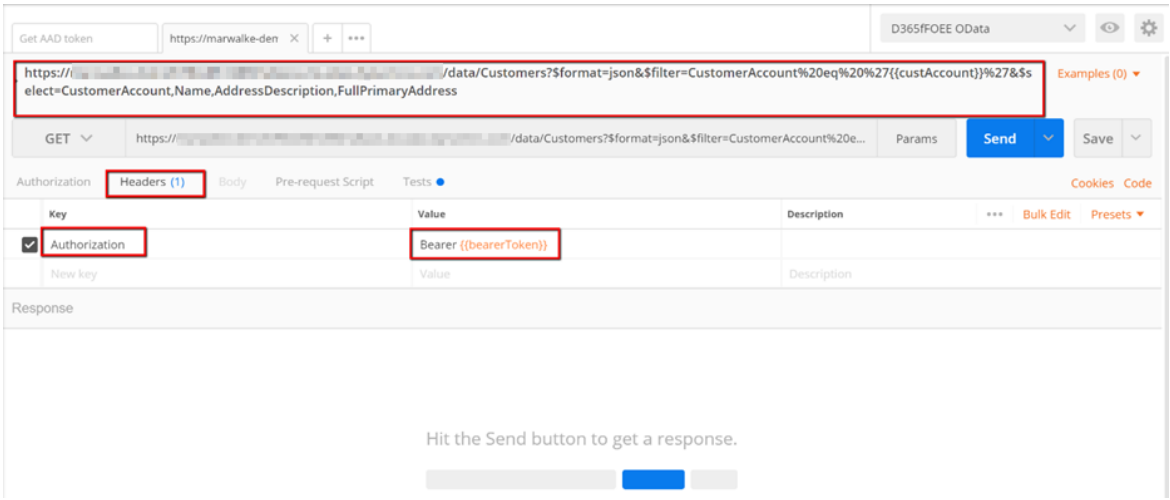


14. Create a request to perform create, read, update, or delete (CRUD) operations on the desired data entity

via the OData service. Create the URL according to your requirements. For more information, see Open Data Protocol (OData). You might find it useful to parameterize the request by using a variable that is stored in the environment, as shown earlier. The following example of a GET query uses a **Customer Account** parameter. The query returns name and address details for the customer account that is specified in the environment variable. Note that special characters must be correctly URL-encoded.

```
https://[Finance and Operations instance URL]/data/Customers?
$format=json&$filter=CustomerAccount%20eq%20%27{{custAccount}}%27&$select=CustomerAccount,Name,Addres
sDescription,FullPrimaryAddress
```

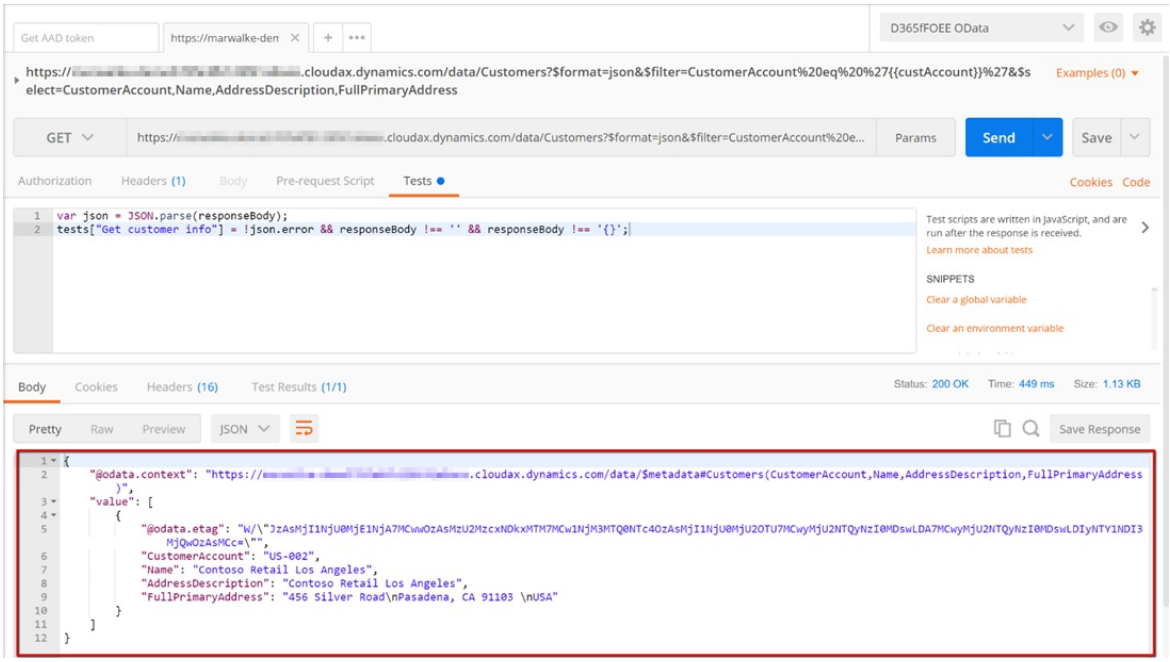
15. Add an Authorization header that refers to the authorization token that was retrieved earlier and stored in the **bearerToken** environment variable. The token must be prefixed by **Bearer** in the header.



16. Create a test to help validate the response. The following example tests that non-empty, JSON-formatted data is returned in the response body.

```
var json = JSON.parse(responseBody);
tests["Get customer info"] = !json.error && responseBody !== '' && responseBody !== '{}';
```

17. Save and send the request, and then verify the result. You must ensure that the user account being used is set to a default company that has data. Alternatively, you can also specify cross-company=true as the query parameter in the OData request.



In our example, we have now successfully authenticated and then used the OData service to read a customer record.

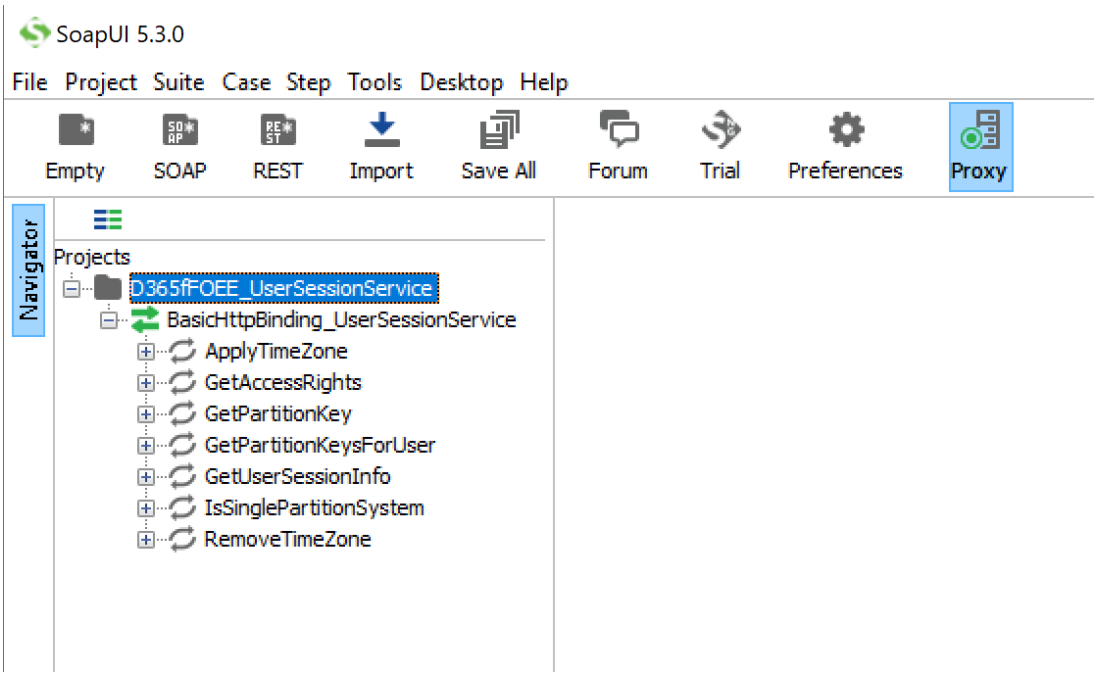
Query the SOAP custom service in your application by using SoapUI

SoapUI (https://www.soapui.org/) is a tool that is often used to interact with SOAP and REST web services in scenarios that involve API development and testing. This procedure isn't an endorsement of SoapUI, and other similar tools are available. However, we are using SoapUI to illustrate the concepts and messages that are involved when you use OAuth to authenticate with Azure AD, and then make SOAP requests to and receive responses.

1. Start SoapUI, and select the **SOAP** button to create a project.
2. Complete the information for the project:
   - In the **Project Name** field, enter a name for the project.
   - In the **Initial WSDL** field, enter the service address, and add the suffix **?wsdl**. (The service address should be in the format [Finance and Operations instance base URL]/soap/services/[service group name].) For more information, see the Services home page.

     For example, we are querying the user session service at the URL `https://[Finance and Operations base URL]/soap/services/UserSessionService?wsdl`.
   - Select the **Create sample requests for all operations?** check box.

     Because you selected to create sample requests, one sample request is created for each service operation that is available.



3. Right-click the new project, and then select **New TestSuite** to create a test suite. This test suite will generate a POST request for an Azure AD authorization token.
4. Right-click the test suite, and then select **New TestCase**.
5. Expand the test case, right-click **Test Steps**, select **Add Step**, and then select **HTTP Request**.
6. Enter a name for the request, and then select **OK**.
7. Enter a name for the test step. The endpoint that you should use for the POST request is

```
[https://login.microsoftonline.com/[tenant_id]/oauth2/token]
(https://login.microsoftonline.com/%5btenant_id%5d/oauth2/token)
```

.

8. Use the plus sign (+) button next to **Parameters** to add the following values.

PARAMETER	VALUE
grant_type	client_credentials
client_id	The application ID from the Azure AD application registration
client_secret	The secret key value from the Azure AD application registration
resource	The URL of the instance without the trailing '/'

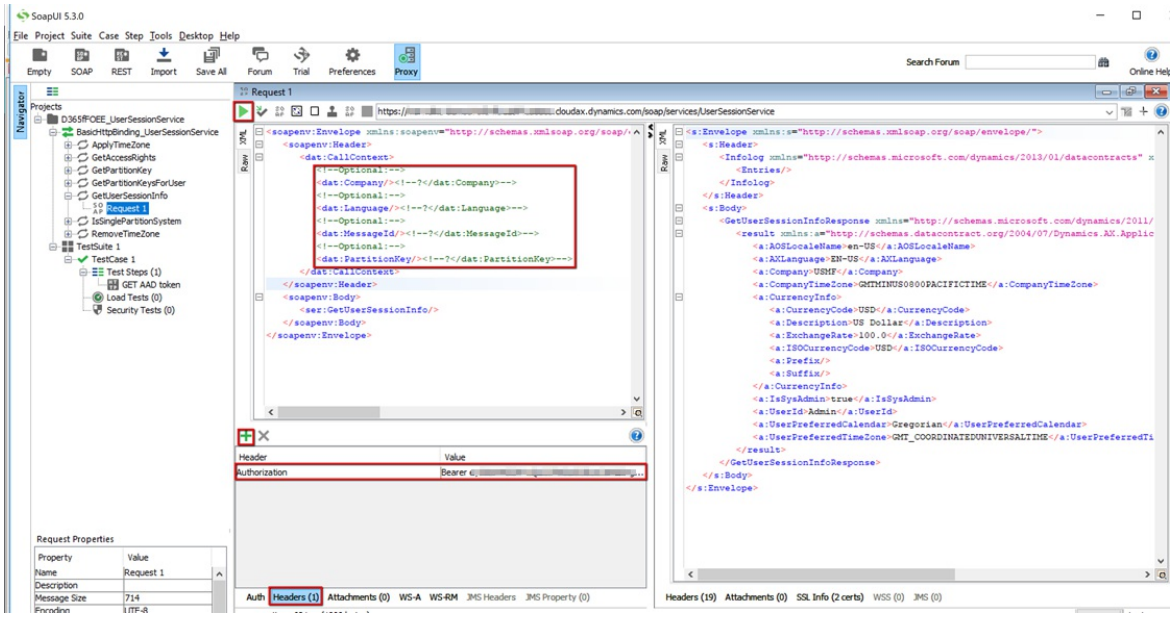
9. To make sure that the parameters are in the POST body, select **Post QueryString**, and then select **Play**. An access token should be returned in the response pane. The values will be most readable if you use the **JSON response** tab. Copy the access token so that you can use it in the authorization header of subsequent requests.

10. Go back to the first request node under the **GetUserSessionInfo** SOAP sample request. In the request pane on the left, select the plus sign (+) button to add a header that is named **Authorization**. Paste the access token into the **Value** field, and add the prefix **Bearer**.

11. The sample requests that SoapUI creates won't work unless you modify them. You must edit the call context and body so that they are consistent with the schema for what you're trying to do.

    For our simple scenario, you can edit the optional call context elements so that they are null-valued. Insert a forward slash (/) before the greater than sign (>) in the opening tags. Then comment out the question marks (?) and the closing tags by using the standard <!--...--> syntax to delimit the start and end of the comments. (Question marks aren't valid content for the XML schema.) Alternatively, you can just delete the question marks (?) so that the context elements are empty.

12. The SOAP request is now ready. Select **Play**, and validate the result on the right.



In our example, we have now successfully authenticated and then queried UserSessionService via SOAP.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Create read-only entities that expose financial dimensions

2/18/2021 • 5 minutes to read • Edit Online

In this topic, we describe how to build an entity for registered transactions that are registered.

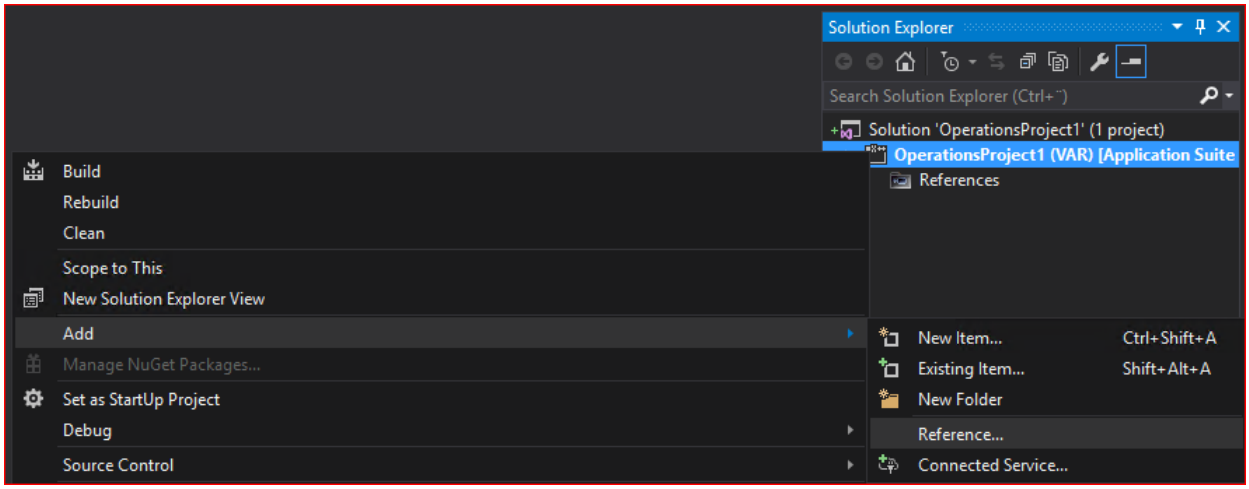
> **NOTE**
>
> This topic comes from Per Baarsoe Jorgensen of the Solutions Architecture team. It describes a real-world scenario that we have encountered as we work with customers.

Imagine a scenario where we must expose all vendor invoice line transactions together with the financial dimensions that were applied through the distributions. Because easy consumption by a third-party tool is essential, we will create an entity for this scenario. As a result, the entity should not have to be joined with other related entities but should be able to provide value on its own.

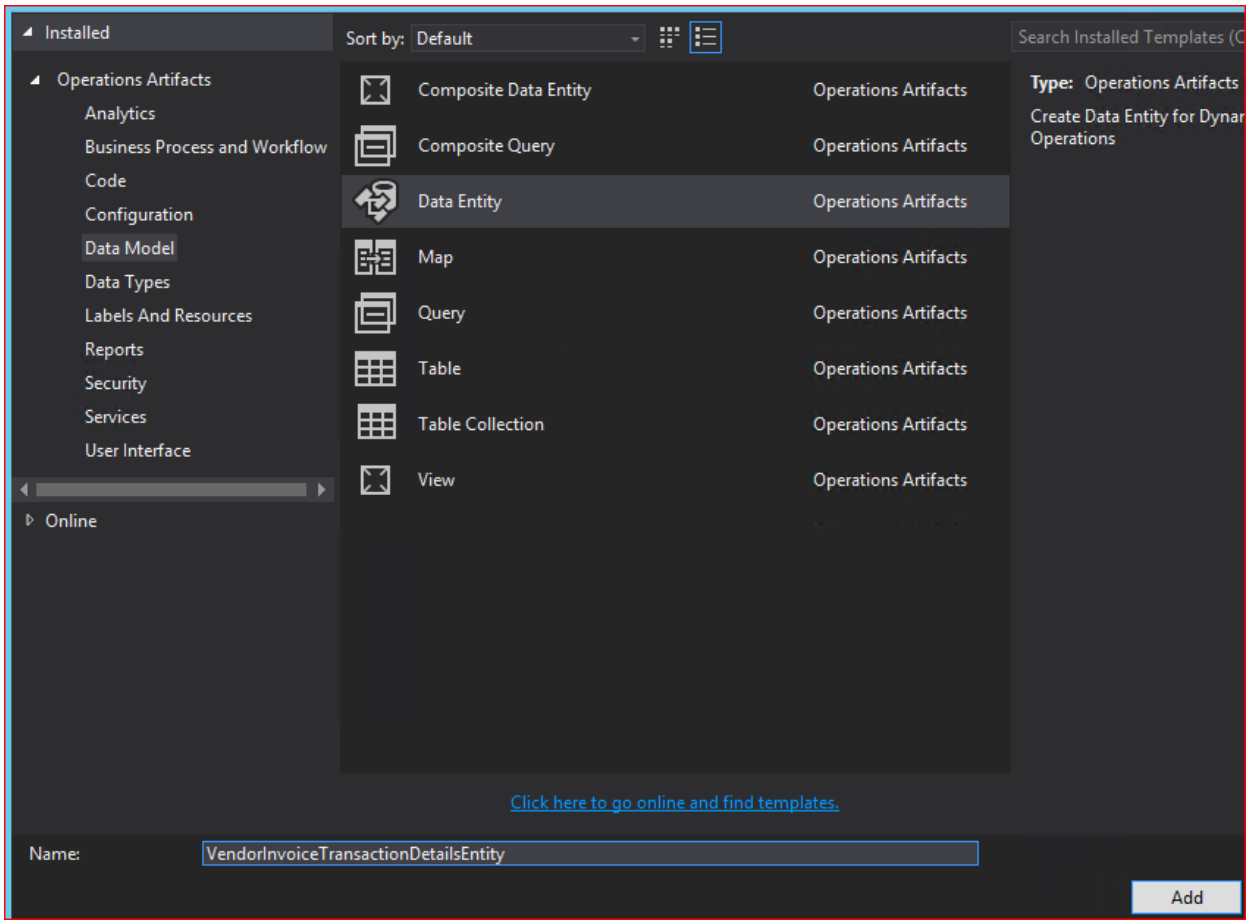
We will walk through the process of creating a sample entity to meet these requirements. (We will leave out instructions for integrating with Microsoft Azure DevOps, because those steps are already well documented.)

Create a basic entity

The first step is to create a new element in a project by selecting **New Item**.



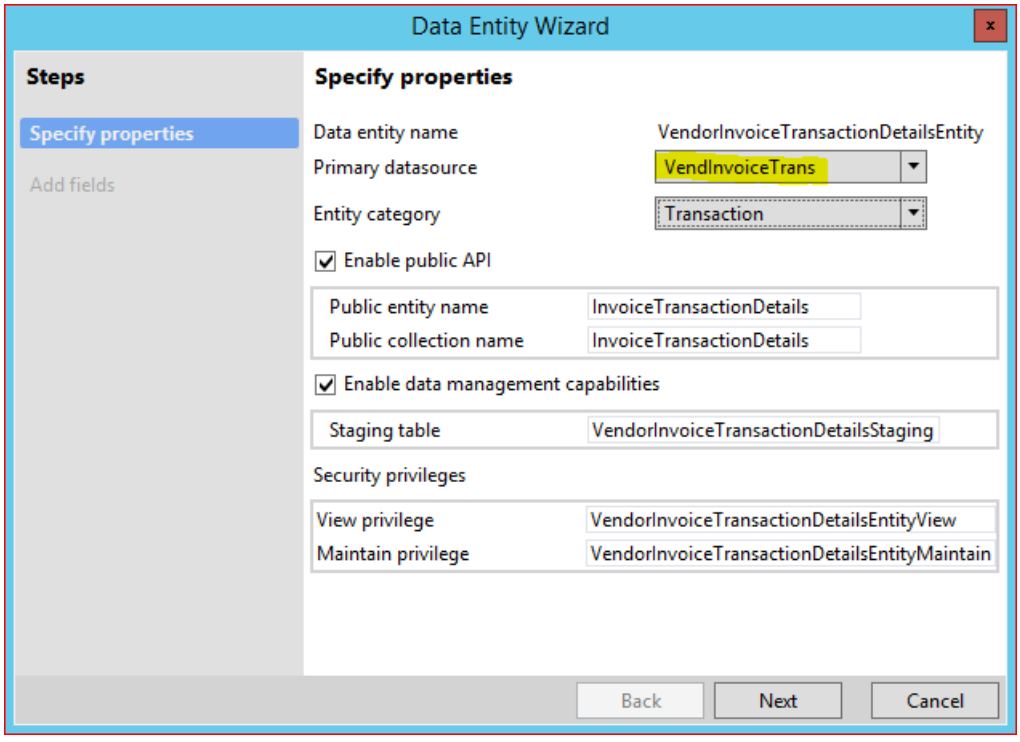
In the form that opens, under Data Model, we select the **Data Entity** element type.



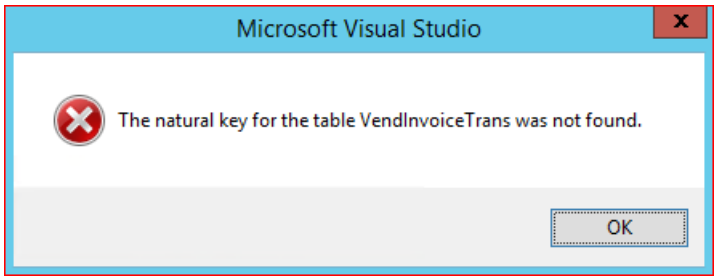
NOTE

Be careful when you name the entity, because you can't rename it later.

Next, in the Data Entity wizard, we select the appropriate primary data source. For our scenario, this data source is VendInvoiceTrans.

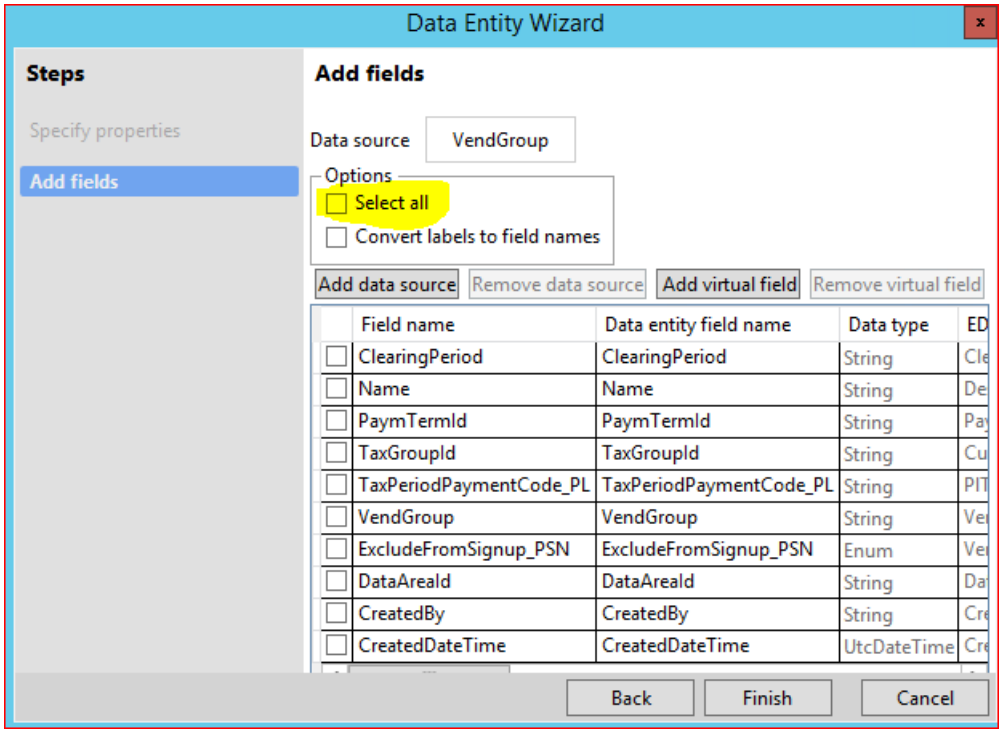


The wizard doesn't accept tables that don't have a natural key, as is the case with VendInvoiceTrans. Therefore, we receive the following error message.



The workaround is to select any other primary data source that has a natural key associated, such as VendGroup.

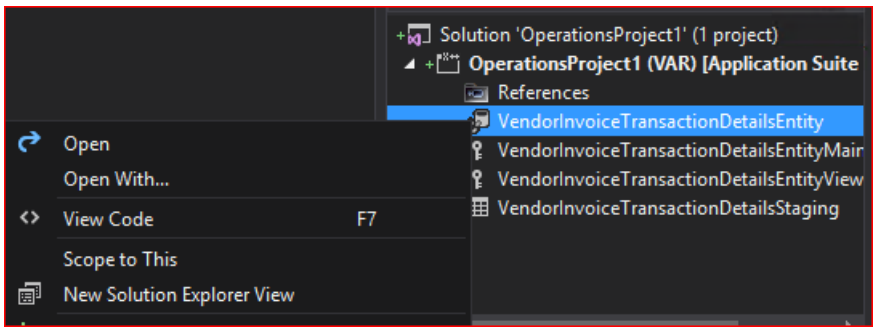
Because we don't really need this data source, we also don't need any fields for it. Therefore, we clear the **Select all** check box.



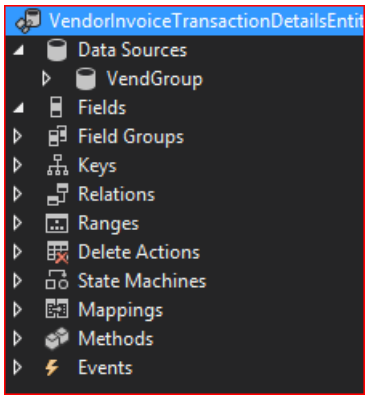
Finally, we create the template for our entity by clicking **Finish**.

Customize the basic entity

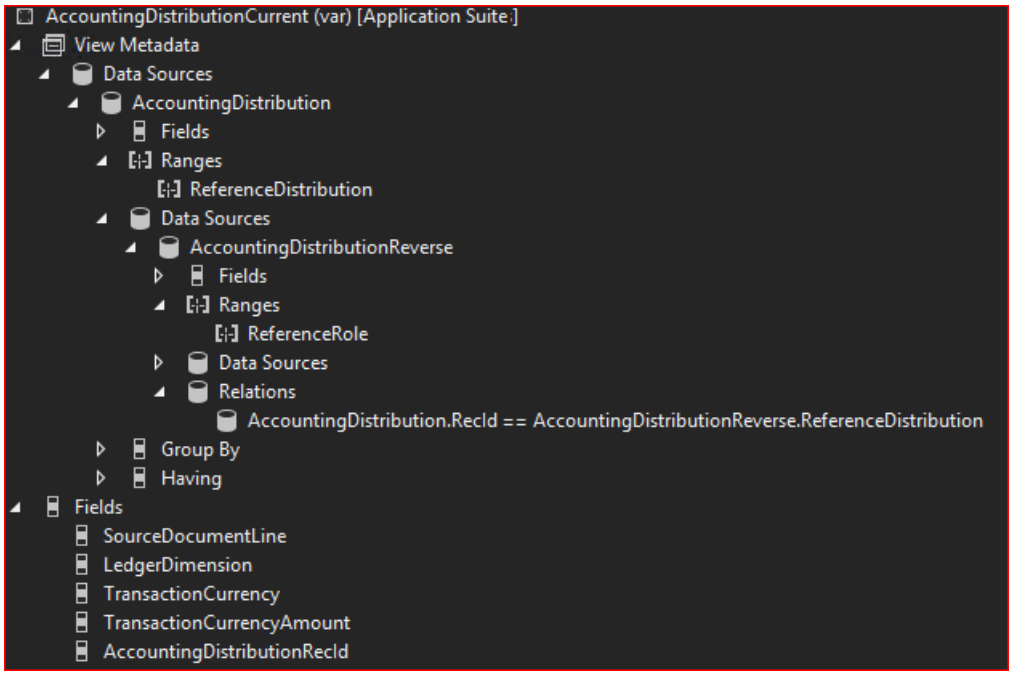
The entity, staging table, and security assets have been created, and we can now produce our custom entity. In the project, we open the VendorInvoiceTransactionDetailsEntity entity in the designer.



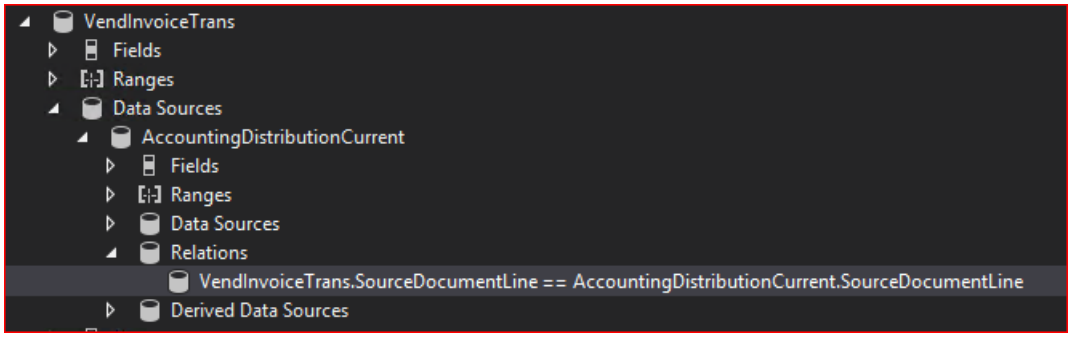
In the designer, we replace the dummy table (VendGroup) that we applied in the wizard with the transaction table VendInvoiceTrans. Because we didn't choose to add the fields, we don't have to remove fields in the entity.



> **NOTE**
>
> Because we are exposing transactional data by using this entity, it's important that we mark the entity as read-only. Therefore, we set the **Is read only** property to **Yes** on the top node. Because accounting distributions are versioned, it's important that only the current version be returned when we query. Therefore, we create a view that makes this part easily reusable across multiple entities.



In the properties, we assign **ReferenceDistribution** a range filter value of **0** and **ReferenceRole** a range filter value of **1**. The join mode property for the AccountDistributionReverse data source must be **NoExistsJoin**. After the new view is in place, we can add it to the VendorInvoiceTransactionDetailsEntity entity that we are currently building.



Expose financial dimensions as fields

The next important step is to expose the financial dimensions as separate fields on the entity. Because our

scenario builds on top of a posted transaction, we must add the fields to the DimensionCombinationentity entity. We want to make the adjustments in a resilient manner by using the extension approach, so that minimal maintenance will be required when we upgrade the code base to newer versions in the future.

Microsoft Dynamics 365 for Finance and Operations, Enterprise edition version 1611

For version 1611 or later, you should use the wizard that is available in Microsoft Visual Studio (at **Dynamics 365** > **Addins** > **Add financial dimensions for Odata**). For instructions, see Add dimensions to Excel templates.

Earlier versions

If you're working with earlier versions, you must complete the steps that are outlined here. First, we add the entity extension itself. Select **Create extension** on the context menu (shortcut menu). Next, we create the code that retrieves the data. Because the entity extension is already in place, we must create a new class. The following example adds code for an arbitrary dimension that is named **ProductLine**.

```
[ExtensionOf(dataentityviewstr(DimensionCombinationentity))]
  public final class DimensionCombinationentity_Extension
  {
    private static server str getEmptyOrDimensionValueSqlString(str _attributeName)
    {
        str sqlStatement;

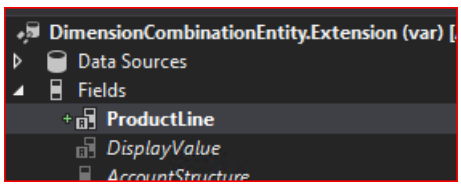
        DimensionAttribute dimensionAttribute = DimensionAttribute::findByName(_attributeName);

        if (!dimensionAttribute)
        {
            sqlStatement = SysComputedColumn::returnLiteral('');
        }
        else
        {
            sqlStatement = strFmt('SELECT TOP 1 T1.%1 ', dimensionAttribute.DimensionValueColumnName);
        }

        return sqlStatement;
    }

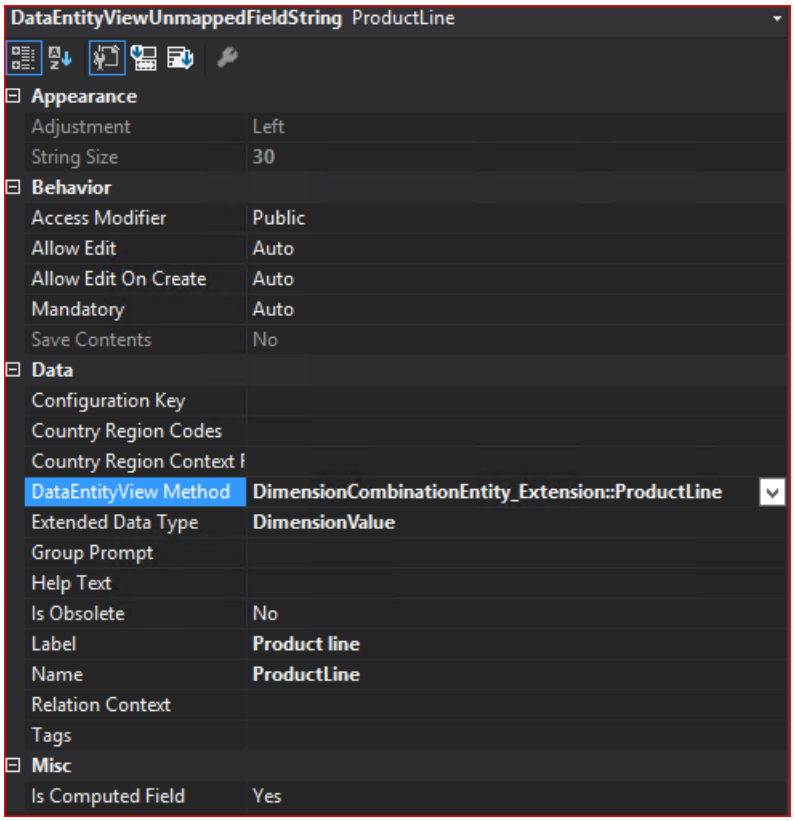
    /// Generates the sql to populate the FOTA view field.
    public static server str ProductLineValue()
    {
        return DimensionCombinationentity::getEmptyOrDimensionValueSqlString('ProductLine');
    }
  }
```

We now add fields to the newly created entity extension by using custom fields that reference these methods.



Next, we set the property values to reflect the fact that the field is unmapped and should be retrieved through the code that we made for the extension class. When you create the relation, also set the following values:

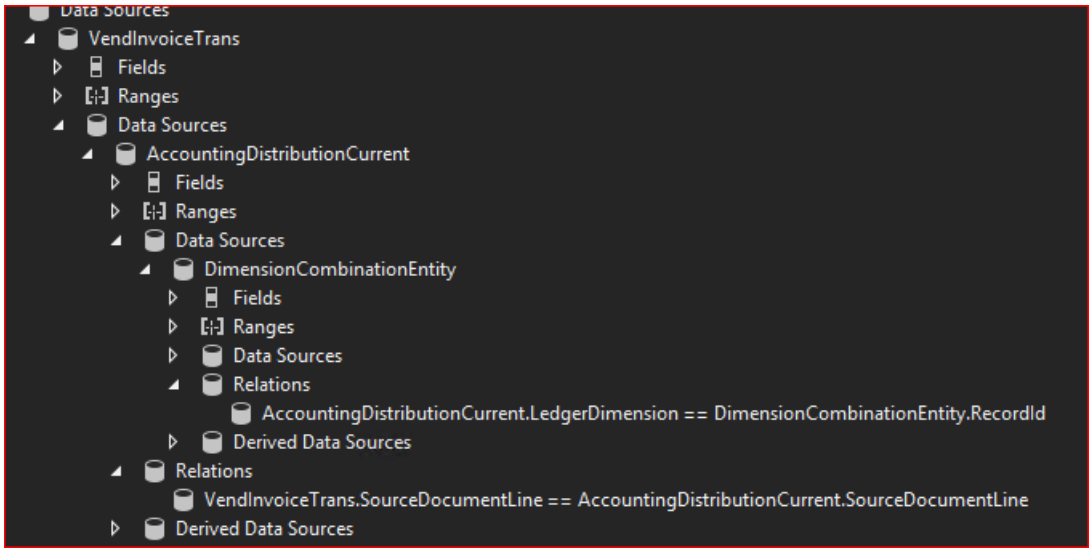
- Cardinality: **ZeroMore**
- Related data entity: **DimensionCombinationentity**
- Related data entity cardinality: **ZeroOne**
- Relationship type: **Association**



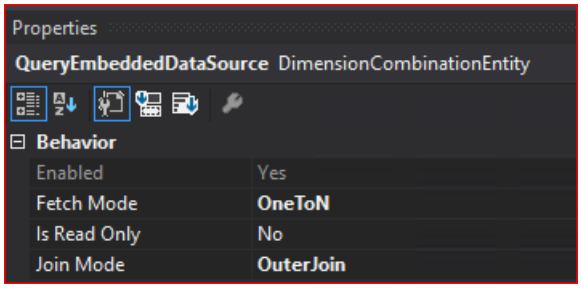
NOTE

We must fully qualify the data method with the class name.

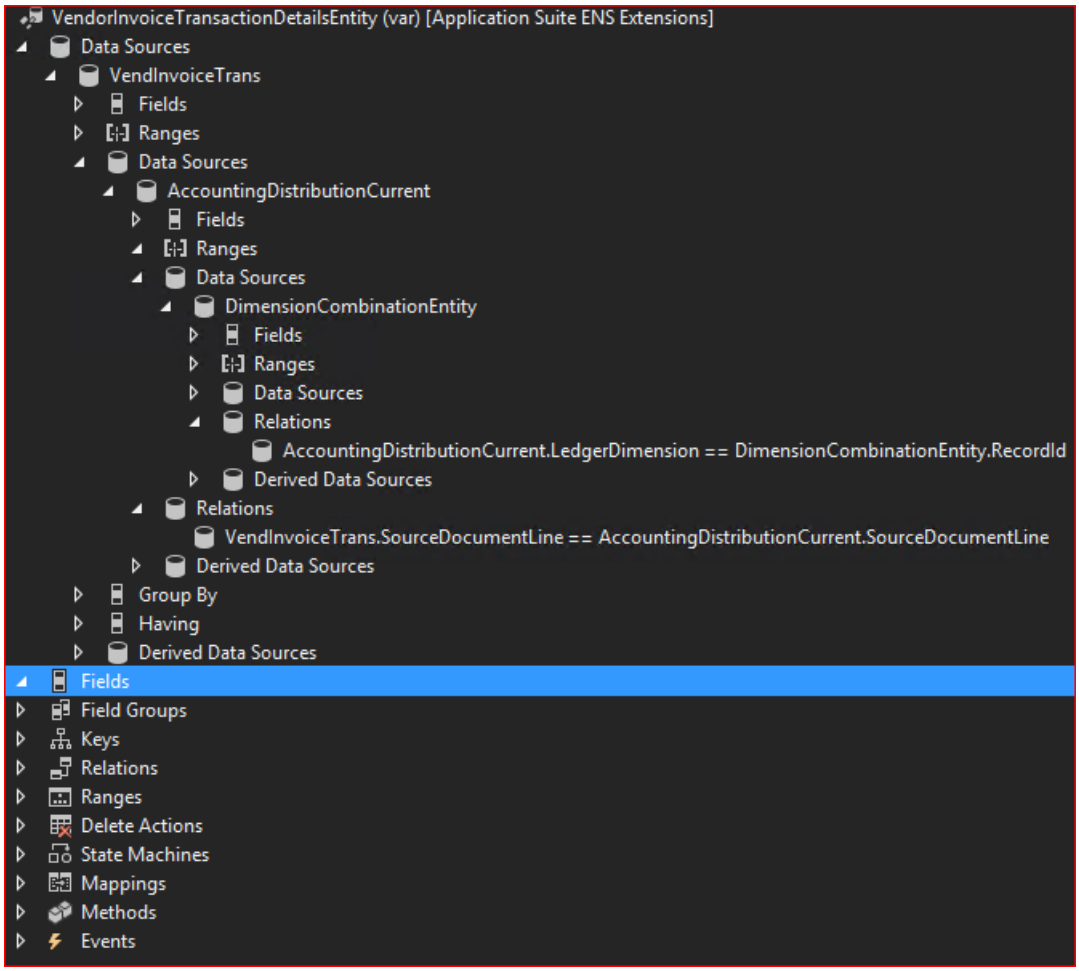
We are now ready to add the DimensionCombinationentity entity to our new VendInvoiceTransactionentity entity.



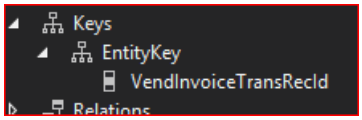
Notice that both the AccountingDistributionCurrent and the DimensionCombinationentity entity should be outer-joined.



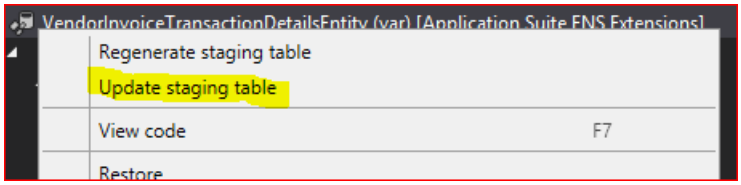
Now, we just have to drag the required fields from the various data sources to the **Fields** node on the new entity (that is, to our newly created ProductLine).



We should add a key to the entity to enable the incremental update functionality during the export configuration. Therefore, we must make sure that RecId from the VendInvoiceTrans data source is part of the fields named e.g. VendInvoiceTransRecId. After the field is in the field list, we can drag it to the **EntityKey** node.



To make sure that the staging table is associated with the fully configured entity, we must update it. On the context menu for the entity, we select **Update staging table**.



The entity work is now complete, and we can build it.

> **NOTE**
>
> In this scenario, a LedgerDimension was associated with the DimensionCombinationentity entity. In scenarios where there is a DefaultDimension, we must associate it with the DimensionSetentity entity. The improvements and extensions that are required are identical to the improvements and extensions that we made to the DimensionCombinationentity entity.

Additional resources

Export Dynamics AX 7 Entities to your own Azure SQL Database

Bring your own database (BYOD)

2/18/2021 • 13 minutes to read • Edit Online

This topic explains how administrators can export data entities from the application into their own Microsoft Azure SQL database. This feature is also known as *bring your own database* (BYOD). The BYOD feature was released in Microsoft Dynamics AX with platform update 2 (August 2016). Minor improvements and bug fixes have been included in subsequent platform updates.

The BYOD feature lets administrators configure their own database, and then export one or more data entities that are available in the application into the database. (Currently, more than 1,700 data entities are available.) Specifically, this feature lets you complete these tasks:

- Define one or more SQL databases that you can export entity data into.
- Export either all the records (*full push*) or only the records that have changed or been deleted (*incremental push*).
- Use the rich scheduling capabilities of the batch framework to enable periodic exports.
- Access the entity database by using Transact-SQL (T-SQL), and even extend the database by adding more tables.

Entity store or BYOD?

If you followed the series of blog posts about Microsoft Power BI integration, you will be familiar with Entity store. Entity store is the operational data warehouse. Entity store provides built-in integration of operational reports with Power BI. Ready-made reports and analytical workspaces use Entity store. If you write Power BI reports by using data in your application environment, you should use Entity store.

However, the BYOD feature is recommended for the following scenarios:

- You must export data into your own data warehouse.
- You use analytical tools other than Power BI, and those tools require T-SQL access to data.
- You must perform batch integration with other systems.

> **NOTE**
>
> The application doesn't allow T-SQL connections to the production database. If you're upgrading from a previous version of Finance and Operations, and you have integration solutions that require direct T-SQL access to the database, BYOD is the recommended upgrade path.

You can use either Entity store or BYOD. The default operational reports that are available take advantage of embedded Power BI and Entity store. We recommend that you use our default operational reports as your first choice. You can also extend the ready-made operational reports to meet your requirements. You should consider BYOD a complementary option that you use as you require.

Creating a SQL database

Before you can configure the entity export option and use the BYOD feature, you must create a SQL database by using Azure portal.

For one-box development environments, you can create a database in the local Microsoft SQL Server database. However, this database should be used only for development and testing purposes. For production

environments, you must create an Azure SQL database.

You should also create a SQL user account for sign-in to the database. Write down the server name, database name, and the SQL user ID and password. You will use this information when you configure the entity export option in the next section.

If you're using the BYOD feature for integration with a business intelligence (BI) tool, you should consider using clustered columnstore indexes (CCIs). CCIs are in-memory indexes that improve the performance of read queries that are typical in analytical and reporting workloads.

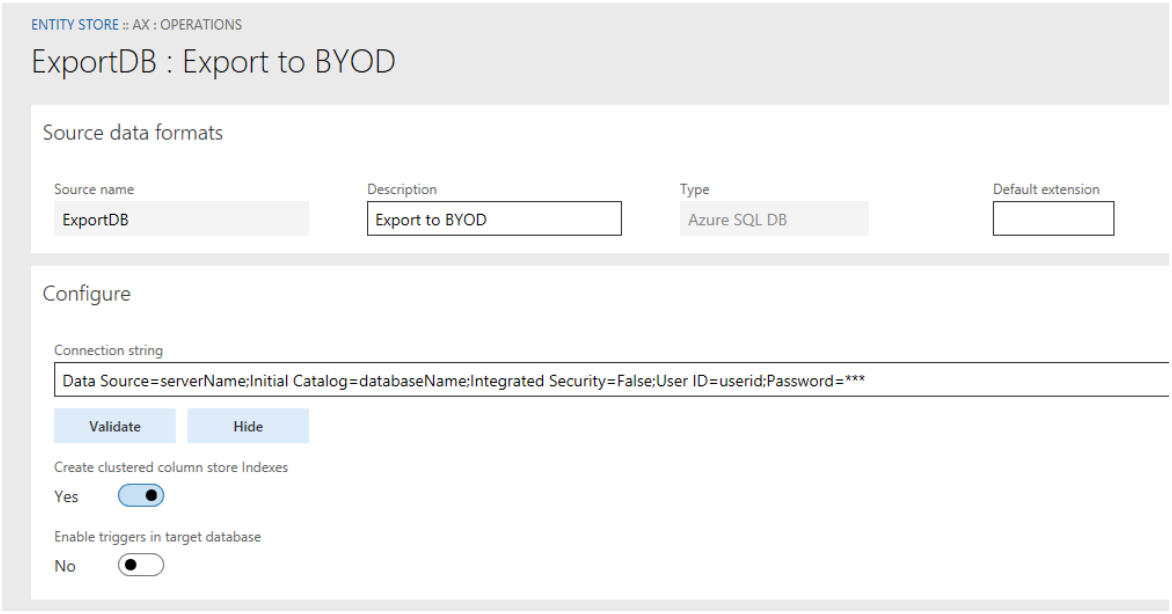
> **NOTE**
>
> Your BYOD database must be accessible to Finance and Operations apps. If you encounter issues where you are unable access to access BYOD, you must ensure firewall rules in your BYOD are configured appropriately.

Configuring the entity export option

1. Start the client, and then, in the **Data management** workspace, select the **Configure Entity export to database** tile.
2. If you've configured any databases, a list is shown. Otherwise, you must configure a new database. In this case, select **New**, and then enter a unique name and a description for the new database. Note that you can export entities into multiple databases.
3. Enter the connection string in the following format:

   Data Source=<logical server name>,1433; Initial Catalog=<your DB name>; Integrated Security=False; User ID=<SQL user ID>; Password=<password>

   In this connection string, the logical server name should resemble **nnnn.database.windows.net**. You should be able to find the logical server name in Azure portal. The following illustration shows an example of a connection string.



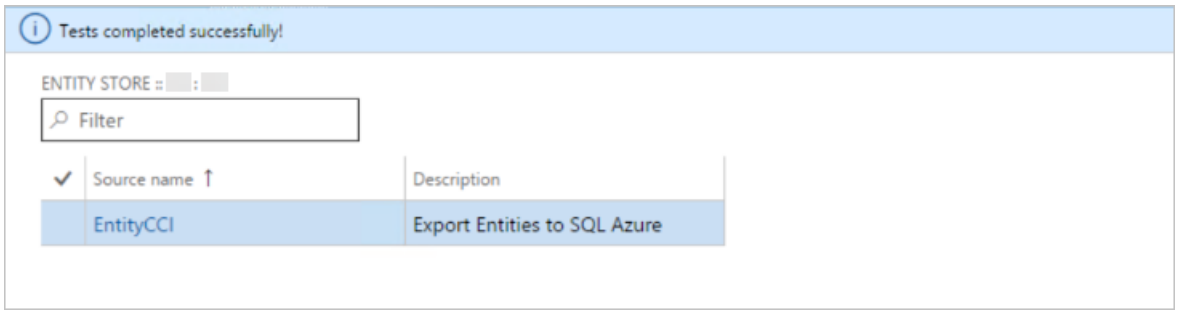
> **NOTE**
>
> The default extension field shown in the image above does not apply to BYOD.

4. Select **Validate**, and make sure that the connection is successful.

- The **Create clustered column store indexes** option optimizes the destination database for selected queries by defining CCIs for entities that are copied. However, CCIs are currently supported only on SQL premium databases. Therefore, to enable this option, you must create a SQL premium database.
- The **Enable triggers in target database** option sets export jobs to enable SQL triggers in the target database. This option lets you hook downstream processes into the trigger to orchestrate actions that must be started after records have been inserted. One trigger is supported per bulk insert operation. The size of the bulk insert is determined by the **Maximum insert commit size** parameter in the Data management framework.

For scenarios in which reporting systems read data from BYOD, there is always the challenge of ensuring that the reporting systems get consistent data from BYOD while the sync is in progress. You can achieve this result by not having the reporting systems read directly from the staging tables created by the BYOD process. The staging tables hold the data while data is being synced from the instance and hence will be constantly changing. Use the SQL trigger feature to determine when the data sync has been completed, and then hydrate the downstream reporting systems.

When the validation is passed, the database that you configured for entity export appears in lists of databases, as shown in the following illustration.



You can now publish one or more entities to the new database by selecting the **Publish** option on the menu.

Publishing the entity schema to the database

The **Publish** page enables several scenarios:

- Publish new entities to the database.
- Delete previously published entities from the database. (For example, you might want to re-create the schema.)
- Compare published entities with the entity schema. (For example, if new fields are added later, you can compare the fields with your database schema.)
- Configure change tracking functionality that enables incremental updates of your data.

The following sections discuss each option.

Publish

The **Publish** option defines the entity database schema on the destination database. When you select one or more entities, and then select the **Publish** option, a batch job is started. This job creates the entities in the destination database. When the database definition job is completed, you receive a message, which you can access by using the bell symbol in the upper right.

The actual data update occurs when you export data. At this point, you're just creating the schema.

Drop entity

The **Drop entity** option deletes the data and the entity definition from the destination database.

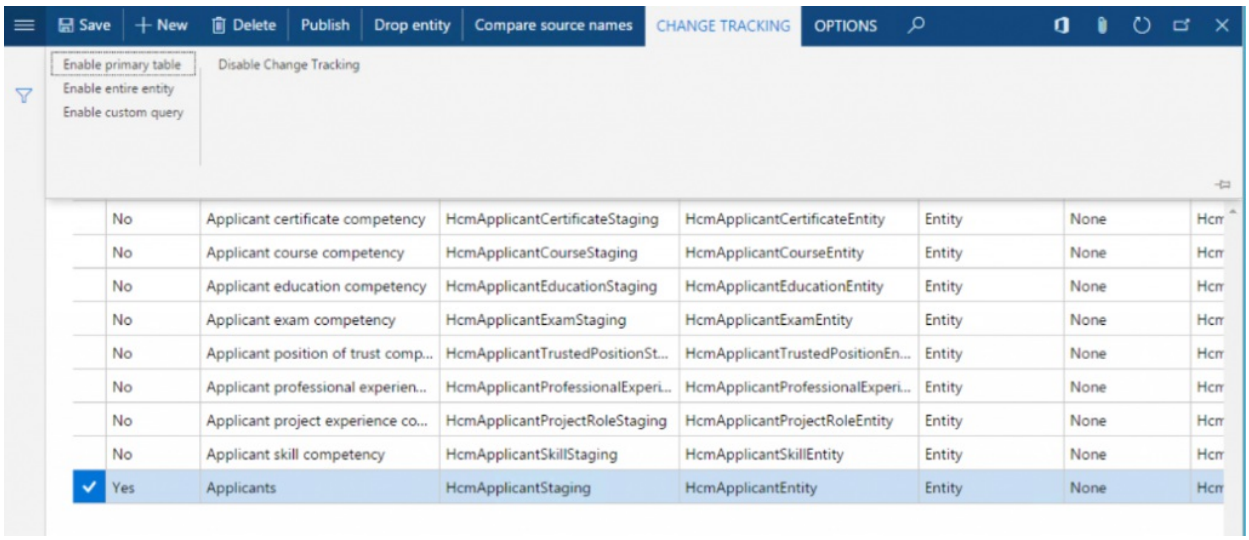
Compare source names

The **Compare source names** option lets you compare the entity schema in the destination with the entity schema in the application. This option is used for version management. You can also use this option to remove any unwanted columns from the destination table.

Configure change tracking

Change tracking is a feature that is provided in SQL Server and SQL Database. Change tracking enables the database to track changes including deletes that are made on tables. The system uses change tracking to identify changes that are made to tables as transactions. However, because the application must track changes at the data entity level, there is additional logic on top of SQL change tracking to make this functionality work. The steps to enable change tracking are explained later in this section.

The **Change tracking** option on the **Publish** page lets you configure how changes are tracked on the underlying entity.



The following table describes the change tracking options that are available.

OPTION	DESCRIPTION
Enable primary table	An entity consists of several tables. Select this option to track all changes that are made to the primary table of the entity. When changes are made to the primary table, the corresponding record is inserted into or updated in the destination database. Although data from the whole entity is written to the destination table, the system triggers the insert or update option only when the primary table is modified.
Enable entire entity	Select this option to track all changes to the entity. (These changes include changes to all the tables that make up the entity.) When changes are made to the entity, corresponding updates are made to the destination.
Enable custom query	This option lets a developer provide a custom query that the system runs to evaluate changes. This option is useful when you have a complex requirement to track changes from only a selected set of fields. You can also select this option when the entities that will be exported were built by using a hierarchy of nested views. For more information, see Enable change tracking for entities.

To use change tracking, you must enable the **Change tracking** option as shown above in data management. This action is available on the **Data entities** list page, by going to **Data management > Data entities**. You need to select an entity and select from one of the options listed above to enable change tracking on the data entity.

If you republish an entity that exists in the destination database, the system warns you that existing data will be

deleted because of the new operation.

When you confirm the publish operation, the system publishes the schema to the database, and you're notified when the operation is completed.

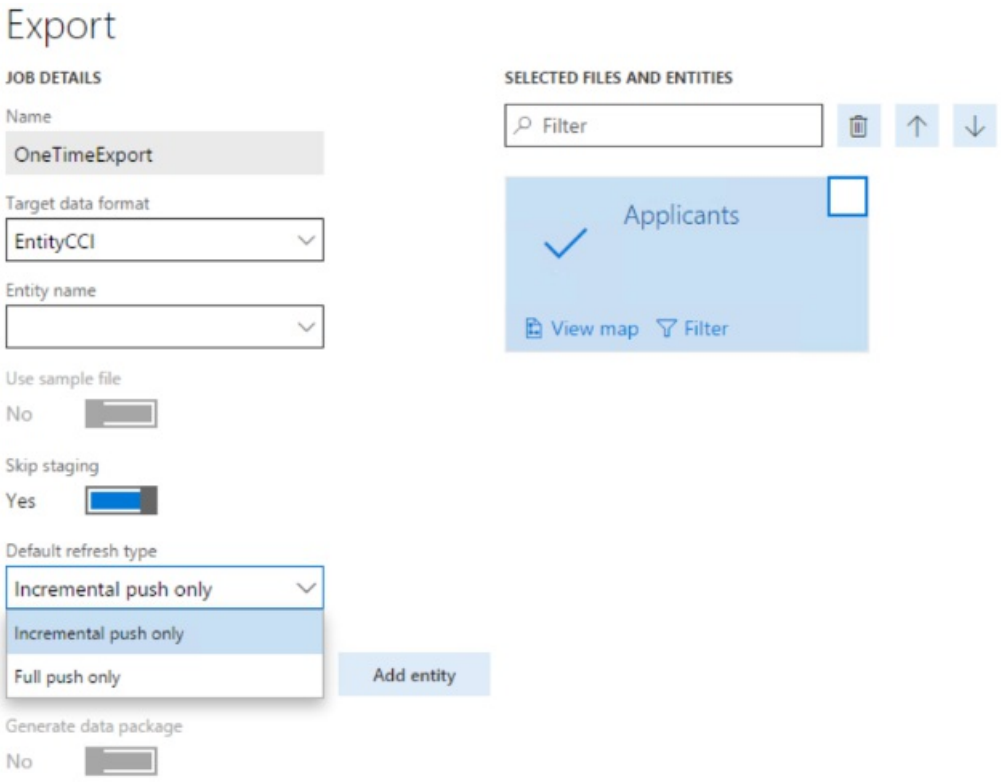
By selecting the **Show published only** option on the **Publish** page, you can show only the entities that were published to a given destination database. The Publish function creates the entity schema in the database. You can navigate to the database and see the table schemas that were created, together with corresponding indexes.

> **NOTE**
> Currently, you can't use BYOD to export composite entities into a database. You must export each entity in the composite entity.

Exporting data into your database

After entities are published to the destination database, you can use the Export function in the **Data management** workspace to move data. The Export function lets you define a Data movement job that contains one or more entities.

You can use the **Export** page to export data into many target data formats, such as a comma-separated values (CSV) file. This page also supports SQL databases as another destination.



You can create a data project that has multiple entities. You can schedule this data project to run by using the batch framework. You also schedule the data export job to run on a periodic basis by selecting the **Export in batch** option.

The same job can also be used to export data from all companies. In prior to Platform update 27, this feature can be enabled by enabling the flight DMFEnableAllCompanyExport as explained in Data management overview. Starting in Platform update 27, this feature can be enabled in data management framework parameters. After the feature is enabled, a new option will appear when adding an entity to a data project. This option can be enabled to export data from all companies for the specific entity.

> **NOTE**
>
> Adding multiple entities to an export project for BYOD must be done carefully to ensure the overall reliability of the BYOD export is not compromised. Different parameters must be taken into consideration when deciding the number of entities that are added to the same project. Some of these parameters should be the degree of complexity of the entities, data volume per entity that is expected, and the overall time for export to complete at the job level. Adding hundreds of entities must be avoided, therefore creating multiple jobs with smaller number of entities is recommended.
>
> Use of recurring exports in **Manage > Manage recurring data jobs** for BYOD is discouraged. You must use the **Export in batch** option.

Incremental export

When you add an entity for data export, you can select to do an incremental export (which is also known as incremental push) or a full push. For incremental push to work, you must enable the **Change tracking** option in the database and specify an appropriate change tracking option, as described earlier in this topic.

> **NOTE**
>
> A full push deletes all existing records from an entity and then inserts the current set of records from the selected entity.

If you select an incremental push, the first push is always going to be a full push. This is because SQL needs to know which records have been 'tracked' in order to be able to track subsequent changes. Whenever a new record is inserted, or a record is added or deleted, the corresponding change will be reflected in the destination entity.

Because the first push is always a full push, we do not recommend that you do an explicit full push before you enable change tracking.

We recommend that you first enable change tracking and schedule a export job with incremental push. This will take care of the first full push and the subsequent incremental exports.

Timeouts

The default timeouts for BYOD exports are set to ten minutes for truncation operations and one hour for actual bulk insert operations. When volumes are high, these timeout settings may not be sufficient and must be updated. Starting with the release of Platform update 18, you can update the timeout settings by navigating to **Data management > Framework parameters > Bring your own database**. These timeouts are company specific and must be set separately for each company.

Known limitations

The BYOD feature has the following limitations.

There should be no active locks on your database during synchronization

Because BYOD is your own database, you must ensure that there are no active locks on your Azure SQL database when data is being synced. Having active locks on your database during synchronization can result in slow writes or even failure to export to your Azure SQL database.

You can't export composite entities into your own database

Currently, composite entities aren't supported. You must export individual entities that make up the composite entity. However, you can export both the entities in the same data project.

Entities that don't have unique keys can't be exported by using incremental push

You might face this limitation especially when you try to incrementally export records from a few ready-made entities. Because these entities were designed to enable the import of data, they don't have a unique key. However, you can enable change tracking only for entities that have a unique key. Therefore, there is a limitation on incremental push. One workaround is to extend the required entity and define a unique key.

Troubleshooting

Incremental push not working correctly

Issue - When a full push occurs for some entity then a large set of records can be seen in BYOD using a select statement. However, an incremental push results in only a few records in BYOD. It seems as if the incremental push deleted all the records and added only the changed records in BYOD.

Solution - In cases like this it is recommended to disable and re-enable change tracking for the entity in question. The state of the SQL change tracking tables might not be in the expected state. Also verify that there are no other incremental exports that cover the same tables (DMF, MR, Retail).

SSIS Error Code DTS_E_OLEDBERROR. An OLE DB error has occurred. Error code: 0x80004005

Issue - Export to BYOD fails with an SSIS exception shown below.

```
An OLE DB error has occurred. Error code: 0x80004005.

An OLE DB record is available. Source: "Microsoft SQL Server Native Client 11.0"  Hresult: 0x80004005
Description: "Communication link failure".

An OLE DB record is available. Source: "Microsoft SQL Server Native Client 11.0"  Hresult: 0x80004005
Description: "TCP Provider: An existing connection was forcibly closed by the remote host.

Failed to open a fastload rowset for <entityStaging>. Check that the object exists in the database.

OLE DB Destination failed the pre-execute phase and returned error code 0xC0202040.
```

Solution - This can occur if the connection policy on the Azure SQL BYOD server is set to Proxy. This must be changed to 'Redirect' as explained in SQL DB Connectivity Architecture

Automated Entity store refresh

2/18/2021 • 2 minutes to read • Edit Online

Overview

Entity store refresh is automated and managed by the system. Administrators do not need to schedule or monitor the Entity store refresh with the system batch schedules. The refresh operation is based on anticipated latency. This functionality is enabled in Platform update 23. As an administrator you do need to opt-in to use this feature.

Enable automated refresh

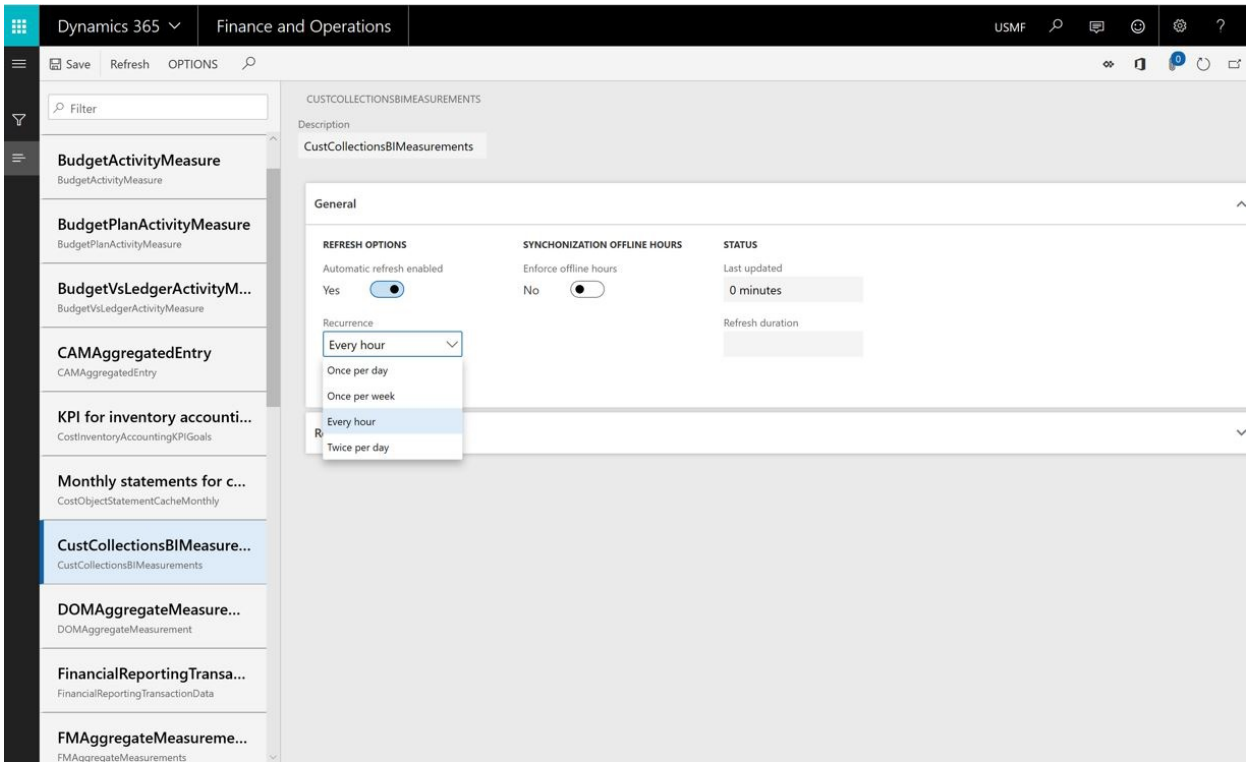
Complete the following steps to enable automated Entity store refresh.

1. Go to **System administration** > **Set up** > **Entity store**. On the **Entity store** page, a message indicates that you can switch to the **Automated Entity store refresh** option. This option is managed by the system. An admin does not have to schedule or monitor the Entity store refresh.
2. Select **Switch now**.

> **IMPORTANT**
> This action isn't reversible. After you switch to the **Automated Entity store refresh** option, you can't revert to the old user interface (UI) experience.

3. Select **Yes** to continue.

You will now see the new experience.



After the new experience is turned on, you can define the refresh for each aggregate measurement. The following refresh options are available:

- Every hour
- Twice a day
- Once a day
- Once a week

An admin can also refresh any aggregate measurement on demand by clicking the **Refresh** button. Additional options will be added in future platform updates. These options will include options for real-time refresh.

> **IMPORTANT**
>
> When automated refresh is enabled, the system can disable the refresh of aggregate measurements. You must revisit aggregate measurements and validate that appropriate refresh intervals have been applied.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Data task automation

2/18/2021 • 17 minutes to read • Edit Online

Data task automation lets you easily repeat many types of data tasks and validate the outcome of each task. Data task automation is very useful for projects that are in the implementation phase. For example, you can automate the creation and configuration of data projects. You can also configure and trigger the execution of import/export operations, such as the setup of demo data and golden configuration data, and other tasks that are related to data migration. You can also create automated testing of data entities by using task outcome validation.

> **IMPORTANT**
>
> Data task automation isn't currently supported for on-premises environments. The user who executes data task automation must be in the same tenant as the application environment and LCS project.

We recommend the following approach for data task automation.

1. Identify the data-related tasks that will benefit from automation.

   We recommend that implementation teams review their configuration management plan and data migration plan to identify potential data tasks for automation, and also to identify data entity test cases.

2. Define tasks.

   Tasks are defined in an XML manifest. You can keep your manifest under source control as part of configuration management in your application lifecycle management (ALM) strategy.

3. Put the data packages that are related to data task automation in the Shared asset library of Microsoft Dynamics Lifecycle Services (LCS). You can also use a specific LCS project as you require.

   Data task automation manager can consume packages from any sandbox and/or production environment that is related to the LCS project.

   > **IMPORTANT**
   >
   > - The user account that runs Data task automation manager must have access to LCS and to the LCS project that is referenced in the manifest for data packages.
   > - Although data task automation can be run in any environment in the cloud, we strongly recommend that you not run any import/export tasks that use integration application programming interfaces (APIs) in a production environment. Data task automation that involves integration APIs should be used only for automated testing.

4. Run the data tasks, and then review the outcomes.

   Data task automation manager provides the success or failure outcome for each task. It also provides insights into the reason why a task failed.

> **IMPORTANT**
>
> Although data task automation can be run in any environments in the cloud, we recommend that you not run any import/export tasks that use integration APIs in a production environment. Data task automation that involves integration APIs should be used only for automated testing.

The following video is a 55-minute TechTalk that walks you through an early release of Data task automation manager: Task automation framework.

Task manifest

A task must be defined in an XML manifest. This section describes the manifest. For guidance about how to name and design the manifest, see the "Best practices for manifest design" section later in this topic.

Manifest root

The **<TestManifest>** element is the root of the manifest. All other elements are children of this element.

```
<?xml version='1.0' encoding='utf-8'?>
<TestManifest name='Data management demo data set up'>
    <SharedSetup />
        <JobDefinition ID='ImportJobDefinition_1' />
        <EntitySetup ID='Generic' />
    </SharedSetup>
    <TestGroup />
</TestManifest>
```

ELEMENT	ELEMENT CARDINALITY	ATTRIBUTES	ATTRIBUTE DESCRIPTION
<TestManifest>	1..1	name	The *name* helps to identify the purpose of the manifest.

Shared setup

The **Shared setup** section defines general task parameters and behaviors for all tasks in the manifest.

PARENT ELEMENT	ELEMENT	ELEMENT CARDINALITY	ATTRIBUTES	ATTRIBUTE DESCRIPTION
<TestManifest>	<SharedSetup>	1..1	-	This element takes no attributes.

Data files

<DataFile> elements define the data packages and data files that the tasks in the manifest will use. The data files must be either in the LCS asset library of your LCS project or in the Shared asset library.

```
<DataFile ID='SharedSetup' name='Demo data-7.3-100-System and Shared'  assetType='Data package'
lcsProjectId=''/>
<DataFile ID='FinancialsHQUS' name='Demo data-7.3-200-Financials-HQUS' assetType='Data package'
lcsProjectId=''/>
<DataFile ID='FinancialsPICH' name='Demo data-7.3-200-Financials-PICH' assetType='Data package'
lcsProjectId=''/>
<DataFile ID='FinancialsPIFB' name='Demo data-7.3-200-Financials-PIFB' assetType='Data package'
lcsProjectId=''/>
```

PARENT ELEMENT	ELEMENT	ELEMENT CARDINALITY	ATTRIBUTES	ATTRIBUTE DESCRIPTION
<SharedSetup>	<DataFile>	1..n	-	-
	<DataFile>	-	ID	
	<DataFile>	-	name	Name of the asset that represents the data file.
	<DataFile>	-	assetType	The asset type in LCS asset library that stores the data file. This is the asset type name as shown in LCS asset library.
	<DataFile>	-	lcsProjectId	The LCS project that has the data file in its asset library. If the project ID is specified as '' then, it indicates the Shared asset library.

Data project definition

The **<JobDefinition>** element defines the data project definition. There can be more than one job definition in a manifest.

```
<JobDefinition ID='ImportJobDefinition_1'>
    <Operation>Import</Operation>
    <ConfigurationOnly>No</ConfigurationOnly>
    <Truncate></Truncate>
    <Mode>Import async</Mode>
    <BatchFrequencyInMinutes>1</BatchFrequencyInMinutes>
    <NumberOfTimesToRunBatch >2</NumberOfTimesToRunBatch>
    <UploadFrequencyInSeconds>1</UploadFrequencyInSeconds>
    <TotalNumberOfTimesToUploadFile>1</TotalNumberOfTimesToUploadFile>
    <SupportedDataSourceType>Package</SupportedDataSourceType>
    <ProcessMessagesInOrder>No</ProcessMessagesInOrder>
    <PreventUploadWhenZeroRecords>No</PreventUploadWhenZeroRecords>
    <UseCompanyFromMessage>Yes</UseCompanyFromMessage>
    <LegalEntity>DAT</LegalEntity>
    <PackageAPIExecute>true</PackageAPIExecute>
    <PackageAPIOverwrite>false</PackageAPIOverwrite>
    <PackageAPIReexecute>false</PackageAPIReexecute>
    <DefinitionGroupID>TestExport</DefinitionGroupID>
    <PackageName>TestExportPackage</PackageName>
</JobDefinition>
```

PARENT ELEMENT	ELEMENT	ELEMENT CARDINALITY	ATTRIBUTE	DESCRIPTION
<SharedSetup>	<JobDefinition>	1..n	ID	The job definition ID is used in the tasks to reference the definition to be used for the data project.

PARENT ELEMENT	ELEMENT	ELEMENT CARDINALITY	ATTRIBUTE	DESCRIPTION
<JobDefinition>	<Operation>	1..1	-	The operation to be performed is specified by the following values: - Import - Export
	<Truncate>	1..1	-	This is a Boolean field with possible values of Yes or No. This is applicable only when operation is set to *Import*.
	<Mode>	1..1	-	The mode specifies the method using which the operation must be performed. The possible values are: - Import async - Export async - Recurring batch: This uses the enqueue API. Dequeue API is not supported yet. Package API supports both export and import.
	<ConfigurationOnly>	0..1	-	This is a Boolean field with possible values of Yes or No. This must be set to Yes if the task is only to configure the data project but not to perform the specified operation.
	<BatchFrequencyInMinutes>	1..1	-	This specifies the frequency in which the batch must be scheduled. This is applicable only when mode is set to *recurring batch*.
	<NumberOfTimesToRunBatch>	1..1	-	This is used to set a limit to how many times the scheduled batch should run. This is applicable only when mode is set to *recurring batch*.

PARENT ELEMENT	ELEMENT	ELEMENT CARDINALITY	ATTRIBUTE	DESCRIPTION
	<UploadFrequencyInSeconds>	1..1	-	This is used to control the rate at which a file is uploaded to the recurring batch job for import. This must be used only for automated testing of recurring integrations in non-production environments. This is applicable only when mode is set to *recurring batch* and operation is set to *Import*.
	<TotalNumberOfTimesToUpload>	1..1		This controls the total number of times the file should be uploaded to the recurring batch. This must be used only for automated testing of recurring integrations in non-production environments. This is applicable only when mode is set to *recurring batch* and operation is set to *Import*.
	<SupportedDataSourceType>	1..1		This must be used to specify if a file is being sent to the recurring batch or a package. This is only applicable when mode is set to 'recurring batch'.
	<ProcessMessagesInOrder>	1..1		This is a Boolean field with possible values of Yes or No. This is applicable only when mode is set to *recurring batch* and operation is *Import*.
	<PreventUploadWhenZeroRecords>	1..1		This is a Boolean field with possible values of Yes or No. This is applicable only when mode is set to *recurring batch* and operation is *Export*.

PARENT ELEMENT	ELEMENT	ELEMENT CARDINALITY	ATTRIBUTE	DESCRIPTION
	<UseCompanyFrom Message>	1..1		This is a Boolean field which can be set to Yes or No. This is applicable only when mode is set to *recurring batch* and operation is *Import*.
	<LegalEntity>	1..1		This is used to specify the legal entity in which the import/export job must be executed.
	<PackageAPIExecute >	1..1		Refer to package API documentation to understand this parameter. This is a Boolean field which takes 'true' or 'false'.
	<PackageAPIOverwri te>	1..1		Refer to package API documentation to understand this parameter. This is a Boolean field which takes 'true' or 'false'.
	<PackageAPIReexecu te>	1..1		Refer to package API documentation to understand this parameter. This is a Boolean field which takes 'true' or 'false'.
	<DefinitionGroupID>	1..1		Refer to package API documentation to understand this parameter. This is a string field.
	<PackageName>	1..1		Refer to package API documentation to understand this parameter. This is a string field.

Entity setup

The **Entity setup** section defines the characteristics of an entity that a task in the manifest will use. There can be more than one definition, one for each entity that is used by the tasks in the manifest.

```
<EntitySetup ID='Generic'>
    <Entity name='*'>
        <SourceDataFormatName>Package</SourceDataFormatName>
        <ChangeTracking></ChangeTracking>
        <PublishToBYOD></PublishToBYOD>
        <DefaultRefreshType>Full push only</DefaultRefreshType>
        <ExcelWorkSheetName></ExcelWorkSheetName>
        <SelectFields>All fields</SelectFields>
        <SetBasedProcessing></SetBasedProcessing>
        <FailBatchOnErrorForExecutionUnit>No</FailBatchOnErrorForExecutionUnit>
        <FailBatchOnErrorForLevel>No</FailBatchOnErrorForLevel>
        <DisableEntity>No</DisableEntity>
        <SkipStaging>Yes</SkipStaging>
        <ParallelProcessing>
            <Threshold></Threshold>
            <TaskCount></TaskCount>
        </ParallelProcessing>
        <MappingDetail StagingFieldName='RoundingRulePrices' AutoGenerate='Yes' AutoDefault='No'
DefaultValue='' IgnoreBlankValues='No' TextQualifier='No' UseEnumLabel='No'/>
        </Entity>
</EntitySetup>
```

PARENT ELEMENT	ELEMENT	ELEMENT CARDINALITY	ATTRIBUTE	DESCRIPTION
<SharedSetup>	<EntitySetup>	1..n	ID	An identification that will be used by tasks to reference an entity definition to be used.
<EntitySetup>	<Entity>	1..1	name	The entity element is identified by the entity's name. However, to facilitate easy manifest definition, this element also supports * as a wild card which will mean all entities being used in a task. This comes in very handy when using data packages with hundreds of entities in a task.
<Entity>	<SourceDataFormat Name>	1..1	-	This is the file format to be used for the entity.
	<ChangeTracking>	1..1	-	This is a Boolean field with possible values of Yes or No. It enables or disables change tracking on the entire entity.
	<PublishToBYOD>	1..1	-	This is a Boolean field with possible values of Yes or No.

PARENT ELEMENT	ELEMENT	ELEMENT CARDINALITY	ATTRIBUTE	DESCRIPTION
	<DefaultRefreshType>	1..1	-	This sets the default refresh rate on the entity. The possible values are *Incremental push only* or *Full push*.
	<ExcelWorkSheetName>	1..1	-	This is used to specify the worksheet to be used for the entity.
	<SelectFields>	1..1	-	This can be used to specify the fields to be included in the template for an export operation.
	<SetBasedProcessing>	1..1	-	This is a Boolean field with possible values of Yes or No. It is used to enable or disable set based processing on an entity.
	<FailBatchOnErrorForExecutionUnit>	1..1	-	This is a Boolean field with possible values of Yes or No. It is used to enable or disable failure at execution unit level on an entity.
	<FailBatchOnErrorForLevel>	1..1	-	This is a Boolean field with possible values of Yes or No. It is used to enable or disable failure at execution level on an entity.
	<DisableEntity>	1..1	-	This is a Boolean field with possible values of Yes or No. It is used to enable or disable an entity in a data project.
	<SkipStaging>	1..1	-	This is a Boolean field with possible values of Yes or No. It is used to skip staging table for an entity during exports.

PARENT ELEMENT	ELEMENT	ELEMENT CARDINALITY	ATTRIBUTE	DESCRIPTION
	<ParallelProcessing>	1..1	-	This is used to define the parallel processing set up for an entity. The task will delete these settings if already exits at the beginning of the task and it will delete the created settings at the end of its execution.
<ParallelProcessing>	<Threshold>	1..1	-	This specifies the threshold for the parallel processing rule.
	<TaskCount>	1..1	-	This is used to specify the number of parallel tasks to be used for parallel processing.
<Entity>	<MappingDetail>	0..n	-	Allows to configure the *auto generate*, *auto default* and other settings on the mapping for an entity.
	<MappingDetail>	-	StagingFieldName	This attribute is used to identify the entity column for which the settings are to be specified.
	<MappingDetail>	-	AutoGenerate	This is a Boolean field with possible values of Yes or No for enabling/disabling auto generate option.
	<MappingDetail>	-	AutoDefault	This is a Boolean field with possible values of Yes or No for enabling/disabling auto default option.
	<MappingDetail>	-	DefaultValue	This is the default value to be used if auto defaulting is enabled.

PARENT ELEMENT	ELEMENT	ELEMENT CARDINALITY	ATTRIBUTE	DESCRIPTION
	<MappingDetail>	-	IgnoreBlankValues	This is a Boolean field with possible values of Yes or No for enabling/disabling this option.
	<MappingDetail>	-	TextQualifier	This is a Boolean field with possible values of Yes or No for enabling/disabling this option.
	<MappingDetail>	-	UseEnumLabel	This is a Boolean field with possible values of Yes or No for enabling/disabling this option.

Test groups

Test groups can be used to organize related tasks in a manifest. There can be more than one test group in a manifest.

```
<TestGroup name='Set up Financials'>
    <TestCase Title='Import shared set up data package' ID='3933885' RepeatCount='1' TraceParser='off'
TimeOut='20'>
        <DataFile RefID='SharedSetup' />
        <JobDefinition RefID='ImportJobDefinition_1' />
        <EntitySetup RefID='Generic' />
    </TestCase>

    <TestCase Title='Import financials for HQUS' ID='3933886' RepeatCount='1' TraceParser='off'
TimeOut='20'>
        <DataFile RefID='FinancialsHQUS' />
        <JobDefinition RefID='ImportJobDefinition_1'>
            <LegalEntity>HQUS</LegalEntity>
        </JobDefinition>
        <EntitySetup RefID='Generic' />
    </TestCase>

    <TestCase Title='Import financials for PICH' ID='3933887' RepeatCount='1' TraceParser='off'
TimeOut='20'>
        <DataFile RefID='FinancialsPICH' />
        <JobDefinition RefID='ImportJobDefinition_1'>
            <LegalEntity>PICH</LegalEntity>
        </JobDefinition>
        <EntitySetup RefID='Generic' />
    </TestCase>

    <TestCase Title='Import financials for PIFB' ID='3933888' RepeatCount='1' TraceParser='off'
TimeOut='20'>
        <DataFile RefID='FinancialsPIFB' />
        <JobDefinition RefID='ImportJobDefinition_1'>
            <LegalEntity>PIFB</LegalEntity>
        </JobDefinition>
        <EntitySetup RefID='Generic' />
    </TestCase>
</TestGroup>
```

PARENT ELEMENT	ELEMENT	ELEMENT CARDINALITY	ATTRIBUTES	DESCRIPTION
<TestManifest>	<TestGroup>	1..n	-	-
	<TestGroup>	1..1	Name	This is the name for the group to identify its functional reason.
<TestGroup>	<TestCase>	1..n	-	The task is defined in this element. The task can refer to the shared set up to inherit task parameters and task behavior. The task can also override parameters and behavior at its level thus making the management of the manifest simple.
	<TestCase>	-	Title	This is the title for the task.
	<TestCase>	-	ID	This is the ID for the task. This can be alphanumeric with a max character limit of 10.
	<TestCase>	-	RepeatCount	This is a placeholder for a future functionality. However, this must be specified with a value of *1*.
	<TestCase>	-	TraceParser	This is a placeholder for a future functionality. However, this must be specified with a value *off*.
	<TestCase>	-	Timeout	This is the maximum duration a task will be monitored by the task automation manager. If the task is still active beyond the timeout specified, the manager will proceed to the next task in the manifest.

PARENT ELEMENT	ELEMENT	ELEMENT CARDINALITY	ATTRIBUTES	DESCRIPTION
<TestCase>	<DataFile>	1..n	-	This element is used to define the file or data package to be used by the task. This can reference to an already declared file or a data package in the shared section of the manifest. A task can have more than one data file specified for recurring batch import scenarios only. For other scenarios, even if more than one files are specified, the first file is what will be used by the task.
	<JobDefinition>	1..1	-	This element is used to define the data project to be used by the task. This can reference to an already declared job definition in the shared section of the manifest. The task can override elements of job definition to a new value than what is defined in the shared set up.
	<EntitySetup>	1..1	-	This element is used to define the entity set up for entities used by the task. This can reference to an already declared entity set up in the shared section of the manifest. The task can override elements of entity setup to a new value than what is defined in the shared set up.

Best practices for manifest design

You can define a manifest in many ways. Here are a few pointers that you should consider when you design a manifest.

Granularity

We recommend that you determine the granularity of your manifest as a functional decision. Your team must

decide whether it wants to manage many manifests or manage changes in a single manifest.

- Start with as many manifests as your team thinks you logically need. Later, when the team actually starts to run the manifests, it might find that it uses fewer manifests than it expected, and it might want to merge them. In this case, you can merge manifests.
- Consider separation of duties. For example, you might have one manifest for the setup of demo data and another manifest for the setup of the golden configuration for your environment. In this way, you can make sure that team members use only the manifests that they are supposed to use.
- Consider user access to LCS. For example, larger and globally distributed implementation teams might have multiple instances of the application or multiple LCS projects.

Inheritance

The manifest schema supports inheritance of common elements that will apply to all tasks in the manifest. A task can override a common element to define a unique behavior. The purpose of the **Shared setup** section is to minimize repetition of configuration elements, so that elements are reused as much as possible. The goal is to keep the manifest concise and clean, to improve maintenance and readability.

Source control

Manifests that must be used by all the members of an implementation team should be stored in source control in the Application Object Tree (AOT). This approach not only provides the benefits of source control, but also enables a process to distribute or make manifests available to all users in a consistent manner. This approach also enables configuration management for data projects that are related to data management, if manifests are used for configuration.

Number of job definitions and entity definitions

For most of the use cases, one job definition in a manifest should be enough, because inheritance can be used to change the behavior at the task level. This principle also applies to entity definitions.

Validations

Data task automation manager performs validations, based on the setup of a task. If a task fails, you can quickly determine the reason for the failure by viewing the validations after the task is completed. The level of information that Data task automation manager provides is optimized to facilitate initial discovery. For detailed investigation, you must look at the corresponding data project and its execution details.

The following data validations are currently supported:

- **Job status** – This validation checks whether the job was successful.
- **Batch status** – This validation checks whether the batch was successful.
- **Message status** – If the test is about integrations, the message status is validated.
- **Truncation** – If truncation is enabled, this validation checks whether truncation occurred.
- **Skip staging** – If **Skip staging** is enabled on a test, this validation checks whether staging was skipped.

Example 1: Configuration management for data projects

The **<ConfigurationOnly>** element can be used to create configuration tasks for data projects. When ConfigurationOnly is set to Yes, the data projects are only created but not executed. This allows for managing data projects across environments in an automated manner.

```
<?xml version='1.0' encoding='utf-8'?>
<TestManifest name='Data management demo data set up'>
    <SharedSetup>
        <DataFile ID='SharedSetup' name='Demo data-7.3-100-System and Shared'  assetType='Data package'
lcsProjectId=''/>
        <DataFile ID='FinancialsHQUS' name='Demo data-7.3-200-Financials-HQUS' assetType='Data package'
```

```
lcsProjectId=''/>
        <DataFile ID='FinancialsPICH' name='Demo data-7.3-200-Financials-PICH' assetType='Data package'
lcsProjectId=''/>
        <DataFile ID='FinancialsPIFB' name='Demo data-7.3-200-Financials-PIFB' assetType='Data package'
lcsProjectId=''/>

        <JobDefinition ID='ImportJobDefinition_1'>
            <ConfigurationOnly>Yes</ConfigurationOnly>
            <Operation>Import</Operation>
            <Truncate>No</Truncate>
            <Mode>Import async</Mode>
            <BatchFrequencyInMinutes>1</BatchFrequencyInMinutes>
            <NumberOfTimesToRunBatch >2</NumberOfTimesToRunBatch>
            <UploadFrequencyInSeconds>1</UploadFrequencyInSeconds>
            <TotalNumberOfTimesToUploadFile>1</TotalNumberOfTimesToUploadFile>
            <SupportedDataSourceType>Package</SupportedDataSourceType>
            <ProcessMessagesInOrder>No</ProcessMessagesInOrder>
            <PreventUploadWhenZeroRecords>No</PreventUploadWhenZeroRecords>
            <UseCompanyFromMessage>Yes</UseCompanyFromMessage>
            <LegalEntity>DAT</LegalEntity>
        </JobDefinition>

        <EntitySetup ID='Generic'>
            <Entity name='*'>
                <SourceDataFormatName>Package</SourceDataFormatName>
                <ChangeTracking>No</ChangeTracking>
                <PublishToBYOD>No</PublishToBYOD>
                <DefaultRefreshType>Full push only</DefaultRefreshType>
                <ExcelWorkSheetName></ExcelWorkSheetName>
                <SelectFields>All fields</SelectFields>
                <SetBasedProcessing>No</SetBasedProcessing>
                <FailBatchOnErrorForExecutionUnit>No</FailBatchOnErrorForExecutionUnit>
                <FailBatchOnErrorForLevel>No</FailBatchOnErrorForLevel>
                <FailBatchOnErrorForSequence>No</FailBatchOnErrorForSequence>
                <ParallelProcessing>
                    <Threshold></Threshold>
                    <TaskCount></TaskCount>
                </ParallelProcessing>
            </Entity>
        </EntitySetup>
    </SharedSetup>

    <TestGroup name='Set up import jobs for Financials'>
        <TestCase Title='Set up import job for shared set up data package' ID='3933885' RepeatCount='1'
TraceParser='off' TimeOut='20'>
            <DataFile RefID='SharedSetup' />
            <JobDefinition RefID='ImportJobDefinition_1' />
            <EntitySetup RefID='Generic' />
        </TestCase>

        <TestCase Title='Set up import job for financials HQUS' ID='3933886' RepeatCount='1'
TraceParser='off' TimeOut='20'>
            <DataFile RefID='FinancialsHQUS' />
                <JobDefinition RefID='ImportJobDefinition_1'>
                    <LegalEntity>HQUS</LegalEntity>
                </JobDefinition>
                <EntitySetup RefID='Generic' />
        </TestCase>

        <TestCase Title='Set up import job for financials PICH' ID='3933887' RepeatCount='1'
TraceParser='off' TimeOut='20'>
            <DataFile RefID='FinancialsPICH' />
                <JobDefinition RefID='ImportJobDefinition_1'>
                    <LegalEntity>PICH</LegalEntity>
                </JobDefinition>
                <EntitySetup RefID='Generic' />
        </TestCase>

        <TestCase Title='Set up import job for financials PIFB' ID='3933888' RepeatCount='1'
```

```
TraceParser='off' TimeOut='20'>
            <DataFile RefID='FinancialsPIFB' />
                <JobDefinition RefID='ImportJobDefinition_1'>
                    <LegalEntity>PIFB</LegalEntity>
                </JobDefinition>
                <EntitySetup RefID='Generic' />
        </TestCase>
    </TestGroup>
</TestManifest>
```

Example 2: Automated setup of demo data

The following manifest shows the setup of demo data for three legal entities when the demo data packages are stored in the Shared asset library. The difference in this example from the previous example is the actual execution of the data projects to set up the demo data. This is accomplished by not using the ConfigurationOnly option or setting it to No to use it for consistency of the manifest.

```
<?xml version='1.0' encoding='utf-8'?>
<TestManifest name='Data management demo data set up'>
    <SharedSetup>
        <DataFile ID='SharedSetup' name='Demo data-7.3-100-System and Shared'  assetType='Data package'
lcsProjectId=''/>
        <DataFile ID='FinancialsHQUS' name='Demo data-7.3-200-Financials-HQUS' assetType='Data package'
lcsProjectId=''/>
        <DataFile ID='FinancialsPICH' name='Demo data-7.3-200-Financials-PICH' assetType='Data package'
lcsProjectId=''/>
        <DataFile ID='FinancialsPIFB' name='Demo data-7.3-200-Financials-PIFB' assetType='Data package'
lcsProjectId=''/>

        <JobDefinition ID='ImportJobDefinition_1'>
                <Operation>Import</Operation>
                <Truncate></Truncate>
                <Mode>Import async</Mode>
                <BatchFrequencyInMinutes>1</BatchFrequencyInMinutes>
                <NumberOfTimesToRunBatch >2</NumberOfTimesToRunBatch>
                <UploadFrequencyInSeconds>1</UploadFrequencyInSeconds>
                <TotalNumberOfTimesToUploadFile>1</TotalNumberOfTimesToUploadFile>
                <SupportedDataSourceType>Package</SupportedDataSourceType>
                <ProcessMessagesInOrder>No</ProcessMessagesInOrder>
                <PreventUploadWhenZeroRecords>No</PreventUploadWhenZeroRecords>
                <UseCompanyFromMessage>Yes</UseCompanyFromMessage>
                <LegalEntity>DAT</LegalEntity>
        </JobDefinition>

        <EntitySetup ID='Generic'>
            <Entity name='*'>
                <SourceDataFormatName>Package</SourceDataFormatName>
                <ChangeTracking></ChangeTracking>
                <PublishToBYOD></PublishToBYOD>
                <DefaultRefreshType>Full push only</DefaultRefreshType>
                <ExcelWorkSheetName></ExcelWorkSheetName>
                <SelectFields>All fields</SelectFields>
                <SetBasedProcessing></SetBasedProcessing>
                <FailBatchOnErrorForExecutionUnit>No</FailBatchOnErrorForExecutionUnit>
                <FailBatchOnErrorForLevel>No</FailBatchOnErrorForLevel>
                <FailBatchOnErrorForSequence>No</FailBatchOnErrorForSequence>
                <ParallelProcessing>
                    <Threshold></Threshold>
                    <TaskCount></TaskCount>
                </ParallelProcessing>
            </Entity>
        </EntitySetup>
    </SharedSetup>

    <TestGroup name='Set up Financials'>
```

```
        <TestCase Title='Import shared set up data package' ID='3933885' RepeatCount='1' TraceParser='off'
TimeOut='20'>
            <DataFile RefID='SharedSetup' />
            <JobDefinition RefID='ImportJobDefinition_1' />
            <EntitySetup RefID='Generic' />
        </TestCase>

        <TestCase Title='Import financials for HQUS' ID='3933886' RepeatCount='1' TraceParser='off'
TimeOut='20'>
            <DataFile RefID='FinancialsHQUS' />
            <JobDefinition RefID='ImportJobDefinition_1'>
                <LegalEntity>HQUS</LegalEntity>
            </JobDefinition>
            <EntitySetup RefID='Generic' />
        </TestCase>

        <TestCase Title='Import financials for PICH' ID='3933887' RepeatCount='1' TraceParser='off'
TimeOut='20'>
            <DataFile RefID='FinancialsPICH' />
            <JobDefinition RefID='ImportJobDefinition_1'>
                <LegalEntity>PICH</LegalEntity>
            </JobDefinition>
            <EntitySetup RefID='Generic' />
        </TestCase>

        <TestCase Title='Import financials for PIFB' ID='3933888' RepeatCount='1' TraceParser='off'
TimeOut='20'>
            <DataFile RefID='FinancialsPIFB' />
            <JobDefinition RefID='ImportJobDefinition_1'>
                <LegalEntity>PIFB</LegalEntity>
            </JobDefinition>
            <EntitySetup RefID='Generic' />
        </TestCase>
    </TestGroup>
</TestManifest>
```

Data validation checklist workspace

2/18/2021 • 2 minutes to read • Edit Online

This topic provides an overview of the **Data validation checklist workspace** and the associated configuration.

The **Data validation checklist** workspace lets you track data validation processes across companies, areas, and people. The checklist can be used during a new implementation, after an upgrade, or after a migration. Depending on your view of the **Data validation checklist** workspace, you'll see either all tasks and statuses for a data validation project, or just the tasks that are assigned to you.

You must first select a data validation project at the top of the workspace. All data that is shown in the workspace is then filtered by the selected data validation project.

Summary tiles

The **Summary** tiles provide an overview of the process, and indicators help you keep the data validation process on track. You can see all remaining tasks, completed tasks, in progress tasks, and not started tasks for the process. This information is for all companies that are included in the selected data validation project.

Tasks and status section

In the **Tasks and status** section, the status of the overall data validation project is displayed in various ways: status by legal entity, by area, and by task list. You can select the filter to view the status for a specific company. Each status tab provides a breakdown by both the percentage that has been completed and the number of tasks that remain.

The last tab is for the detailed task list. This list shows the full task list. You can filter the task list in several ways. Click **Edit task** to change the status of a task or assign a task. Click **Attachments** to view attachments for a task.

The task name is a hyperlink to the page where the user must go to complete the work. You can set this hyperlink by using the **Menu item name** field when you edit or create a task from the **Configure data validation project** form.

You can attach files, notes, images, and URLs to a task by using the **Attachments** action. For example, you can attach a report file that was printed for a task. An icon appears in the **Attachment** column for the task if an attachment is present.

The **Completed by** option will be automatically filled after the task is completed with the name of the worker who completed the task. When a task is marked as completed, the **Completed date** field is automatically updated to the current date and time.

Configure data validation project page

Before you can use the **Data validation checklist** workspace, you must configure the process by using the **Configure data validation project** page. (Click **Workspaces** > **Data validation checklist** > **Configure data validation project**.)

Task areas

You use task areas to group data validation tasks into logical areas of ownership within your organization. For

example, Accounts payable, Accounts receivable, or General ledger might be used as task areas.

The **Menu item name** is associated with the task work effort and can be used to go directly to the associated page from the task link in the workspace. For example, a data validation task to run the **Accounts payable aging** report for Accounts payable can be linked to the **Accounts payable aging report** page.

Data management error descriptions

2/18/2021 • 3 minutes to read • Edit Online

This topic documents the scenarios when a specific error will be seen. These scenarios aren't a complete list of errors and scenarios, however this list will be continuously updated so keep checking back for updates. Any feedback on this page with regards to specific errors that should be covered are welcome.

This topic describes the error messages that you might encounter in data management.

Import to target failed due to an update conflict as more than one process is trying to update the same record at the same time

When you use recurring imports (enqueue API), if the files are sent to the end point at high frequency and the sequential processing of messages isn't enabled, data management will try to process the files in parallel. When files are processed in parallel, and multiple files have the same record, multiple threads will try to update the same record at the same time. If this is a data issue, you must update the data so that the same records don't repeat across files. If this is not a data issue and the entity is expected to handle such cases, this might be a bug. For bugs, to mitigate the issue, you can choose to sequentially process the files. If this is not a data issue and the entity is not expected to process in parallel, then this entity must not be subjected to parallel processing. You should enable sequential processing of messages in the recurring job.

There are field(s) which are not mapped to Entity <EntityName>

It is a common practice to use the export functionality to generate the entity template file that can be later used for imports. However, while exporting the template, in fixed width format with 'First row header' set to 'No' (in source data formats set up), the exported template will not have the column names. When this file is imported, it will result in this error.

Data package download - Error downloading data package for job ''. Record for ID - {GUID} not found

One of the scenarios where this error can happen is when the environment, such as the dev environment, points to the database in another environment, such as UAT, and the export job is run from the source environment. which is dev in this example. The exported file gets uploaded to the blob storage that is associated with the source environment (dev, in this example). However, this job will also show up in the target environment (UAT) since the database is shared. If you try to download the exported file using the **Download file** option, this error will display because the file does not exist in the blob storage of the target environment (UAT) from where you are trying to download.

XML is not in correct format-Exception from HRESULT: 0xC0010009

This message covers all XML formatting issues in the file. For example, the data project has mappings for columns that do not exist in the file that is being used for the operation. This error can happen if certain columns were removed from the file and this file is now used. Either fix the mapping in the data project or fix the file to have all the columns as expected.

Error while uploading a file during export

When running an export on a development environment, an error could occur relating to not being able to

upload the export file. This could occur if Azure Storage Emulator is not available or an older version of the emulator is installed. To resolve this issue, install the latest emulator, restart the virtual machine (VM), and rerun the export job. The storage emulator can be installed from Azure Storage Emulator.

Azure Data Lake overview

2/18/2021 • 13 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

What is Azure Data Lake?

Microsoft Azure Data Lake is a technology in Azure cloud that enables big data analytics and artificial intelligence (AI). When this topic mentions "Data Lake," it's referring specifically to storage technology that is based on Azure Data Lake Storage Gen2.

Data lakes provide cloud storage that is less expensive than the cloud storage that relational databases provide. Therefore, large amounts of data can be stored in the cloud. This data includes both business data that is traditionally stored in business systems and data warehouses, device and sensor data, such as signals from devices. In addition, Data Lake supports a range of tools and programming languages that enable large amounts of data to be reported on, queried, and transformed.

For an overview of Data Lake Storage Gen2, see Introduction to Azure Data Lake Storage Gen2.

Dynamics 365 products, such as Finance and Operations apps, use Data Lake for AI and analytics scenarios. Therefore, customers can take advantage of the strengths and cost advances that this technology offers. The following sections provide an overview of the scenarios.

Analytical workspaces

Analytical workspaces provide contextual and actionable insights in Finance and Operations apps. They give users a bird's-eye view of a business process, so that they can immediately get relevant information and take appropriate action.

Analytical workspaces are based on Entity store and use embedded Power BI technology to provide rich, interactive visuals of data from Finance and Operations apps. Analytical workspaces are fun and exciting to use: they invite your users to explore data.

Analytical workspaces can be used for operational analytics scenarios in two ways:

- Use and extend the ready-made analytical workspaces, so that you don't have to build workspaces from scratch.
- Build your own Power BI–based analytical reports.

For more information, see Embedded Power BI in workspaces.

BYOD

Bring your own database (BYOD) is a service that lets customers extract data from Finance and Operations apps

into their own data warehouses. We recommend that you use BYOD when you must combine data from Finance and Operations apps with other systems or with reporting that uses earlier data.

For more information, see Bring your own database (BYOD).

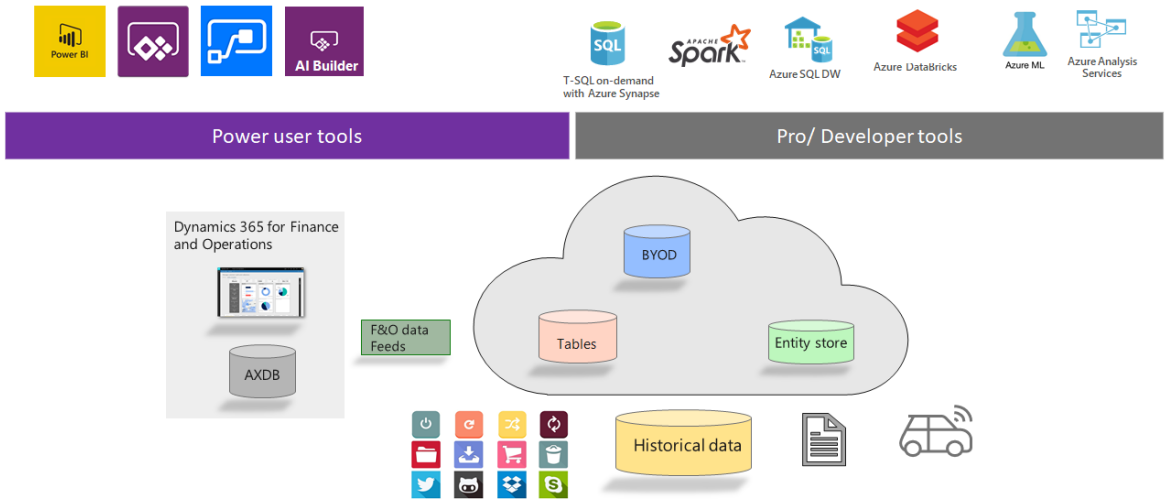
Data Lake combines BYOD and Entity store

Customers use a combination of analytical workspaces (which are based on Entity store) and BYOD for different scenarios. Following table compares the scenarios and capabilities

ANALYTICAL WORKSPACES BUILT-ON ENTITY STORE	DATA WAREHOUSE WITH BRING YOUR OWN DB (BYOD)
Near real-time Operational Analytics (with embedded Power BI)	Data warehouse with PowerBI.com or other tools
Reporting on data from F&O	Mash-up F&O data with other data sources
Ready-made reports shipped with F&O or ISV solution	Reports extended or authored by partner

Reports authored with both these sources can be pinned into Analytical workspaces in F&O with contextual security and drill thru actions Data Lake combines both these services into a single service that offers the "best of both worlds":

- Because Data Lake is included in customer subscriptions, you can bring your own data lake and integrate it with Finance and Operations apps. Finance and Operations apps will use your data lake to store Entity store data and operate analytical workspaces. Analytical workspaces continue to work as they worked before.
- Entity store is staged in your data lake and provides a set of simplified (denormalized) data structures to make reporting easier. Your users can now be given direct access to the data that is most relevant to them, and they can create their own reports by using a tool of their choice.
- Instead of exporting data by using BYOD, customers can select the data that is staged in the data lake. Data feed service, which is part of Finance and Operations services, keeps the data in the data lake fresh.
- You can bring your own data into the data lake to supplement the data that Finance and Operations apps provide. This capability allows for easy data mash-up scenarios in the data lake.
    - Data from external sources can easily be ingested into the data lake via hundreds of ready-made connectors that are available in tools such as Power BI dataflows and Azure Data Factory.
    - Historical data and earlier data that is often inherited as a part of the transition to Finance and Operations apps can be ingested directly into the data lake.
    - Data lakes provide options for ingesting non-business data. For example, device data can easily be ingested into the data lake.
- Cloud-based services let both power users and developers consume this data.

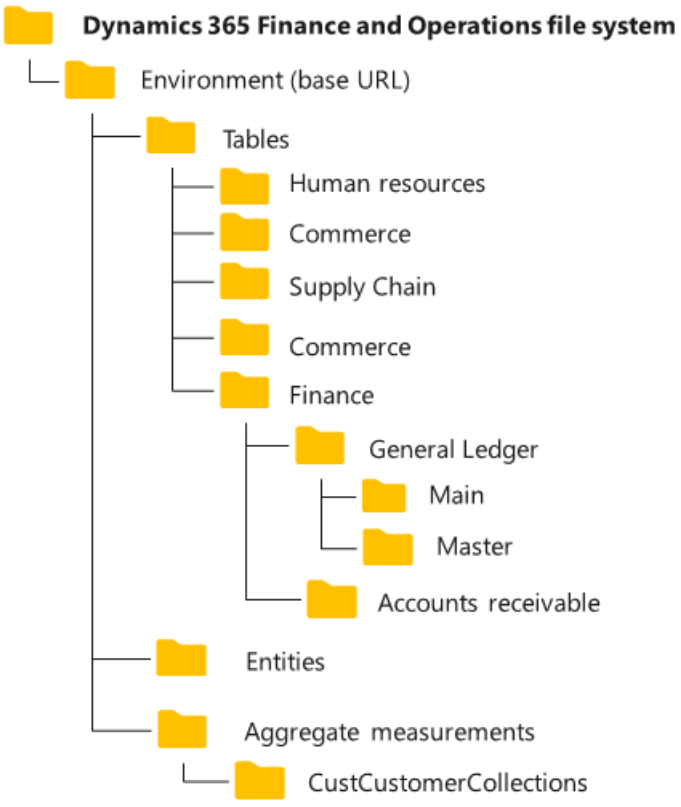


Common Data Model folders

Data is stored in Data Lake to comply with the Common Data Model folder standard. Here are some results:

- Data that Finance and Operations apps stage in Data Lake is organized into a set of folders.
- Common Data Model folders contain metadata definitions in addition to data files. Metadata definitions are kept in model files, according to the standard that is specified by the Common Data Model language.
- Because metadata is present and data storage complies with the Common Data Model folder standard, Azure and other services can read and transform the data.

The following illustration shows the Common Data Model folder structure from Finance and Operations apps.



For more information about Common Data Model in Data Lake, see Use the Common Data Model to optimize Azure Data Lake Storage Gen2.

Here is an example:

- You can attach a Common Data Model folder to Power BI dataflows as a reference dataflow. You can work

with Power BI dataflows and further reshape the data, or you can create Power BI datasets and reports.
- You can use Data Factory or other data transformation tools to further shape the data.

Like Finance and Operations apps, other services (including Dataverse), Azure IoT, and many third-party tools and service can understand and work with data in Common Data Model folders. The list of services is growing. Here are some examples:

- Dataverse lets you export data to your own data lake. For more information, see Exporting Dataverse data to Azure Data Lake is Generally Available.
- Power users can transform data in Data Lake by using Dataverse dataflows. For more information, see Use the Common Data Model to optimize Azure Data Lake Storage Gen2.

How you can use Data Lake later if you're currently using BYOD

In most cases, customers use BYOD to extract data from Finance and Operations apps so that they can use that data for reporting or analytics. BYOD requires that customers provision and maintain an Azure SQL database to store data that is exported from Finance and Operations apps.

Some customers use the exported data in BYOD for reporting. These customers can just point reporting tools to the SQL database and create reports.

Some customers use BYOD as a staging area, where a "snapshot" of the Finance and Operations data is retained. These customers have an enterprise data warehouse, and they use additional data pipelines that copy the data from the BYOD "staging area" to their data warehouse. They might also have other downstream processing and transformation pipelines.

If you're currently using BYOD for these scenarios, you will gain several benefits by onboarding to Data Lake.

Because data is already present, export isn't required

Data Lake integration lets users select tables and entities, just as they can in the BYOD experience. After tables and entities are selected, the system updates the data in Data Lake. The system also continuously exports data as it changes in Finance and Operations apps. The data lake reflects the updated Finance and Operations data within a few minutes after the changes occur.

> **NOTE**
> Table data is updated within minutes after a change occurs in Finance and Operations apps. According to the current service-level agreement (SLA) that the services offer, data is updated within **10 minutes**.

Because of Data Lake integration, customers don't have to monitor and manage complex data export and orchestration schedules. No user intervention is required to update data in the data lake.

Reduced cost of data storage

Data is stored in a data lake (Gen2) instead of the SQL database that BYOD requires. Therefore, customer can use a storage medium that is much less expensive than Azure SQL Database.

> **NOTE**
>
> Because Data Lake Storage Gen2 is included in a customer's subscription, the customer must pay for data storage and input/output (I/O) costs that are incurred when data is read and written to the data lake. The customer might also incur I/O costs because Finance and Operations apps write data to the data lake or update the data in it. To help reduce intra-region I/O costs, Finance and Operations apps require that data lakes be provisioned in the same country or region as the Finance and Operations environment.

For more information about cost, see the Azure Data Lake Storage Gen2 pricing page.

Existing downstream/consumption pipelines can be preserved

As was discussed earlier, BYOD is mostly used in two scenarios:

- Reporting tools and other tools access BYOD directly.
- BYOD is used as a temporary staging area to store data while it's being exported to other downstream systems, such as data warehouses.

For the first scenario, you can point reporting tools to the SQL database. Many reporting tools work with SQL databases, because they can use Transact-SQL (T-SQL) to read data.

For the second scenario, you can use data integration/transformation tools such as Data Factory. Many data integration tools can consume data directly from the data lake.

For both scenarios, if you're using T-SQL to read the database, you can create a SQL Server endpoint by using Azure Synapse Analytics. Azure Synapse include SQL-on-demand capability that enables Data Lake to be queried by using the T-SQL language. Downstream tools don't have to be modified, because you can preserve the data shape, as you can when you use BYOD.

Simplified data pipeline for near-real-time reporting

In a traditional data warehouse, data is stored in a staging area before it can be aggregated and simplified for reporting. You might also have reporting tools that aggregate data for better user experiences. If you have multiple data stops (for staging, denormalization, and aggregation), data staleness increases (that is, after activity occurs, more time passes before reports reflect the results).

Traditional data warehouses that have multiple data stops are ideal for reporting on data that changes every day or even several times a day. However, if you must report on near-real-time data (for example, if you must report on retail sales data within minutes after the sales occur), you might have to design a data pipeline that has minimal data stops. This pattern is known as *hot-path, cold-path reporting* (or *lambda architecture*). Multiple data stops and multiple data pipelines increase complexity and management efforts.

Data Lake integration makes *warm-path reporting* available as the default reporting option. Because data in a data lake is updated within minutes, this approach might be acceptable for most reporting scenarios, even scenarios that involve near-real-time reporting.

For near-real time reporting, you can minimize the data staging and preparation steps by using on-demand query (for example, Azure Synapse) and Power BI direct query mode, which queries semi-prepared data in the data lake. For analytical reporting, you can denormalize and aggregate the data in the data lake.

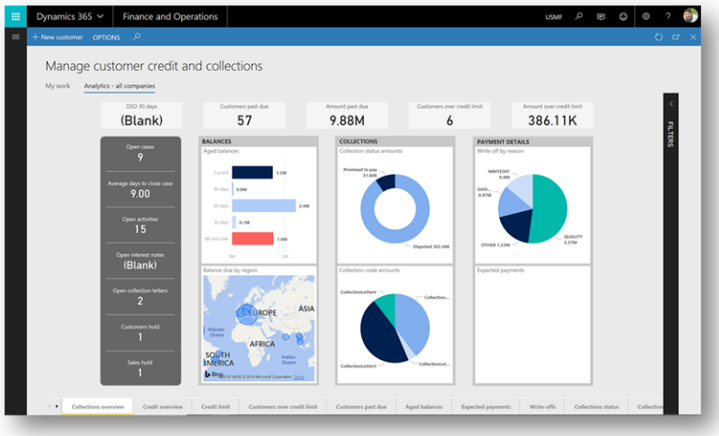
How you can use Data Lake later if you're currently using analytical workspaces

Analytical workspaces make in-context operational analytics available in Finance and Operations apps. Analytical workspaces are built as an extension to Finance and Operations workspaces. They are a type of a "cockpit" that

provides a bird's-eye view of business processes. There are more than 25 ready-made analytical workspaces that provide interactive, near-real-time data exploration.

You can drive action by adding contextual gestures to analytical workspaces. Therefore, users can act on results without having to leave the report that they are viewing. An extensive programming model that is built into Finance and Operations apps enables enterprise resource planning (ERP) actions and business logic to be driven from analytical workspaces.

You can use the ready-made analytical workspaces from Microsoft, partners, and independent software vendors (ISVs) as starting points. Alternatively, if you've built your own Power BI–based analytical reports by using BYOD or your own data warehouse, you can pin them to analytical workspaces. In both cases, you can enrich business processes through in-context reporting that drives action.



- **Real-time insights** into business processes
- **Interactive visuals** - explore lots of data in a fun and exciting way
- jump into details right here and **take action**
- **In context, embedded** - you don't need to go anywhere else
- Power users can modify or create new pages with **Web edit**
- **Pin your own** reports into Analytical workspaces

The ready-made analytical workspaces that are part of core Finance and Operations apps (or part of ISV extensions) include reports that are built by using Entity store. Entity store contains *aggregate measurements* (that is, simplified data structures, such as fact tables and dimensions).

Aggregate measurements will be available in the data lake instead of Entity store. If you're using ready-made analytical workspaces that were built by using Entity store, they will be pointed to Data Lake in an upcoming monthly service update. The Microsoft service team will notify you when your Entity store is ready for the transition to Data Lake. You won't have to modify the reports themselves. Therefore, no development work will be required.

If you've pinned your own Power BI reports to analytical workspaces, you can transition them to Data Lake on your own schedule.

How you can modernize your existing data warehouse by using Data Lake

Although you can use Data Lake integration to transition from BYOD and gain immediate benefits, Data Lake also lets you do much more.

Data Lake is designed to store large amounts of data: hundreds of terabytes (TB) or more. It takes advantage of Azure Blob storage, which is inexpensive storage technology that many underlying services of Azure use.

Data lakes are designed for big data analytics. You can bring your historical data and earlier data from your own systems into Data Lake. This data can consist of scanned documents in addition to business data. You can then apply machine learning models to the documents to make sense of the content and the opinions that the document authors expressed. You can also collect signals from devices and vehicles or machines on the shop floor, and store the data in data lakes. You can then apply machine learning models to detect anomalies and patterns in the signals, and you can join the results with business data to take proactive action.

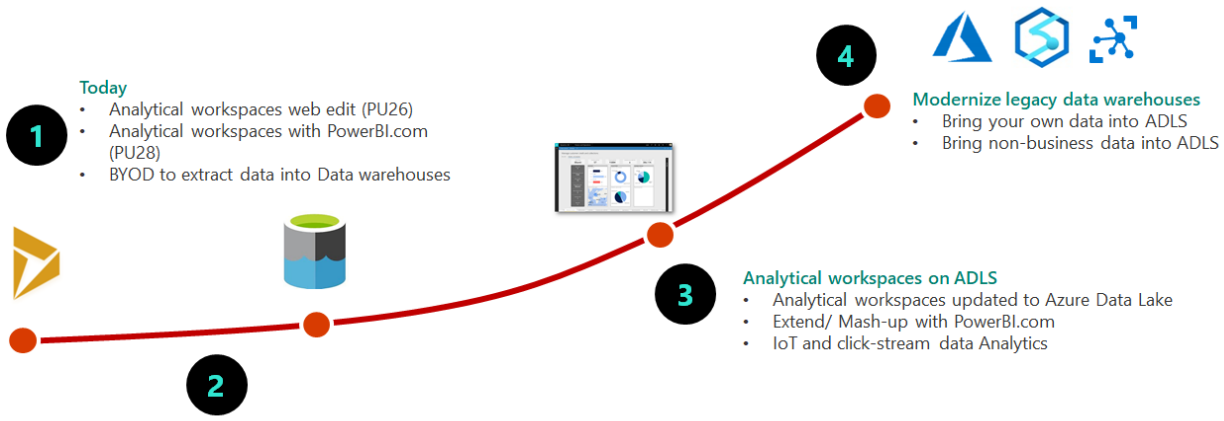
Data Lake has many associated services that enable analytics, data transformation, and the application of AI and

machine learning. Azure services from Microsoft, partners, and open-source tools can be used to reason over the data.

Instead of downloading Finance and Operations data from a data lake into your on-premises data warehouses, you can bring your on-premises data into a data lake. Microsoft refers to this transformation as *modernization of data warehouses*.

Planning the transition

You can plan your transition to Data Lake in multiple stages, as shown in the following illustration. Each stage offers business benefits that can be justified on their own. You can use the stages that are shown here as a planning guideline.



1. **Your current situation:** You might already be using BYOD and analytical workspaces that are based on Entity store.
2. **Getting easy access to data:** As you gain access to tables, entities, and aggregate measurements in Data Lake, you will be able to retire BYOD and use the data that is readily available. Therefore, management effort and costs can be reduced, as was discussed earlier. You can keep your existing downstream data warehouses and pipelines to manage project scope and budget.
3. **Empowering power users:** Analytical workspaces will be transitioned to Data Lake as a service update. Therefore, ready-made analytical workspaces are based on the same data that is available in the data lake. Power users can easily extend analytical workspaces. When the service update occurs, you will be able to make the full capabilities of PowerBI.com available to your power users. By using capabilities such as Power BI dataflows, power users can easily combine data from online services and data is already available in the data lake. The same reports that are available in analytical workspaces can be consumed directly on PowerBI.com.
4. **Modernizing your previous data warehouse:** Modernization will probably be the investment that brings the most benefits. You can move data that currently exists in your on-premises data warehouse to the cloud. You can rely on cloud-based computing services and apply the same transformations on a pay-per-use basis. You can also combine your business data with sensor and device data.

Configure export to Azure Data Lake

2/18/2021 • 10 minutes to read • Edit Online

NOTE

The **Export to Azure Data Lake** feature is in limited preview and may not be available in all regions and environments supported by Finance and Operations apps. If you are unable to find the **Export to Azure Data Lake** functionality in Lifecycle Services (LCS) or your Finance and Operations apps, this feature is not currently available in your environment.

Currently, previews are closed. In the coming months we will enable additional environments in several regions. We are accepting requests from customers who would like to join the preview. If you would like to join a future preview, complete the survey. We will contact you when we are ready to include you. You can also join a Yammer group by completing the survey. You can use the Yammer group to stay in contact and ask questions that will help you understand the feature.

Until the feature is enabled in your environment, you have the option to prototype/plan the feature implementation using GitHub tools. The tools will enable you to export data from your sandbox environment into a storage account in the same format as exported by the feature.

At this time, **Export to Azure Data Lake** feature is not available in Tier-1 (developer) environments. You need a cloud-based Tier-2 or higher environment to enable this feature.

To make aggregate measurements available in a data lake, continue to use the feature in the manner that is described in Make entity store available as a Data Lake.

Create Service Principle for Microsoft Dynamics ERP Microservices

The **Export to Azure Data Lake** feature is built using a microservice that exports Finance and Operations app data to Azure Data Lake and keeps the data fresh. Microservice uses the Azure service principle, **Microsoft Dynamics ERP Microservices**, to securely connect to your Azure resources. Before you configure the Export to Data Lake feature, add the **Microsoft Dynamics ERP Microservices** service principle to your Azure Active Directory (Azure AD). This step enables Azure AD to authenticate the microservice.

NOTE

You will need **Azure Active Directory tenant administrator** rights to perform these steps.

To add the service principle, complete the following steps.

1. Launch the Azure portal and go to the Azure Active Directory.
2. On the left menu, select **Manage** > **Enterprise Applications**, and search for the following applications.

APPLICATION	APP ID
Microsoft Dynamics ERP Microservices	0cdb527f-a8d1-4bf8-9436-b352c68682b2

If you are unable to find the above applications, complete following steps.

3. On your local machine, open the **Start** menu, and search for **PowerShell**.
4. Right-click **Windows PowerShell**, and then select **Run as administrator**.
5. Run the following command to install **AzureAD** module:

```
Install-Module -Name AzureAD
```

- If NuGet provider is required to continue, select **Y** to install it.
- If an **Untrusted repository** message appears, select **Y** to continue.

6. Run the following commands to add the application to the Azure Active Directory.

```
Connect-AzureAD
```

```
New-AzureADServicePrincipal –AppId '0cdb527f-a8d1-4bf8-9436-b352c68682b2'
```

7. Sign in as the Azure Active Directory administrator when prompted.

Configure Azure Resources

To configure the export to Data Lake, create a storage account in your own Azure subscription. This storage account is used to store data. Next, create an Azure AD application ID that grants access to the root of your storage account. Your Finance or Operations app will use the Azure AD application to gain access to storage, create the folder structure, and write data. Create a key vault in your subscription and store the name of the storage account, application ID, and the application secrets. If you don't have permission to create resources in Azure portal, you will need assistance from someone in your organization with the required permissions.

The steps, which take place in the Azure portal, are as follows:

> **NOTE**
> When you are working in the Azure portal, you will be instructed to save several values for subsequent steps. You will also provide some of these values to your Finance and Operations apps by using Lifecycle Services (LCS). You will need Administrator access to LCS in order to do this.

1. Create an application in Azure Active Directory
2. Grant access control roles to applications
3. Create a Data Lake Storage (Gen2 account) in your subscription
4. Create a key vault
5. Add secrets to the key vault
6. Authorize the application to read secrets in the key vault
7. Install the Export to Data Lake add-in in LCS

Create an application in Azure Active Directory

1. In the Azure portal, select **Azure Active Directory**, and then select **App registrations**.
2. Select **New registration**, and enter the following information:
   - **Name:** Enter a name for the app.
   - **Supported Account types**: Choose the appropriate option.
3. After the application is created, select it and then copy and save the Application (client) ID at the top of the page. You will need this later.
4. On the left navigation pane, select **API permissions**.
5. Select **Add a permission**, and in the **Request API permissions** dialog box, select **Azure Key vault**.
6. Select **Delegated permissions**, select **user_impersonation**, and then select **Add permissions**.

7. On the left navigation pane, select **Certificates & secrets**, and then select **New client secret**.
8. In the **Description** field, enter a name.
9. In the **Expires** field, select an option, and then select **Add**. The system will generate a secret and display it under the grid.
10. Copy the secret **Value** to the clipboard. This is the value you will need to provide when you set up the key vault later.

Create a Data Lake Storage (Gen2) account in your subscription

The Data Lake Storage account will be used to store data from your Finance and Operations apps. To manually create a storage account, you must have administrative rights to your organization's Azure subscription. To create a storage account, complete the following steps.

1. In the Azure portal, select **Create new resource**, and then search for and select **Storage account – blob, file, table, queue**.
2. In the **Create storage account** dialog box, provide values for the following parameter fields:
   - **Location:** Select the data center where your environment is located. If the data center that you select is in a different Azure region, you may incur additional data movement costs. If your Microsoft Power BI or your data warehouse is in a different region, you can use replication to move storage between regions.
   - **Performance**: We recommend you select **Standard**.
   - **Account kind**: You must select **StorageV2**. In the **Advanced options** dialog box, you will see the option, **Data Lake storage Gen2**.
3. On the **Advanced tab**, select **Data Lake storage Gen2** > **Hierarchical namespaces**, and then select **Enabled**. If you disable this option, you may not be able to consume data that is written by Finance and Operations apps with services such as Power BI data flows and AI builder.
4. Select **Review and create**. When the deployment is complete, the new resource will be shown in the Azure portal.
5. In the Azure portal, select the storage account you created. Copy and save the storage account name.

Grant access control roles to applications

You need to grant your application permissions to read and write to the storage account. These permissions are granted by using Roles in Azure AD.

1. In **Azure portal**, open the storage account that you created earlier.
2. Select **Access Control (IAM)** in the left navigation.
3. On the **Access control** page, select the **Role assignments** tab.
4. Select **Add** at the top of the page, and then select **Add role assignment**.
5. In the **Add role assignment** dialog, select the **Role** field, and then select **Owner**.

> **NOTE**
> Don't make any changes to the fields, **Assign access to** and **Azure AD user, group, or service principal**.

6. In the **Select** field, select the application that you registered earlier.
7. Select **Save**.
8. Add the remaining roles shown in the following table by repeating steps 4-7.

APPLICATION TO BE SELECTED	ROLE TO BE ASSIGNED
The application you created earlier	Storage blob data contributor
The application you created earlier	Storage blob data reader

Create a key vault

A key vault is a secure way to share details such as storage account name to your Finance and Operations apps. Complete the following steps to create a key vault and a secret.

1. In the Azure portal, select **Create a new resource** and then search for and select, **Key Vault**.
2. In the **Create key vault** dialog box, in the **Location** field, select the datacenter where your environment is located.
3. After the key vault is created, select it from the list, and on the left navigation pane, select **Overview**.
4. Save the value in the **DNS name** field. You will need this value later.

Add secrets to the key vault

You are going to create three secrets in the Key vault and then add the values saved from previous steps. For each of the secrets, you will need to provide a secret name and provide the value you saved from earlier steps.

SUGGESTED SECRET NAME	SECRET VALUE THAT YOU SAVED EARLIER	EXAMPLE
app-id	The ID of the application created earlier.	8936e905-197b-xxx-xxxx-xxxxxxxxx
app-secret	The client secret specified earlier.	NaeIxxxxxxx---xxxx7eixxx~1g-
storage-account-name	The name of the storage account created earlier.	contosod365datalake

You will need to complete the following steps three times, once for each secret.

1. In the Azure portal, go to the key vault you created earlier and on the left navigation pane, select **Secrets**.
2. Select **Generate/Import**, and in the **Create a secret** dialog box, in the **Upload options** field, select **Manual**.
3. Enter a name for the secret. See the table in the introduction of this section for suggested names.
4. Copy and paste the corresponding secret value in the **Value** field.
5. Select **Enabled**, and then select **Create**.

You will notice the secret created in the list of secrets.

Authorize the application to read secrets in the key vault

1. In **Azure portal**, open the key vault that you created earlier.
2. In the **Add access policy** dialog box, select **Add**.
3. On the left navigation pane, select **Access policies > Add Access Policy** to create a new policy.
4. In the **Add access policy** dialog box, in the **Select principal** field, locate and select the application, **Microsoft Dynamics ERP Microservices**, and then click **Select**.

> **NOTE**
>
> If you can't find **Microsoft Dynamics ERP Microservices**, see the Create Service Principle section in this document.

5. In the **Secret permissions** fields, select **Get** and **List**.
6. In the **Access policy** dialog, select **Add**.

You should see application with access to your key vault as shown below.

APPLICATION	SECRET PERMISSIONS
Microsoft Dynamics ERP Microservices	Get, List

7. Select **Save**.

Install the Export to Data Lake add-in in LCS

Before you can export data to your Data lake from your Finance and Operations apps, you must install the **Export to Data Lake** add-in in LCS. To complete this task, you must be an environment administrator in LCS for the environment that you want to use.

You need the following information before you start. Keep the information handy before you begin.

INFORMATION YOU NEED FOR EXPORT TO DATA LAKE ADD-IN	WHERE CAN YOU FIND IT	EXAMPLE
Your environment Azure AD Tenant ID	Your Azure AD tenant ID in the Azure portal. Sign in to the **Azure portal** and open the **Azure Active Directory** service. Open the **Properties** page and copy the value in the **Directory ID** field.	72f988bf-0000-0000-00000-2d7cd011db47
DNS name of your key vault	This name should have been previously saved. Enter the DNS name of your key vault	https://contosod365datafeedpoc.vault.azure.net/
The secret that contains the name of your storage account	If you used the suggested name, enter **storage-account-name**. If not, enter the secret name you defined.	storage-account-name
Secret that contains the Application ID	If you used the suggested name, enter **app-id**. If not, enter the secret name you defined.	app-id
Secret that contains the Application secret	If you used the suggested name, enter **app-secret**. If not, enter the secret name you defined.	app-secret

1. Sign in to LCS and navigate to your environment.
2. On the **Environment** page, select the **Environment add-ins** tab. If **Export Data Lake** appears in the list, the Data Lake add-in is already installed, and you can skip the rest of this procedure. Otherwise, complete the remaining steps.
3. Select **Install a new add-in**, and in the dialog box, select **Export to Data lake** in the list. If **Export to data lake** isn't listed, the feature might not be available for your environment at this time.

4. In the **Setup add-in** dialog box, provide the required information. To answer the questions, you must already have a storage account. If you don't already have a storage account, create one, or ask your admin to create one on your behalf.
5. Accept the terms of the offer by selecting the check box, and then select **Install**.

The system installs and configures the data lake for the environment. After installation and configuration are complete, you should see **Azure Data Lake** listed on the **Environment** page.

Finance and Operations apps data in Azure Data Lake

2/18/2021 • 9 minutes to read • Edit Online

NOTE

The **Export to Azure Data Lake** feature is in limited preview and may not be available in all regions and environments supported by Finance and Operations apps. If you are unable to find the **Export to Azure Data Lake** functionality in Lifecycle Services (LCS) or your Finance and Operations apps, this feature is not currently available in your environment.

Currently, previews are closed. In the coming months we will enable additional environments in several regions. We are accepting requests from customers who would like to join the preview. If you would like to join a future preview, complete the survey. We will contact you when we are ready to include you. You can also join a Yammer group by completing the survey. You can use the Yammer group to stay in contact and ask questions that will help you understand the feature.

Until the feature is enabled in your environment, you have the option to prototype/plan the feature implementation using GitHub tools. The tools will enable you to export data from your sandbox environment into a storage account in the same format as exported by the feature.

At this time, **Export to Azure Data Lake** feature is not available in Tier-1 (developer) environments. You need a cloud-based Tier-2 or higher environment to enable this feature.

To make aggregate measurements available in a data lake, continue to use the feature in the manner that is described in Make entity store available as a Data Lake.

The **Export to Azure Data Lake** feature lets you copy data from your Finance and Operations apps into your own Azure Data Lake (Gen 2). The system let's you choose the tables and entities to include. After you select the data that you want, the system will make an initial copy. The system then keeps the selected data up to date by applying changes, deletions, and additions. There may be a delay of a few minutes between data changes in your Finance and Operations apps instances and the time when the data is available in your data lake.

Before you can use this feature, you must configure the **Export to Data Lake**. For more information, see Configure export to Azure Data Lake.

Turn on the Export Data to Azure Data Lake feature

An administrator must turn on the **Export to Azure Data Lake** feature before it can be activated. To do this, go to the **Feature management** workspace, locate and select the **Export Data to Azure Data Lake** feature, and then select **Enable**.

After the feature is enabled, you should see the **Export to Azure Data Lake** option under **System administration**.

Select data

NOTE

If the feature, **Export to Azure Data Lake** isn't available in the **Feature management** module in your environment, sign in to the environment and add the following to the URL in your browser address: &mi=DataFeedsDefinitionWorkspace. For example, https://ax123456.cloud.test.dynamics.com/?cmp=USMF&mi=DataFeedsDefinitionWorkspace.

You can select the tables and entities that should be staged in Data Lake.

1. In your environment, go to **System Administration** > **Export to Azure Data Lake**.
2. Select **Configure Data feeds for export to Lake**.
3. On the **Configure data feeds to Data Lake** page, on the **Choose Tables** tab, select the data tables that should be staged in Data Lake. You can search for tables by display name or system name. You can also see whether a table is being synced.
4. When you've finished, select **Add Tables** to add the selected tables to Data Lake.

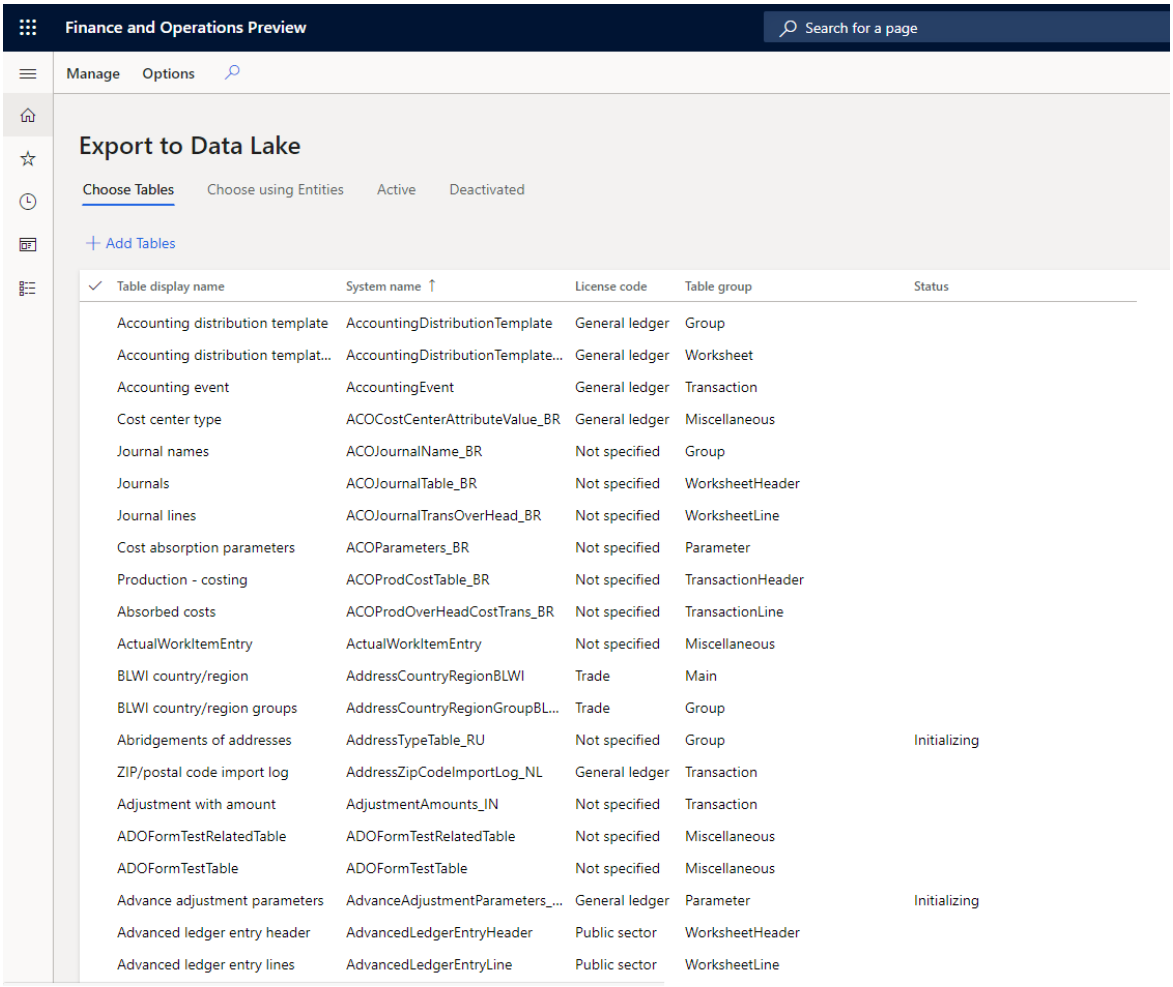


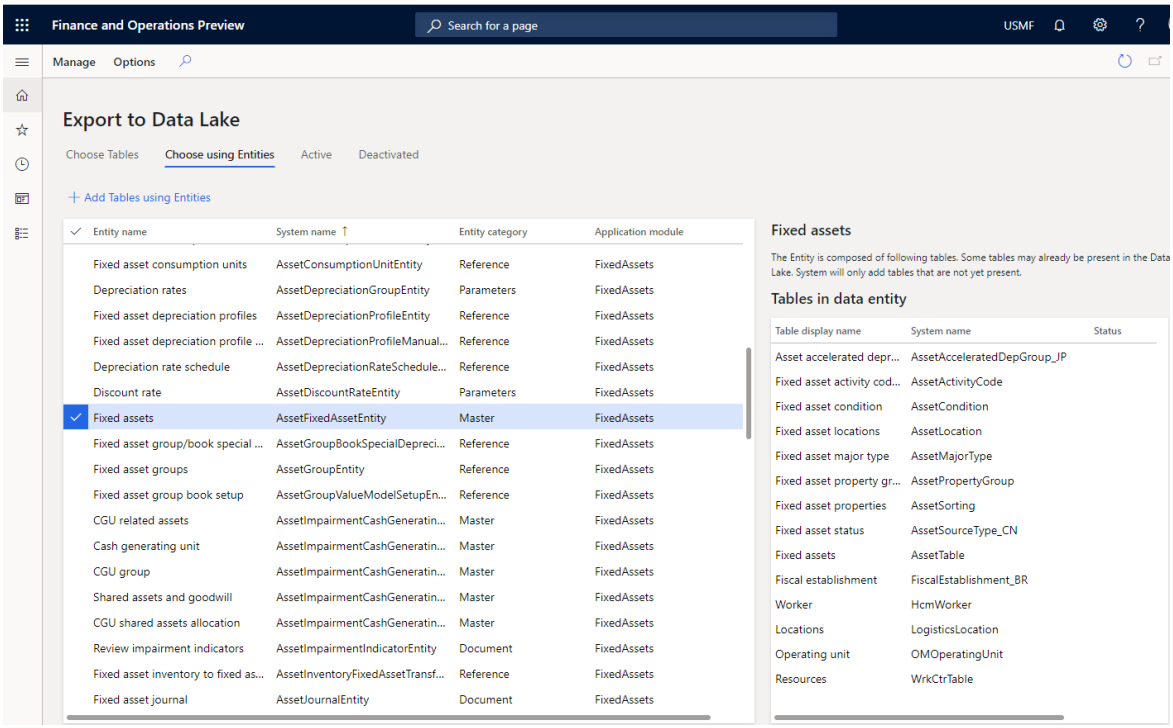
Table display name	System name	License code	Table group	Status
Accounting distribution template	AccountingDistributionTemplate	General ledger	Group	
Accounting distribution templat...	AccountingDistributionTemplate...	General ledger	Worksheet	
Accounting event	AccountingEvent	General ledger	Transaction	
Cost center type	ACOCostCenterAttributeValue_BR	General ledger	Miscellaneous	
Journal names	ACOJournalName_BR	Not specified	Group	
Journals	ACOJournalTable_BR	Not specified	WorksheetHeader	
Journal lines	ACOJournalTransOverHead_BR	Not specified	WorksheetLine	
Cost absorption parameters	ACOParameters_BR	Not specified	Parameter	
Production - costing	ACOProdCostTable_BR	Not specified	TransactionHeader	
Absorbed costs	ACOProdOverHeadCostTrans_BR	Not specified	TransactionLine	
ActualWorkItemEntry	ActualWorkItemEntry	Not specified	Miscellaneous	
BLWI country/region	AddressCountryRegionBLWI	Trade	Main	
BLWI country/region groups	AddressCountryRegionGroupBL...	Trade	Group	
Abridgements of addresses	AddressTypeTable_RU	Not specified	Group	Initializing
ZIP/postal code import log	AddressZipCodeImportLog_NL	General ledger	Transaction	
Adjustment with amount	AdjustmentAmounts_IN	Not specified	Transaction	
ADOFormTestRelatedTable	ADOFormTestRelatedTable	Not specified	Miscellaneous	
ADOFormTestTable	ADOFormTestTable	Not specified	Miscellaneous	
Advance adjustment parameters	AdvanceAdjustmentParameters_...	General ledger	Parameter	Initializing
Advanced ledger entry header	AdvancedLedgerEntryHeader	Public sector	WorksheetHeader	
Advanced ledger entry lines	AdvancedLedgerEntryLine	Public sector	WorksheetLine	

5. Select **Activate data feed**, and then select **OK**. The system may show the status of this table as **Initializing** when you add the table. This means, the system is making an initial copy of data. When the initial copy is complete, the system changes the status to **Running**

In case of an error, the system shows the status **Deactivated**. You can consume data in the lake when you see the **Running** state. If you consume data in the lake while **Initializing** or **Deactivated** status, you may not see all the data.

If you aren't familiar with the specific tables that you require, you can select tables by using entities. Entities are a higher-level abstraction of data and might include multiple tables. By selecting entities, you're also selecting the tables that include them.

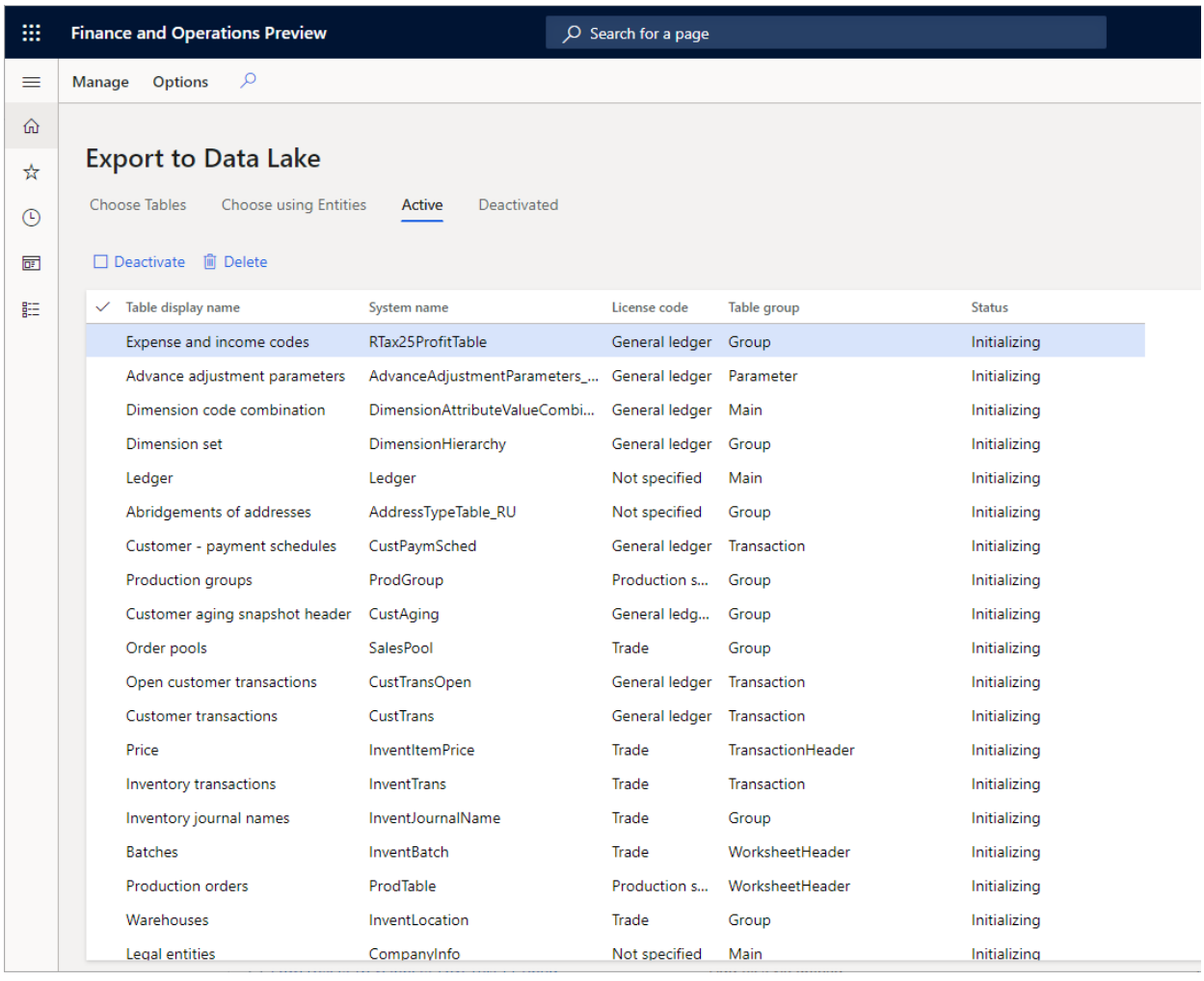
7. On the **Choose using Entities** tab, select the entities, and then select **Add Tables using Entities**.



Regardless of the method that you use to select tables, the tables will be staged in Data Lake.

Monitor the tables in Data Lake

You don't have to monitor or schedule data exports because the system keeps the data updated in Data Lake. However, you can view the status of ongoing data exports on the **Active** tab on the **Configure data feeds to Data lake** page.



Troubleshooting common issues and errors

Export to Data Lake feature is not available in your region and/or your environment at this time

This feature is not available in Tier-1 (developer) environments. You need a sandbox environment (Tier 2 or higher) with Platform updates for version 10.0.13 or higher.

This feature is in limited preview and may not be available in all Azure regions where Finance and Operations apps are available, or this feature may not be available for your environment. If you would like to join a future preview, complete the survey. We will contact you when we are ready to include you. You can also join a Yammer group by completing the survey. You can use the Yammer group to stay in contact and ask questions that will help you understand the feature. We are working hard to make this feature available soon.

Export to Data Lake feature is currently being installed for your environment. Please check back later.

Before you can use this feature, you need to configure the export to Data Lake. For more information, see Configure export to Azure Data Lake.

Export to Data Lake add-in is not installed.

Ask your administrator to install this add-in using Dynamics Lifecycle Services (LCS). Before you can use this feature, you need to configure the export to Data Lake. For more information, see Configure export to Azure Data Lake.

Export to Data Lake feature failed to install in Dynamics Life Cycle Services (LCS).

Ask your administrator to re-install the Export to Data Lake add-in. If this issue persists, contact Support. When you configure the Export to Data Lake feature, the system may report an error. Or, there may be an error when you access the data lake after configuration due to a change in your environments. For more information, see Configure export to Azure Data Lake.

Export to Data Lake feature is temporarily unavailable. Please check back later.

If you see this error for a prolonged period of time, contact Support.

Status codes with extended errors

When an error occurs in a table that you added to Export to Data Lake, you may see an error code in the status column. The following error codes provide the cause of the error and how to correct the issue.

Error status codes 4xx indicate an issue with the table

ERROR CODE	ISSUE	NEXT STEPS
400	The table you added doesn't contain a RecID field.	RecID fields are used by the system to index table data. Tables that don't contain a RecID field can't be added to the Data Lake. If this issue is from a table in a Finance and Operations app, contact Microsoft support. If this table was developed by your partner or ISV, contact the developer to include a RecID field.
401	The table you added is missing in the database.	The table you added is no longer available in the database and the system can't continue updating data in the lake. The table may have been removed because of a software or database update. Contact your database administrator or a developer. If this table was developed by your partner or ISV, contact the developer.

ERROR CODE	ISSUE	NEXT STEPS
402	The RecID field isn't indexed.	The system has detected that the RecID field contained in the table isn't part of an index. This may lead to poor performance in updating the data lake and updates may take longer to reflect in the data lake. If this issue is from a table in a Finance and Operations app, contact Microsoft support. If this table was developed by your partner or ISV, contact the developer.

Error status codes 5xx indicate a system error encountered while exporting data Due to the error, the system has paused data export – data that exists in the lake won't be updated until the error is resolved. Try to deactivate and activate the table to see if this resolves the issue. Note that deactivating and activating the table may cause the system to re-initialize the data in the lake by taking a full copy. If the issue persists, contact Microsoft support with the table name and the error code.

Error status codes 8xx and 9xx indicate a change to the table or database structure since it was added

ERROR CODE	ISSUE	NEXT STEPS
800	There is a change in the data table schema.	The system found a change in the table structure when adding changed data to the lake. This change is typically the result of a software update or a database modification. For example, when a new field is added to a table, or a field is modified or removed, the data updated in the lake may not be in the same format as the existing data. Because of the error, the system has paused data export. The data that exists in the lake won't be updated until the error is resolved. Deactivate and activate the table to see if the issue is resolved.
801	There is an issue with the Change data capture feature.	The Export to Data Lake feature uses the Change data capture feature. Change data can't be exported because the Change data capture feature is disabled in the Finance and Operations apps database. This may be the result of a database maintenance operation. Deactivate and activate the table to see if the issue is resolved. If the issue persists, contact Microsoft support.

ERROR CODE	ISSUE	NEXT STEPS
802	There is an issue with the Change data capture feature for the table.	The Export to Data lake feature uses the Change data capture feature. Change data can't be exported because the Change data capture feature isn't enabled for this table in your Finance and Operations apps database. This may be the result of a database maintenance operation. Deactivate and activate the table to see if the issue is resolved.
900	There is an issue with the Change data capture feature in this environment.	The Export to Data lake feature uses the Change data capture feature. Change data can't be exported because the Change data capture feature isn't enabled for this table in the Finance and Operations apps database. This may be the result of a database maintenance operation. Deactivate and activate the table to see if the issue is resolved. If the issue persists, contact Microsoft support.

NOTE

Deactivating and activating the table may cause the system to re-initialize the data in the lake by creating a full copy. If this is a large table, the initialize process may take some time. In the future, the system may automatically update data in the lake to reflect the table structure changes.

Make Entity store available as a Data Lake

2/18/2021 • 6 minutes to read • Edit Online

IMPORTANT

This feature is currently in public preview. This feature is comprised of the following components:

- **Automated Entity store refresh** - Available in Platform update 23.
- **Entity store data in Microsoft Azure Data Lake (full push)** - Available in Platform update 26.
- **DataFlows for Entity store schemas on PowerBI.com** - Available in a future platform update.
- **Entity store data in Azure Data Lake (trickle feed)** - Available in Platform update 28.
- **Extend analytical workspaces by using PowerBI.com** - Available in a future platform update.

Automated Entity store refresh

You need to enable automated Entity store refresh before enabling Data Lake integration.

1. Go to **System administration** > **Set up** > **Entity store**.

   On the **Entity store** page, a message indicates that you can switch to the **Automated Entity store refresh** option. This option is managed by the system. An admin doesn't have to schedule or monitor the Entity store refresh.

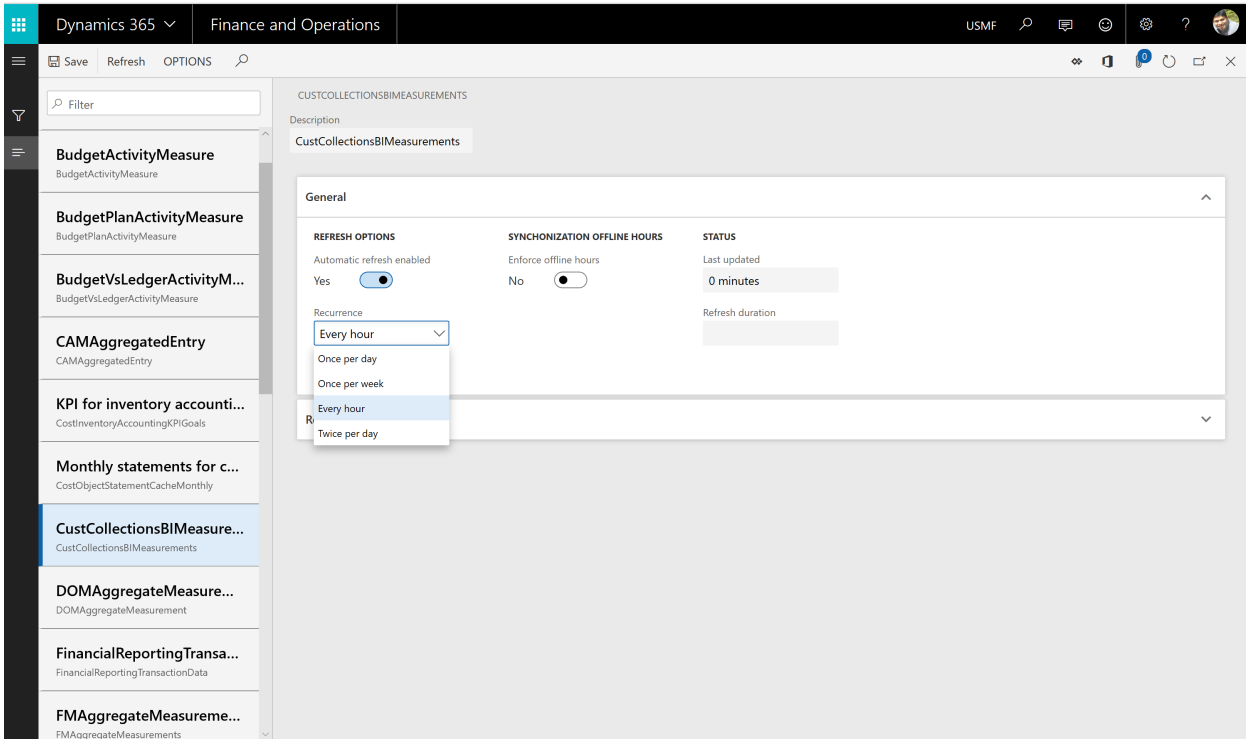
2. Select **Switch now**.

   IMPORTANT

   This action isn't reversible. After you switch to the **Automated Entity store refresh** option, you can't revert to the old user interface (UI) experience.

3. Select **Yes** to continue.

You will now see the new experience.



After the new experience is turned on, you can define the refresh for each aggregate measurement. The following refresh options are available:

- Every hour
- Twice a day
- Once a day
- Once a week

In addition, an admin can refresh any aggregate measurement on demand by selecting the **Refresh** button. Additional options will be added in future platform updates. These options will include options for real-time refresh.

> **IMPORTANT**
>
> When the automated refresh is enabled, in some cases the system may disable refresh of Aggregate measurements. You must revisit aggregate measurements and validate that appropriate refresh intervals have been applied by the system.

Entity store data in Azure Data Lake (full push and trickle feed)

> **IMPORTANT**
>
> This feature is currently in public preview. Do not enable this feature in production environments.

When this feature is turned on, Entity store data isn't populated in the relational Entity store database in the Microsoft subscription. Instead, it's populated in an Azure Data Lake Storage Gen2 account in your own subscription. You can use the full capabilities of PowerBI.com and other Azure tools to work with Entity store.

Before you start, you must complete these tasks in the Azure portal.

1. **Create storage accounts.** Provision a storage account in the same data center where your environment is provisioned. Make a note of the connection string for the storage account, because you will have to provide it later.
2. **Create a Key Vault and a secret.** Provision Azure Key Vault in your own subscription. You will need the

Domain Name System (DNS) name of the Key Vault entry that you created. Also add a secret to Key Vault. As the value, specify the connection string that you made a note of in the previous task. Make a note of the name of the secret, because you will have to provide it later.

3. **Register the app.** Create an Azure Active Directory (Azure AD) application, and grant application programming interface (API) access to Key vault. Make a note of the application ID and its application key (secret), because you will have to provide them later.
4. **Add a service principal to Key Vault.** In Key Vault, use the **Access policies** option to grant the Azure AD application **Get** and **List** permissions. In this way, the application will have access to the secrets in Key Vault.

The following sections describe each task in more detail.

Create storage accounts

1. In the Azure portal, create a new storage account.
2. In the **Create storage account** dialog box, provide values for the following parameter fields:
   - **Location:** Select the data center where your environment is located. If the data center that you select is in a different Azure region, you will incur additional data movement costs. If your Microsoft Power BI and/or your data warehouse is in a different region, you can use replication to move storage between regions.
   - **Performance:** We recommend that you select **Standard**.
   - **Account kind:** You must select **StorageV2**.
3. In the **Advanced options** dialog box, you will see the **Data Lake storage Gen2** option. Select **Enable** under the Hierarchical namespaces feature. If you disable this option, you can't consume data written by Finance and Operations apps with services such as Power BI data flows.
4. Select **Review and create**. When the deployment is completed, the new resource will be shown in the Azure portal.
5. Select the resource, and then select **Settings** > **Access keys**.
6. Make a note of the connection string value, because you will have to provide it later.

Create a Key Vault and a secret

1. In the Azure portal, create a new Key Vault.
2. In the **Create key vault** dialog box, in the **Location** field, select the data center where your environment is located.
3. After Key Vault is created, select it in the list, and then select **Secrets**.
4. Select **Generate/Import**.
5. In the **Create a secret** dialog box, in the **Upload options** field, select **Manual**.
6. Enter a name for the secret. Make a note of the name, because you will have to provide it later.
7. In the value field, enter the connection string that you obtained from the storage account in the previous procedure.
8. Select **Enabled**, and then select **Create**. The secret is created and added to Key Vault.

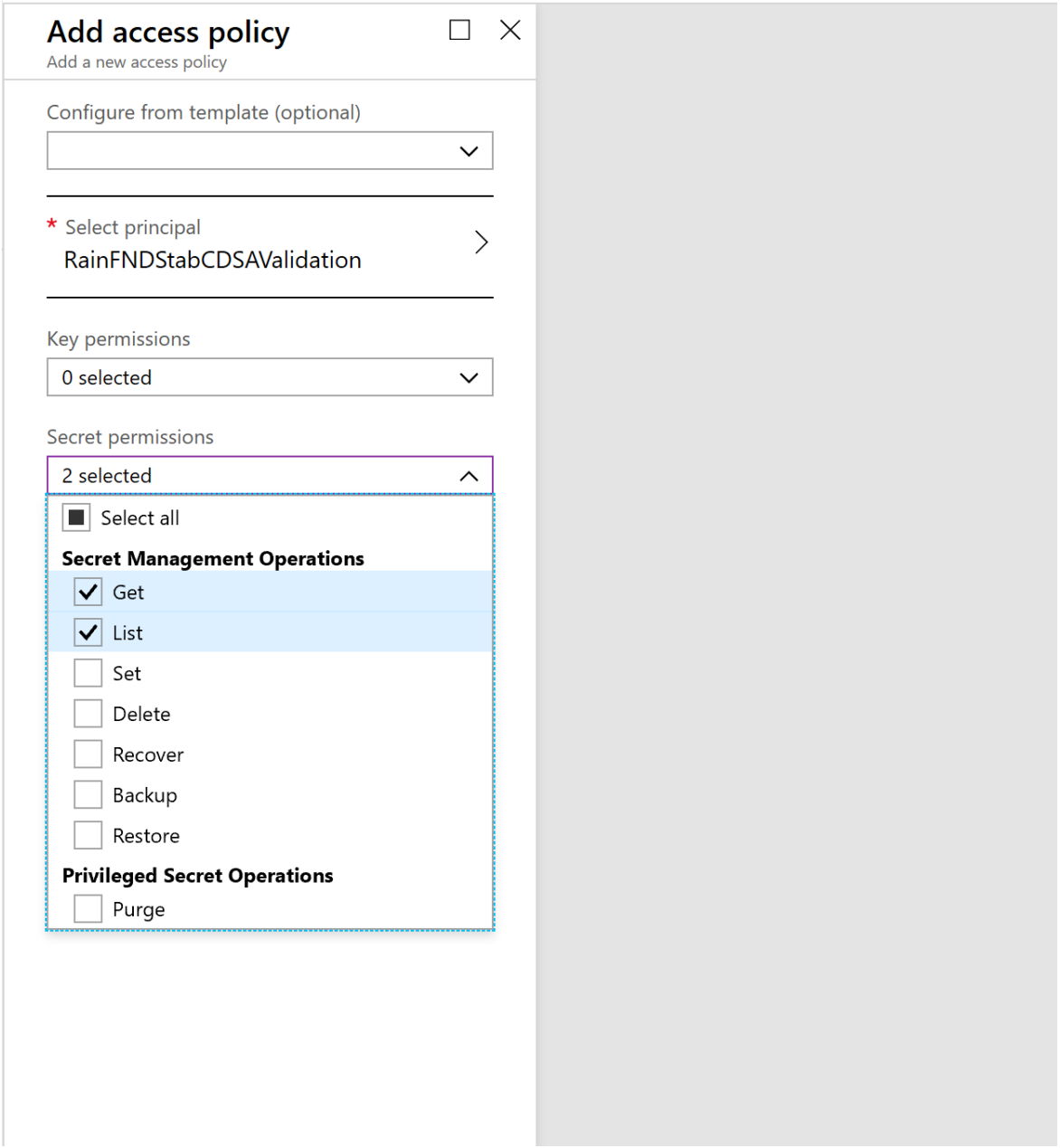
Register the app

1. In the Azure portal, select **Azure Active Directory**, and then select **App registrations**.
2. Select **New registration** at the top of the menu, and enter the following information:
   - **Name** - Enter a friendly name for the app.
   - Select **Accounts in this Organizational directory only** unless your storage account and your Dynamics environment are in different Azure Active Directory domains.
3. After the application is created, select **API permissions**.

4. In the dialog box that appears, select **Add a permission**.
5. You will see a dialog box with a list of APIs. In the list, select **Azure Key Vault**.
6. Select the **Delegated permissions** box, select **user_impersonation**, and then select **Add permissions** to save your changes.
7. Select the **Certificates & secrets** menu on the left navigation pane, and then select **New client secret**.
8. In the **Description** field, enter a name and choose an expiry period. Select **Add**.
9. A secret is generated and shown in the **Value** field.
10. Immediately copy the secret to the clipboard, because it will disappear within one or two minutes. You will have to provide this key to the application later.

Add a service principal to Key Vault

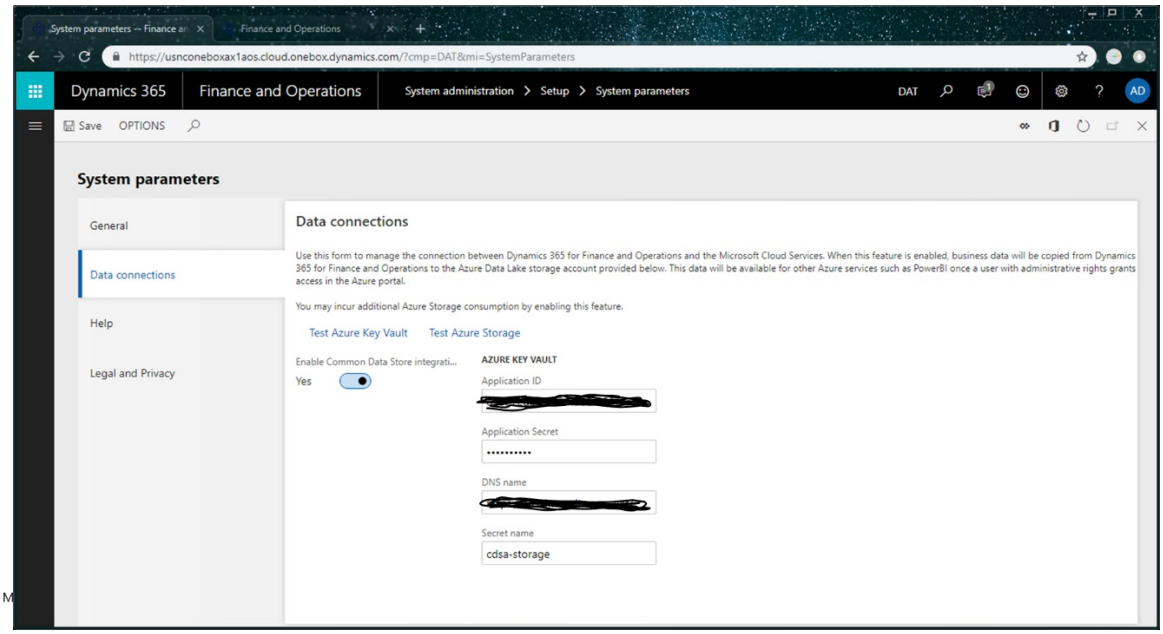
1. In the Azure portal, open Key Vault that you created earlier.
2. Select **Access policies**, and then select **Add** to create a new access policy.
3. In the **Select principal** field, select the name of the application that you previously registered.
4. In the **Key permissions** field, select **Get** and **List** permissions.
5. In the **Secret permissions** field, select **Get** and **List** permissions.



6. Select **Save**.

Work in Entity store in a Data Lake

1. Go to **System administration** > **Set up** > **System parameters**.
2. On the **Data connections** tab, enter the following information that you made a note of earlier in this topic:
    - **Application ID:** Enter the application ID of the Azure AD application that you registered earlier.
    - **Application Secret:** Enter the application key (secret) for the Azure AD application.
    - **DNS name:** Enter the DNS name of Key Vault.
    - **Secret name:** Enter the name of the secret that you added to Key Vault together with connection string information.



3. Select the **Test Azure Key Vault** and **Test Azure Storage** links to validate that system can access the configuration information that you provided.

4. Select the **Enable data connection** check box.

Entity store data should now be populated in the storage location that you provided, not in the relational Entity store database.

The aggregate measurements and refresh options that you select in the Entity store UI should now apply to data that is copied to Data Lake.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Database movement operations home page

2/18/2021 • 2 minutes to read • Edit Online

Database movement operations are a suite of self-service actions that can be used as part of Data Application Lifecycle Management (also referred to as *DataALM*). These actions provide structured processes for common implementation scenarios such as golden configuration promotion, debugging/diagnostics, destructive testing, and general refresh for training purposes.

In this topic, you will learn how to use database movement operations to perform refresh, export, import, and various flavors of point-in-time restore.

Database movement quick start guides

Learn how to perform the individual operations on your Standard or Premier Acceptance Test environments:

- Refresh database
- Export a database
- Import a database
- Point-in-time restore (PITR)
- Point-in-time restore of the production database to a sandbox environment
- Enable just-in-time database access

Step-by-step tutorials

Learn how to achieve common implementation scenarios using DataALM to your advantage:

- Refresh for training purposes
- Debug a copy of the production database
- Export a copy of the standard user acceptance testing (UAT) database
- Golden configuration promotion
- Destructive testing

Database Movement API

The Database Movement application programming interface (API) lets you integrate several of the previously mentioned database movement operations into your overall ALM process. In addition, by using the API together with your preferred scheduling engine, you can build recurrence into the process, so that it runs daily or on demand.

- Overview
- Versioning and support
- Authentication
- Throttling
- Reference

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Database movement toolkit

2/18/2021 • 2 minutes to read • Edit Online

The Database movement toolkit is a ZIP file hosted in Microsoft Dynamics Lifecycle Services (LCS). This toolkit is available for download. It contains a series of scripts that enhance the customer experience for movement of data between developer environments and sandbox environments. This topic explains the components of the toolkit, how to download it, and any recent changes.

Download the latest version

The toolkit is available in the LCS Shared Asset Library under the **Models** section. Search for the entry titled **Database Movement Toolkit** and then download it to your Tier1 DevTest environment.

Toolkit components

The toolkit contains the following primary components:

- **Sqlpackage.exe** - This tool is used to perform extract and publish actions against Microsoft Azure SQL databases, as well as SQL Server databases hosted on Tier1 DevTest environments.
- **PowerShell 7.0** - This is a self-contained version of PowerShell that includes capabilities for parallelism, which increases the performance of transferring data between environments.
- **PowerShell script** - There are several scripts included to provide an enhanced and more automated experience for scenarios such as AX 2012 data upgrade.

Supported scenarios

The toolkit currently supports the following scenarios. More will be added over time.

- Upgrade from AX 2012 - Data upgrade in sandbox environments

Versions

Version 5

FEATURE	DETAILS
Schema publish	Added CleanupSource and CleanupTarget scripts.
Data transfer	Added timer to capture transfer time.

Version 4

FEATURE	DETAILS
Schema publish	Minor improvements.

Version 3

FEATURE	DETAILS
Schema publish	Fixed the scripts to output errors in PublishDiag.log.

Version 2

FEATURE	DETAILS
Schema publish	Fixed the scripts to use current directory reference to Sqlpackage directory.
Schema publish	Fixed the scripts to delete local users from the source database, and tidy up environment-specific tables.
Data transfer	Fixed the scripts to install missing PowerShell modules.

Version 1

FEATURE	DETAILS
Initial	Toolkit uploaded to LCS Shared Asset Library.

Refresh database

2/18/2021 • 9 minutes to read • Edit Online

You can use Microsoft Dynamics Lifecycle Services (LCS) to perform a refresh of the database to a sandbox user acceptance testing (UAT) environment. A database refresh lets you copy the transactional and financial reporting databases of your production environment into the target, sandbox UAT environment. If you have another sandbox environment, you can also copy the databases from that environment to your target, sandbox UAT environment.

> **IMPORTANT**
> Copying production data to your sandbox environment for the purpose of production reporting is not supported.

Self-service database refresh

With the goal of providing Data Application Lifecycle Management (also referred to as *DataALM*) capabilities to our customers without relying on human or manual processes, the Lifecycle Services team has introduced an automated **Refresh database** action. This process is outlined below:

1. Visit your target sandbox on the **Environment Details** page, and click the **Maintain** > **Move database** menu option.
2. Select the **Refresh database** option and choose your source environment.
3. Note the warnings and review the list of data elements that are not copied from the source environment.
4. The refresh operation will begin immediately.

Refresh operation failed

In case of failure, the option to perform a rollback is available. By clicking the **Rollback** option after the operation has initially failed, your target sandbox environment will be restored to the state it was before the refresh began. This is made possible by the Azure SQL point-in-time restore capability to restore the database. This is often required if a customization, that is present in the target sandbox, cannot complete a database synchronization with the newly refreshed data.

To determine the root cause of the failure, use the available buttons to download the runbook logs before you start the rollback operation.

Data elements that aren't copied during refresh

The information in this section lists certain elements of the database that are not copied over to the target environment during a database refresh operation.

When refreshing a production environment to a sandbox environment or a sandbox to another sandbox environment

- Email addresses in the LogisticsElectronicAddress table.
- SMTP Relay server in the SysEmailParameters table.
- Print Management settings in the PrintMgmtSettings and PrintMgmtDocInstance tables.
- All users except the admin will be set to **Disabled** status.

When refreshing from sandbox environment to production environment

This is also referred to as Golden configuration promotion.

- Batch job history is stored in the BatchJobHistory, BatchHistory, and BatchConstraintHistory tables.

These elements are removed for all database refresh operations

- Environment-specific records in the SysServerConfig, SysServerSessions, SysCorpNetPrinters, SysClientSessions, BatchServerConfig, and BatchServerGroup tables.
- Document attachments in the DocuValue table. These attachments include any Microsoft Office templates that were overwritten in the source environment.
- All batch jobs are set to **Withhold** status.
- All users will have their partition value reset to the "initial" partition record ID.
- All Microsoft-encrypted fields will be cleared, because they can't be decrypted on a different database server. An example is the **Password** field in the SysEmailSMTPPassword table.
- Maintenance mode settings will be disabled even if it was enabled in source.

Some of these elements aren't copied because they are environment-specific. Examples include BatchServerConfig and SysCorpNetPrinters records. Other elements aren't copied because of the volume of support tickets. For example, duplicate emails might be sent because Simple Mail Transfer Protocol (SMTP) is still enabled in the UAT environment, invalid integration messages might be sent because batch jobs are still enabled, and users might be enabled before admins can perform post-refresh cleanup activities.

Environment administrator

The System Administrator account in the target environment (UserId of 'Admin') is reset to the value found in the web.config file on the target. This should be the same value as that of the Administrator from Lifecycle Services. To preview which account this will be, visit your target sandbox **Environment Details** page in LCS. The value of the **Environment Administrator** field that was selected when the environment was first deployed is updated to be the System Administrator in the transactional database. This also means that the tenant of the environment will be that of the Environment Administrator.

If you have used the Admin User Provisioning Tool on your environment to change the web.config file to a different value, it may not match what is in Lifecycle Services. If you require a different account to be used, you will need to deallocate and delete the target sandbox, and redeploy selecting another account. After this, you can perform another refresh database action to restore the data.

An environment can't be refreshed from one tenant to another. This restriction applies even to .onmicrosoft.com tenants. You should make sure that the admin accounts in the source and target environments are from the same tenant domain.

Conditions of a database refresh

Here is the list of requirements and conditions of operation for a database refresh:

- A refresh performs a delete operation on the original target database.
- The target environment will be available until the database copy has reached the target server. After that point, the environment will be offline until the refresh process is completed.
- The refresh will affect only the application and Financial Reporting databases.
- Documents in Azure blob storage are not copied from one environment to another. This means that attached document handling documents and templates won't be changed and will remain in their current state.
- All users except the Admin user and other internal service user accounts will be unavailable. This process allows the Admin user to delete or obfuscate data before allowing other users back into the system.
- The Admin user must make required configuration changes, such as reconnecting integration endpoints to specific services or URLs.
- All data management framework with recurring import and export jobs must be fully processed and stopped in the target system prior to initiating the restore. In addition, we recommend that you select the database from the source after all recurring import and export jobs have been fully processed. This will ensure there are no orphaned files in Azure storage from either system. This is important because orphaned files cannot be processed after the database is restored in the target environment. After the restore, the integration jobs can be resumed.
- Any user with a role of Project owner or Environment manager in LCS will have access to the SQL and

machine credentials for all non-production environments.

- The databases must be hosted in the same Azure geographic region, unless the databases are Spartan-managed. Databases are Spartan-managed when you see 'spartan' as part of the fully qualified SQL server address.
- The allocated database capacity of the source environment must be less than the maximum database capacity of the target environment.

Steps to complete after a database refresh for environments that use Commerce functionality

> **IMPORTANT**
>
> Some environment-specific records are not included in automated database movement operations and require additional steps. These include the following:
>
> - Commerce self-service installer references
> - Commerce Scale Unit channel database configuration records

If you copy a database between environments, Commerce capabilities in the destination environment will not be fully functional until you perform the following additional steps.

Initialize Commerce Scale Units

If you are moving a database to a sandbox UAT or production environment, you must Initialize Commerce Scale Unit after the database movement operation is complete. The Commerce Scale Unit's association from the source environment will not copy over to the destination environment.

Synchronize Commerce self-service installers

To be able to access Commerce self-service installers in HQ, you must Synchronize self-service installers after the database movement operation is complete.

> **IMPORTANT**
>
> The Environment re-provisioning step has now been fully automated as part of database movement operations, and no longer needs to be run manually. The Environment re-provisioning tool is still available in the Asset Library and may be used in certain situations to mitigate error conditions.

To run the Environment re-provisioning tool on the destination environment, run the following steps:

1. In your project's **Asset Library**, in the **Software deployable packages** section, select **Import**.
2. From the list of shared assets, select the **Environment Reprovisioning Tool**.
3. On the **Environment details** page for your destination environment, select **Maintain** > **Apply updates**.
4. Select the **Environment Reprovisioning** tool that you uploaded earlier, and then select **Apply** to apply the package.
5. Monitor the progress of the package deployment.

For more information about how to apply a deployable package, see Create deployable packages of models. For more information about how to manually apply a deployable package, see Install deployable packages from the command line.

Re-activate POS devices

If you use point of sale (POS) devices, after you import a database you must activate the POS devices again. Previously activated devices in the destination environment will no longer function. For more information, see Point of sale device activation.

Known issues

Refresh is denied for environments that run Platform update 20 or earlier

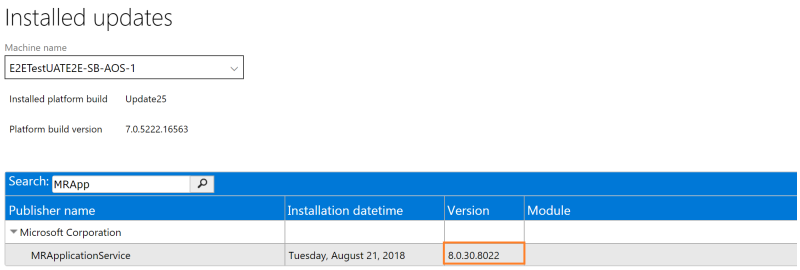
The database refresh process can't currently be completed if the environment is running Platform update 20 or earlier. For more information, see the list of currently supported platform updates.

Incompatible version of Financial Reporting between source and target environments

The database refresh process (self-service or via a service request) can't be completed successfully if the version of Financial Reporting in the target environment is earlier than the version in the source environment. To resolve this issue, update both environments so that they have the latest version of Financial Reporting.

To determine the version you have installed in your source and target environments, visit the **View detailed version information** link on the **Environment Details** page.

Search for **MRApplicationService** and ensure that the target environment is greater than or equal to the source environment.



For customers that are using version 8.1 or later:

1. Go to the **Update** tiles for your UAT environment. Save the updates to your Project asset library.
2. Apply this package to your UAT environment.
3. Verify that the error has been resolved.

For customers that are using version 8.0 or earlier:

1. Review the Environment history of your source environment. Specifically, look for any "Platform and application binary package" that might have been deployed to the source environment and not the target environment.
2. Apply this binary package to your target environment.
3. Verify that the error has been resolved.

Incompatible application versions between source and target environments

The database refresh process (self-service or via service request) cannot be completed if the Application release of your source and target environment are not the same. This is because the data upgrade process is not executed by database movement operations such as refresh, and data loss can occur.

If upgrading your sandbox UAT environment to a newer Application version (for example, 7.3 to 8.1), be sure to perform the database refresh action prior to starting the upgrade. After your sandbox is upgraded to the newer version, you cannot restore an older production environment database in to the sandbox UAT environment.

Conversely, if your production environment is newer than your target sandbox, you will need to either upgrade the target sandbox prior to the refresh or simply deallocate, delete, and redeploy prior to performing the refresh.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

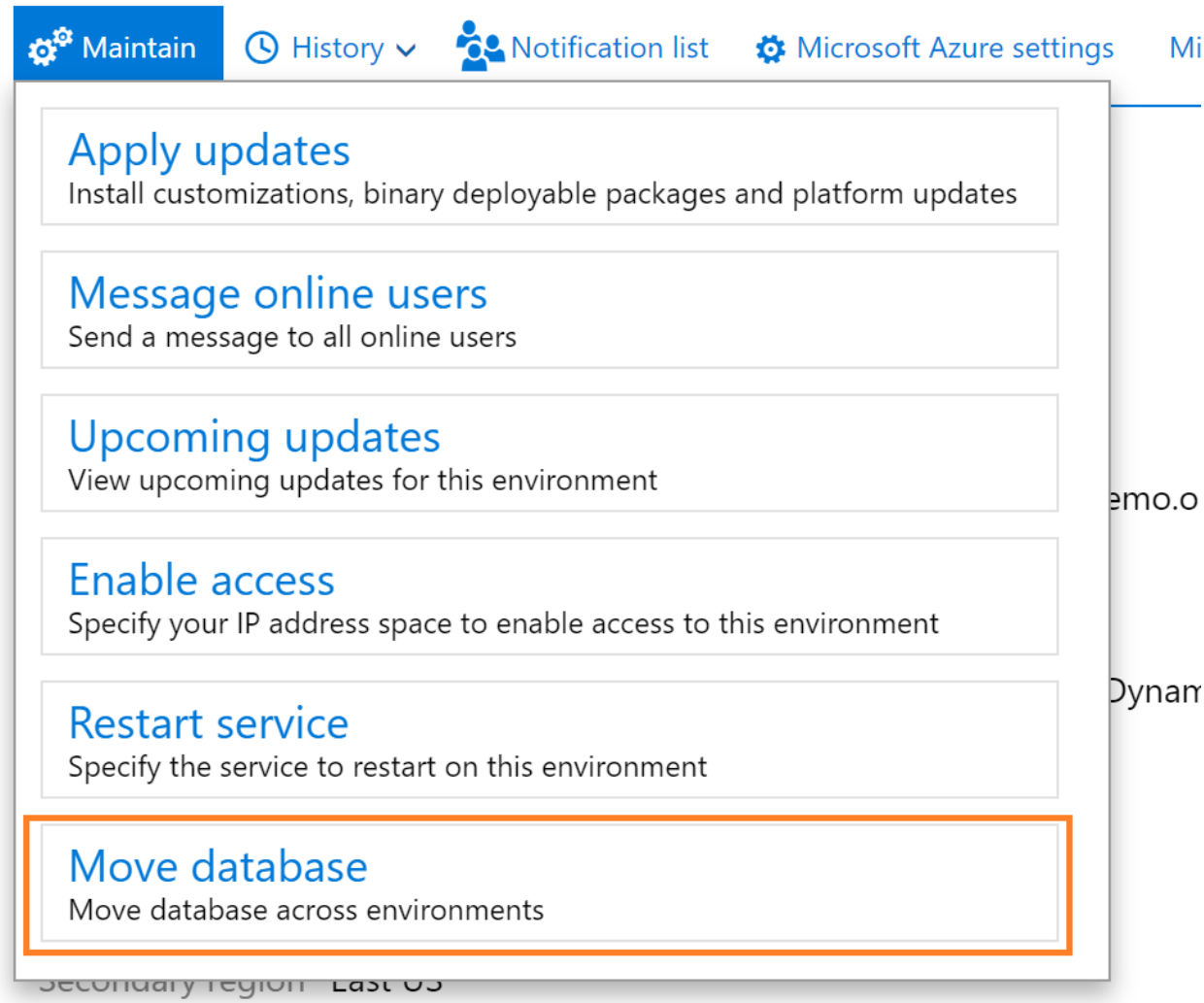
Export a database

2/18/2021 • 4 minutes to read • Edit Online

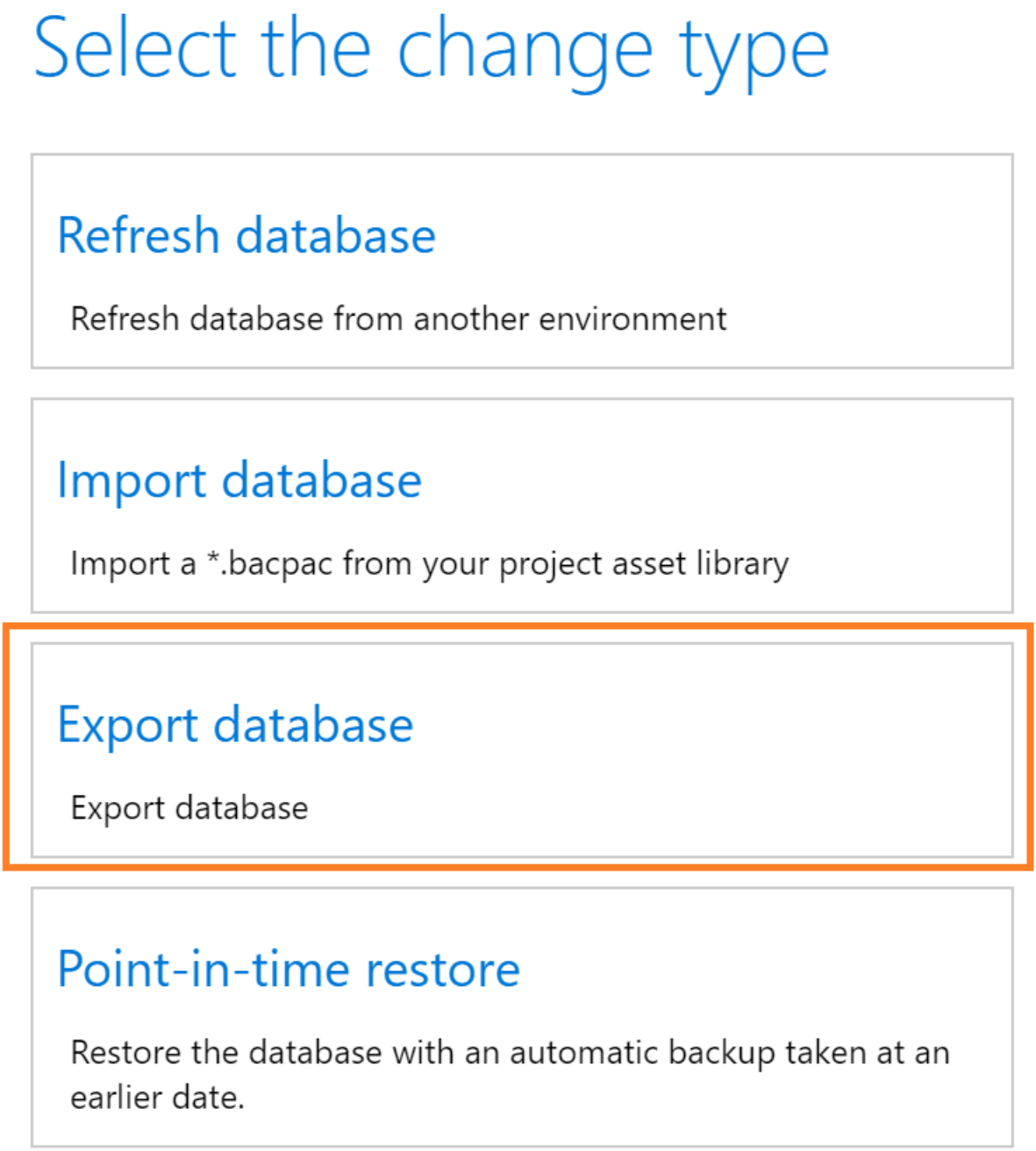
You can use Microsoft Dynamics Lifecycle Services (LCS) to export a database from a sandbox user acceptance testing (UAT) environment to the Asset library.

Self-service export database

From your sandbox **Environment Details** page, click the **Maintain** menu, and then select **Move database**.

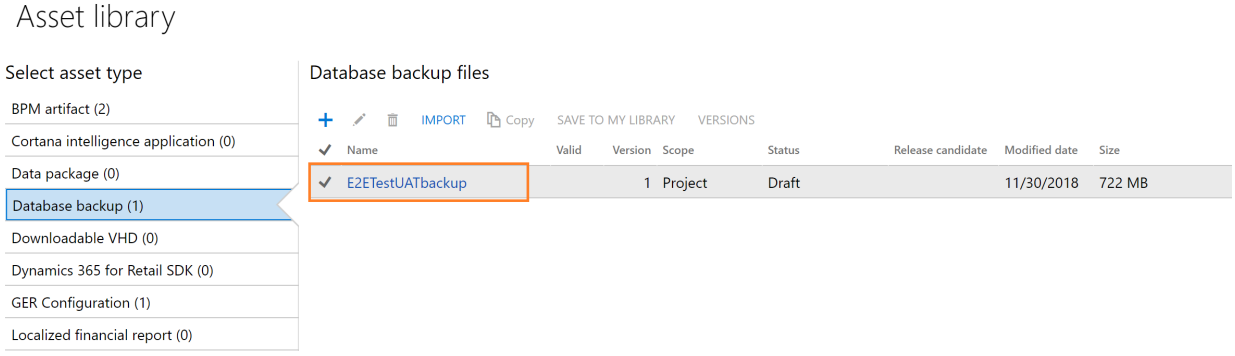


A slider pane will open on the page where you can use the **Export database** action.



The environment will be unavailable for other servicing operations, such as Sandbox refresh or package deployment during this time. The source environment will be usable from a Dynamics user perspective.

After the export operation completes successfully, sign out of the servicing operation on your **Environment details** page. You can then see the asset in your **Asset Library** in the **Database backups** section.



The **.bacpac** files are stored here and can be manually downloaded to your Tier 1 developer environments for import. In the future, Microsoft will provide APIs to trigger the export action, as well as list the available backup files in your asset library. This includes the secured URL for automatically downloading a backup asset file or copying it directly to your secure blob storage using Microsoft Azure Storage SDKs.

Maximum limit 50 GB on exported bacpacs

To maintain the system that performs database export from LCS, a limit on the maximum bacpac size is being imposed. This limit is set at 50 GB for each bacpac exported. The reasons for this limit include:

- A centralized system is performing the exports for multiple customers in the same geographic region, and this system has constraints on disk space.
- Azure SQL neatly compresses the data in the bacpac format and in many cases, where customers exceed 50 GB, customizations or binary data drastically oversize the backup file.

If you experience an export failure because the resulting bacpac is over 50 GB in size, try running the following SQL script against your sandbox database to identify the top 15 tables by size in megabytes. Any tables that are for data entity staging (they will have "staging" at the end of the table name) can be truncated. Any tables that are storing binary or blob data (JSON/XML/binary) should either be truncated or the contents of that field should be deleted to free up space. Binary data cannot be compressed, so storing large volumes of data in the database itself will cause you to quickly reach the 50 GB limit.

```
USE [YourDBName] -- replace your dbname
GO
SELECT top 15
s.Name AS SchemaName,
t.Name AS TableName,
p.rows AS RowCounts,
CAST(ROUND((SUM(a.used_pages) / 128.00), 2) AS NUMERIC(36, 2)) AS Used_MB,
CAST(ROUND((SUM(a.total_pages) - SUM(a.used_pages)) / 128.00, 2) AS NUMERIC(36, 2)) AS Unused_MB,
CAST(ROUND((SUM(a.total_pages) / 128.00), 2) AS NUMERIC(36, 2)) AS Total_MB
FROM sys.tables t
INNER JOIN sys.indexes i ON t.OBJECT_ID = i.object_id
INNER JOIN sys.partitions p ON i.object_id = p.OBJECT_ID AND i.index_id = p.index_id
INNER JOIN sys.allocation_units a ON p.partition_id = a.container_id
INNER JOIN sys.schemas s ON t.schema_id = s.schema_id
GROUP BY t.Name, s.Name, p.Rows
ORDER BY Total_MB DESC
GO
```

Export operation failure

Most often, export operations fail because the process in LCS times out while it's waiting for a response from Microsoft Azure SQL Database. You can use the **Resume** button to reconnect LCS to the ongoing export process and see it through to completion. If more than 24 hours have passed since you began the export, the pending asset in the LCS Project asset library will be expired. In this case, you must roll back the export operation and restart it.

To cancel an export operation that has failed, you can use the **Rollback** button.

Data elements that aren't exported

When you export a database backup from an environment, some elements of the database aren't exported in the backup file. Here are some examples:

- Email addresses in the **LogisticsElectronicAddress** table.
- Batch job history in the **BatchJobHistory**, **BatchHistory**, and **BatchConstraintsHistory** tables.
- SMTP Relay server in the **SysEmailParameters** table.
- Print Management settings in the **PrintMgmtSettings** and **PrintMgmtDocInstance** tables.
- Environment-specific records in the **SysServerConfig**, **SysServerSessions**, **SysCorpNetPrinters**,

SysClientSessions, **BatchServerConfig**, and **BatchServerGroup** tables.

- Document attachments in the **DocuValue** table. These attachments include any Microsoft Office templates that were overwritten in the source environment.
- All users except the admin will be set to **Disabled** status.
- All batch jobs are set to **Withhold** status.
- All users will have their partition value reset to the "initial" partition record ID.
- All Microsoft-encrypted fields will be cleared, because they can't be decrypted on a different database server. An example is the **Password** field in the **SysEmailSMTPPassword** table.

Known issues

Export ran for some time and then reached a "Preparation failed" state

The export process can time out on Azure SQL Database when large databases are involved. In some cases, the export process can be recovered by using the **Resume** action from LCS. The Lifecycle Services team is working to identify known error codes, so these can be added to the logs for the export database operation to help guide users toward a resolution. These known error codes will be added in a future release of LCS.

Export doesn't show any progress in LCS

The export process differs from other database movement operations, and the general package deployment doesn't use a runbook. Therefore, the progress indicator in LCS doesn't show any output, even though it typically shows output in other scenarios. The LCS team is working to identify known error codes so that they can be added to the logs for the export database operation to help guide users toward a resolution. These known error codes will be added in a future release of LCS.

Import a database

2/18/2021 • 5 minutes to read • Edit Online

You can use Microsoft Dynamics Lifecycle Services (LCS) to import a golden configuration database into a sandbox user acceptance testing (UAT) environment.

Self-service import database

To import a database that is prepared from a developer environment to a standard user acceptance test (UAT), or a database previously exported from a UAT environment, follow the steps outlined below:

1. Go to your target sandbox **Environment Details** page, and select the **Maintain** > **Move database** menu option.
2. Select **Import database** and choose your source database backup (.bacpac format) file from the Asset Library.
3. Note the warnings. Review the list of data elements that are cleaned up from the backup file.
4. The import operation will begin immediately.

To import a database to a developer environment after you've downloaded a database backup (.bacpac) file, you can begin the manual import operation on your Tier 1 environment. When you import the database, we recommend that you follow these guidelines:

- Keep a copy of the existing AxDB database, so that you can revert to it later if needed.
- Import the new database under a new name, such as **AxDB_fromProd**.

To ensure the best performance, copy the *.bacpac file to the local computer that you're importing from. Download sqlpackage .NET Core for Windows from Get sqlpackage .NET Core for Windows. Open a **Command Prompt** window, and run the following commands from the sqlpackage .NET Core folder.

```
SqlPackage.exe /a:import /sf:D:\Exportedbacpac\my.bacpac /tsn:localhost /tdn:<target database name>
/p:CommandTimeout=1200
```

Here is an explanation of the parameters:

- **tsn (target server name)** – The name of the Microsoft SQL Server instance to import into.
- **tdn (target database name)** – The name of the database to import into. The database should **not** already exist.
- **sf (source file)** – The path and name of the file to import from.

> **NOTE**
> During import, the user name and password aren't required. By default, SQL Server uses Microsoft Windows authentication for the user who is currently signed in.

For information about how to complete the manual import operations into a Tier 1 environment, see Import the database.

Import operation failure

If the import operation isn't successful, you can do a *rollback*. If you select the **Rollback** option after the initial

failure of the operation, your target sandbox environment is restored to the state that it was in before the import began. The rollback operation is made available by the Microsoft Azure SQL Database point-in-time restore capability for restoring the database. Rollback is often required if a customization that is present in the target sandbox can't complete a database synchronization with the newly imported data.

Data elements that require attention after import

Specific activities must be completed when you import a database backup into a sandbox UAT environment. Here are some examples:

- Make sure that the source database contains only a single record in the Partitions table.
- Make sure that email capabilities are correctly reconfigured or turned off, according to your requirements.
- Make sure that integration settings are turned on or off, according to your requirements.
- Make sure that Application Object Server (AOS) servers are added back to required batch groups.
- Make sure that system Help and task guides are reconnected.
- Make sure that batch jobs are set to a status of **Waiting**.
- Make sure that users are re-enabled.

Environment admin

The system admin account in the target environment (**Admin** user ID) is reset to the value that is found in the web.config file in that environment. This account should be the same as the admin account from LCS. To preview which account this account will be, visit the **Environment details** page for your target sandbox in LCS. The value that was selected in the **Environment Administrator** field when the environment was first deployed is updated to the system admin in the transactional database. Therefore, the tenant of the environment will be the tenant of the environment admin.

If you've used the Admin User Provisioning Tool on your environment to change the web.config file, the value might not match the value in LCS. If you require that a different account be used, you must deallocate and delete the target sandbox, redeploy, and select another account. You can then do another database refresh action to restore the data.

Steps to complete after a database import for environments that use Commerce functionality

> **IMPORTANT**
>
> Some environment-specific records are not included in automated database movement operations and require additional steps. These include the following:
>
> - Commerce self-service installer references
> - Commerce Scale Unit channel database configuration records

If you copy a database between environments, Commerce capabilities in the destination environment will not be fully functional until you perform the following additional steps.

Initialize Commerce Scale Units

If you are moving a database to a sandbox UAT or production environment, you must Initialize Commerce Scale Unit after the database movement operation is complete. The Commerce Scale Unit's association from the source environment will not copy over to the destination environment.

Synchronize Commerce self-service installers

To be able to access Commerce self-service installers in HQ, you must Synchronize self-service installers after the database movement operation is complete.

IMPORTANT

The Environment re-provisioning step has now been fully automated as part of database movement operations, and no longer needs to be run manually. The Environment re-provisioning tool is still available in the Asset Library and may be used in certain situations to mitigate error conditions.

To run the Environment re-provisioning tool on the destination environment, run the following steps:

1. In your project's **Asset Library**, in the **Software deployable packages** section, select **Import**.
2. From the list of shared assets, select the **Environment Reprovisioning Tool**.
3. On the **Environment details** page for your destination environment, select **Maintain** > **Apply updates**.
4. Select the **Environment Reprovisioning** tool that you uploaded earlier, and then select **Apply** to apply the package.
5. Monitor the progress of the package deployment.

For more information about how to apply a deployable package, see Create deployable packages of models. For more information about how to manually apply a deployable package, see Install deployable packages from the command line.

Re-activate POS devices

If you use point of sale (POS) devices, after you import a database you must activate the POS devices again. Previously activated devices in the destination environment will no longer function. For more information, see Point of sale device activation.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Database point-in-time restore (PITR)

2/18/2021 • 4 minutes to read • Edit Online

You can use Microsoft Dynamics Lifecycle Services (LCS) to perform the point-in-time restore (PITR) for a sandbox user acceptance testing (UAT) environment. Microsoft maintains automated backups of the business and financial reporting databases for 28 days for Production environments and 14 days for Sandbox environments.

Self-service point-in-time restore

To restore the database of a standard user acceptance test (UAT) environment to a previous point-in-time, follow the steps outlined below:

1. Go to your target sandbox **Environment Details** page, and select the **Maintain** > **Move database** menu option.
2. Select the **Point-in-time restore** option and choose a point-in-time.
3. Note the warnings. Review the list of data elements that are not copied over from the previous point-in-time.
4. The restore operation will begin immediately.

Restore operation failed

In the event of failure, the option to do a **rollback** is available. If you select the rollback option after the operation originally fails, your target sandbox environment is restored to the state that it was in before the restore began. A rollback is often required if a customization that is present in the target sandbox environment can't complete a database synchronization with the newly restored data.

To determine the root cause of the failure, download the runbook logs using the available buttons before starting the rollback operation.

Data elements that need attention after restore

When you restore a database from a previous point in time, keep in mind that the database is provided "as was." For example, batch jobs and other data elements in the system can be in an in-progress state. These elements will require manual review.

Environment administrator

The system administrator account in the target environment (that is, the account that has a **UserId** value of **Admin**) is reset to the value that is found in the web.config file in the target environment. This value should match the value of the administrator account in LCS. To preview which account will be used, go to the **Environment Details** page for your target sandbox environment in LCS. The value that was selected in the **Environment Administrator** field when the environment was first deployed is updated to the system administrator in the transactional database. The tenant of the environment will be the tenant of the environment administrator.

If you've used the Admin User Provisioning Tool in your environment to change the value in the web.config file, the value might not match the value in LCS. If you require a different account, you must deallocate and delete the target sandbox environment, and then redeploy it by selecting another account. You can then do another refresh database action to restore the data.

Steps to complete after a database restore for environments that use Commerce functionality

IMPORTANT

Some environment-specific records are not included in automated database movement operations and require additional steps. These include the following:

- Commerce self-service installer references
- Commerce Scale Unit channel database configuration records

If you copy a database between environments, Commerce capabilities in the destination environment will not be fully functional until you perform the following additional steps.

Initialize Commerce Scale Units

If you are moving a database to a sandbox UAT or production environment, you must Initialize Commerce Scale Unit after the database movement operation is complete. The Commerce Scale Unit's association from the source environment will not copy over to the destination environment.

Synchronize Commerce self-service installers

To be able to access Commerce self-service installers in HQ, you must Synchronize self-service installers after the database movement operation is complete.

IMPORTANT

The Environment re-provisioning step has now been fully automated as part of database movement operations, and no longer needs to be run manually. The Environment re-provisioning tool is still available in the Asset Library and may be used in certain situations to mitigate error conditions.

To run the Environment re-provisioning tool on the destination environment, run the following steps:

1. In your project's **Asset Library**, in the **Software deployable packages** section, select **Import**.
2. From the list of shared assets, select the **Environment Reprovisioning Tool**.
3. On the **Environment details** page for your destination environment, select **Maintain** > **Apply updates**.
4. Select the **Environment Reprovisioning** tool that you uploaded earlier, and then select **Apply** to apply the package.
5. Monitor the progress of the package deployment.

For more information about how to apply a deployable package, see Create deployable packages of models. For more information about how to manually apply a deployable package, see Install deployable packages from the command line.

Re-activate POS devices

If you use point of sale (POS) devices, after you import a database you must activate the POS devices again. Previously activated devices in the destination environment will no longer function. For more information, see Point of sale device activation.

Known issues

Breaking the chain of available restore points

Several frequently used actions create a new database that won't have the same restore point history as the previously used database. These actions include point-in-time restores, database refreshes, database imports, and point-in-time restores from the production environment to the sandbox environment. In addition, if a software deployable package that you apply to your environment fails during update of the database, and you use the rollback functionality in LCS, the rollback functionality does a point-in-time restore of the database, and that restore also creates a new database.

Although the new database doesn't have any restore point history, it does begin to accrue new restore points from that time onward. After you perform any of the previously mentioned actions, you can't perform it again by using the same restore date and time.

Restore is denied for environments that run Platform update 20 or earlier

The restore database process cannot be completed if the environment is running Platform update 20 or earlier. For more information, see the list of currently supported Platform updates in the Software lifecycle policy and cloud releases.

Point-in-time restore of the production database to a sandbox environment

2/18/2021 • 10 minutes to read • Edit Online

You can use Microsoft Dynamics Lifecycle Services (LCS) to do a point-in-time restore (PITR) of the production database to a user acceptance testing (UAT) sandbox environment. Microsoft maintains automated backups of the business and financial reporting databases for 28 days for Production environments and 7 days for Sandbox environments.

> **IMPORTANT**
> Microsoft doesn't support copying production data to a sandbox environment for the purpose of production reporting.

Self-service point-in-time restore Production to Sandbox

To provide customers with data application lifecycle management (DataALM) capabilities that don't rely on human or manual processes, the Lifecycle Services team has introduced an automated refresh database action. Here is an overview of the process for doing a self-service database refresh.

1. Go to the **Environment Details** page for your target Sandbox, and select **Maintain** > **Move database** button.
2. Select the **Point-in-time restore Prod to Sandbox** option, and then select the desired point in time.
3. Make a note of the warnings, and review the list of data elements that aren't copied from the source environment's previous point in time.
4. The restore operation begins immediately.

> **IMPORTANT**
> Self-service point in time restore (PITR) is not supported between environments that are on different regions. For more details, refer to the "Known issues" section later in this topic.

Restore operation failure

If the restore operation isn't successful, you can do a rollback. If you select the **Rollback** option after the operation originally fails, your target sandbox environment is restored to the state that it was in before the refresh began. Rollbacks are made available by the PITR capability of Azure SQL Database. They are typically required if a customization that is present in the target sandbox environment can't complete a database synchronization with the newly refreshed data.

To determine the root cause of the failure, use the available buttons to download the runbook logs before you start the rollback operation.

Data elements that aren't copied during restore copy

When you refresh a production environment to a sandbox environment, or a sandbox environment to another sandbox environment, some elements of the database aren't copied over to the target environment. Here are some examples:

- Email addresses in the LogisticsElectronicAddress table.
- Batch job history in the BatchJobHistory, BatchHistory, and BatchConstraintHistory tables.

- Simple Mail Transfer Protocol (SMTP) Relay server in the SysEmailParameters table.
- Print Management settings in the PrintMgmtSettings and PrintMgmtDocInstance tables.
- Environment-specific records in the SysServerConfig, SysServerSessions, SysCorpNetPrinters, SysClientSessions, BatchServerConfig, and BatchServerGroup tables.
- Document attachments in the DocuValue table. These attachments include any Microsoft Office templates that were overwritten in the source environment.
- All users except the admin will be set to **Disabled** status.
- All batch jobs will be set to **Withhold** status.
- All users will have their partition value reset to the "initial" partition record ID.
- All Microsoft-encrypted fields will be cleared, because they can't be decrypted on a different database server. An example is the **Password** field in the SysEmailSMTPPassword table.

Some of these elements aren't copied because they are environment-specific. Examples include BatchServerConfig and SysCorpNetPrinters records. Other elements aren't copied because of the volume of support tickets. For example, duplicate emails might be sent because SMTP is still turned on in the UAT environment, invalid integration messages might be sent because batch jobs are still enabled, and users might be enabled before admins can perform post-refresh cleanup activities.

Environment administrator

The system administrator account in the target environment (that is, the account that has a **UserId** value of **Admin**) is reset to the value that is found in the web.config file in the target environment. This value should match the value of the administrator account in LCS. To preview this account, go to the **Environment Details** page for your target sandbox environment in LCS. The value that was selected in the **Environment Administrator** field when the environment was first deployed is updated so that it matches the system administrator account in the transactional database. Therefore, the tenant of the environment will also be the tenant of the environment administrator.

If you've used the Admin User Provisioning Tool on your environment to change the value in the web.config file, that value might not match the value in LCS. If you require that a different account be used, you must deallocate and delete the target sandbox environment, and then redeploy it and select another account. You can then perform another refresh database action to restore the data.

An environment can't be refreshed from one tenant to another. This restriction applies even to .onmicrosoft.com tenants. You should make sure that the admin accounts in the source and target environments are from the same tenant domain.

Conditions for doing a PITR copy of a production environment to a sandbox environment

Here is the list of requirements and conditions of operation for a database refresh:

- A refresh performs a delete operation on the original target database.
- The target environment will be available until the database copy has reached the target server. After that point, the environment will be offline until the refresh process is completed.
- The refresh will affect only the application and Financial Reporting databases.
- Documents in Azure Blob storage aren't copied from one environment to another. Therefore, attached document handling documents and templates won't be changed and will remain in their current state.
- All users except the Admin user and other internal service user accounts will be unavailable. Therefore, the Admin user can delete or obfuscate data before other users are allowed back into the system.
- The Admin user must make required configuration changes, such as reconnecting integration endpoints to specific services or URLs.
- All data management integration jobs that have recurring import and export enabled must be fully processed and stopped in the target system before the restore is started. In addition, we recommend that you select the database from the source after all recurring import and export jobs have been fully processed. In this way, you ensure that there are no orphaned files from either system in Azure storage. This step is

important because orphaned files can't be processed after the database is restored in the target environment. After the restore, the integration jobs can be resumed.

- Business events end points must be deleted and reconfigured in the environment where the database is restored to ensure the same end points are not used. This will also require the business events to be deactivated and re-activated to the new end points that were configured in the environment.
- Any user who has the Project owner or Environment manager role in LCS will have access to the SQL and machine credentials for all non-production environments.
- The databases must be hosted in the same Azure geographic region, unless the databases are Spartan-managed. Databases are Spartan-managed when you see 'spartan' as part of the fully qualified SQL server address.
- The allocated database capacity of the source environment must be less than the maximum database capacity of the target environment.

Steps to complete after a restore is done for environments that use Commerce functionality

> **IMPORTANT**
>
> Some environment-specific records are not included in automated database movement operations and require additional steps. These include the following:
>
> - Commerce self-service installer references
> - Commerce Scale Unit channel database configuration records

If you copy a database between environments, Commerce capabilities in the destination environment will not be fully functional until you perform the following additional steps.

Initialize Commerce Scale Units

If you are moving a database to a sandbox UAT or production environment, you must Initialize Commerce Scale Unit after the database movement operation is complete. The Commerce Scale Unit's association from the source environment will not copy over to the destination environment.

Synchronize Commerce self-service installers

To be able to access Commerce self-service installers in HQ, you must Synchronize self-service installers after the database movement operation is complete.

> **IMPORTANT**
>
> The Environment re-provisioning step has now been fully automated as part of database movement operations, and no longer needs to be run manually. The Environment re-provisioning tool is still available in the Asset Library and may be used in certain situations to mitigate error conditions.

To run the Environment re-provisioning tool on the destination environment, run the following steps:

1. In your project's **Asset Library**, in the **Software deployable packages** section, select **Import**.
2. From the list of shared assets, select the **Environment Reprovisioning Tool**.
3. On the **Environment details** page for your destination environment, select **Maintain** > **Apply updates**.
4. Select the **Environment Reprovisioning** tool that you uploaded earlier, and then select **Apply** to apply the package.
5. Monitor the progress of the package deployment.

For more information about how to apply a deployable package, see Create deployable packages of models. For

more information about how to manually apply a deployable package, see Install deployable packages from the command line.

Re-activate POS devices

If you use point of sale (POS) devices, after you import a database you must activate the POS devices again. Previously activated devices in the destination environment will no longer function. For more information, see Point of sale device activation.

Known issues

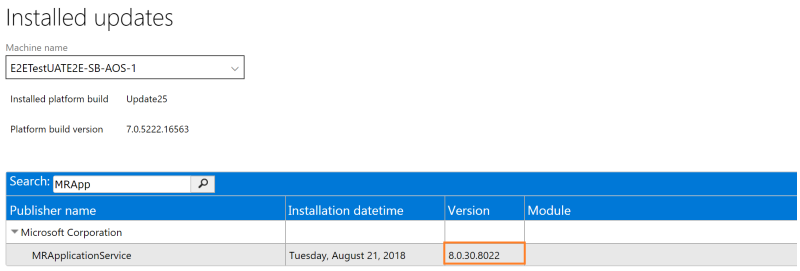
Restore is denied for environments that run Platform update 20 or earlier

The database refresh process can't currently be completed if the environment is running Platform update 20 or earlier. For more information, see the list of currently supported platform updates.

The source and target environments have incompatible versions of Financial Reporting

The database refresh process (self-service or via a service request) can't be successfully completed if the version of Financial Reporting in your target environment is earlier than the version in your source environment. To resolve this issue, update both environments so that they have the latest version of Financial Reporting.

To determine the version that is installed in your source and target environments, select **View detailed version information** on the **Environment Details** page. Then search for **MRApplicationService**, and make sure that the version in the target environment is later than or the same as the version in the source environment.



Customers that are using version 8.1 or later should follow these steps.

1. Go to the **Update** tiles for your UAT environment. Save the updates to your Project asset library.
2. Apply the package to your UAT environment.
3. Verify that the error has been resolved.

Customers that are using version 8.0 or earlier should follow these steps.

1. Review the environment history of your source environment. Specifically, look for any "Platform and application binary package" package that has been deployed to the source environment but not to the target environment.
2. Apply the binary package to your target environment.
3. Verify that the error has been resolved.

The source and target environments have incompatible application versions

The database refresh process (self-service or via service request) can't be completed if the application release of your source environment and the application release of your target environment aren't the same. Because the data upgrade process isn't run by using database movement operations such as refresh, data loss can occur.

If you're upgrading your sandbox UAT environment to a newer application version (for example, from 7.3 to 8.1), be sure to perform the database refresh action before you start the upgrade. After your sandbox environment is upgraded to the newer version, you can't restore an older production environment database to the sandbox UAT environment.

Conversely, if your production environment is newer than your target sandbox environment, you must either upgrade the target sandbox environment before the refresh, or just deallocate, delete, and redeploy the environment before you do the refresh.

The source and target are on different infrastructure (Microsoft-managed vs. self-Service)

The PITR process is not supported between Microsoft-managed and self-service environments across different regions. For example, if the production environment is Microsoft-managed and in East US2 and a PITR is needed to the sandbox environment, which is self-service and in East US, PITR is not supported. The alternative is to move the production environment to self-service or opt for a regular database refresh instead.

Point in time restore between source and target that are both on self-Service, in different regions

The PITR process is not supported between self-service environments across different regions. For example, if the production environment is in East US and a PITR is needed for the sandbox environment, which is self-service and in West Europe, PITR is not supported. The alternative is to get both the environments in the same region or opt for a regular database refresh instead.

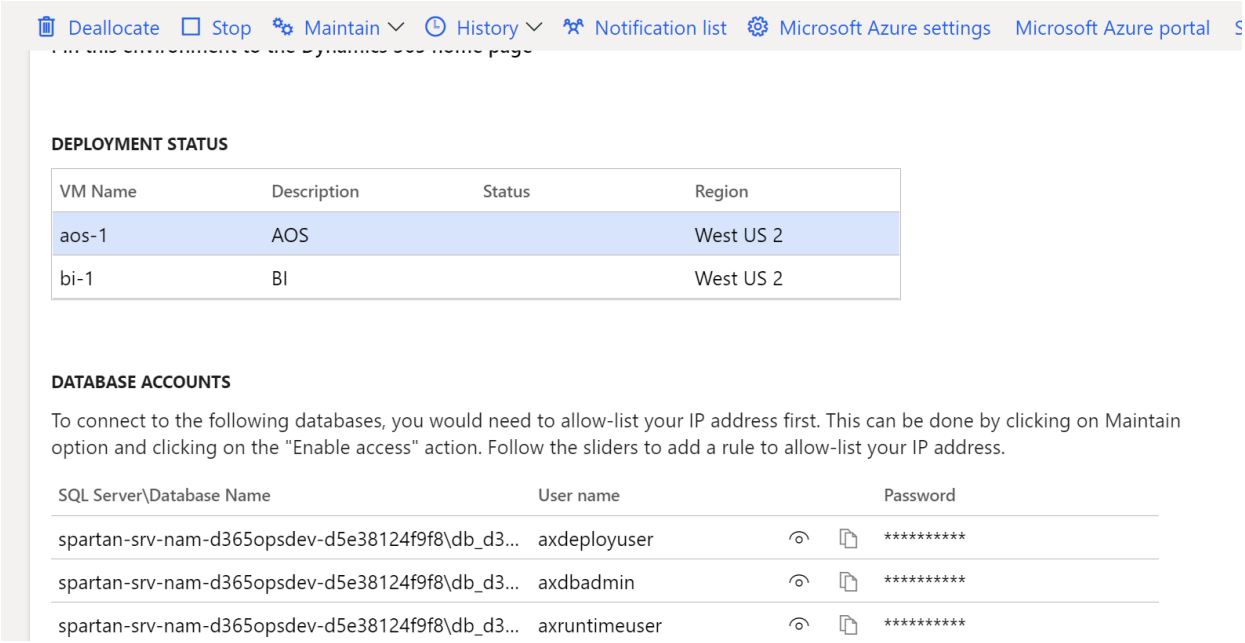
Enable just-in-time database access

2/18/2021 • 2 minutes to read • Edit Online

This topic provides the steps necessary to enable database access using a just-in-time (JIT) fashion. This is useful if access to the database is required for various troubleshooting efforts, running unplanned queries, or data upgrade problem solving. This process is available for both self-service as well as Microsoft-managed sandbox acceptance test environments. For more information about the available environment types, see Deployment overview.

Microsoft-managed environments without RDP access

If you no longer have Remote Desktop Protocol (RDP) access to your sandbox, you can add your IP address to the allow-list in a self-service manner from Lifecycle Services (LCS). When RDP is removed from an environment, the machine credentials section of the environment details page is removed. This leaves just the database accounts section, as shown in the following screenshot.



From the environment details page for your sandbox environment, select **Maintain** > **Enable access**, and then in the dialog box, add the IP address of your source environment. This entry will expire, with the expire date shown alongside the IP address you entered. It also will be lost after the database is replaced by a database movement operation, such as database refresh or database import.

You can now use tools like SQL Server Management Studio (SSMS) to connect to the database, using the accounts from LCS and the IP address that you enabled. Note that LCS shows the server and database in the following format: **serverName\databaseName**. To connect in SSMS, you will need to append the domain name suffix, such as **serverName.database.windows.net** if you are in Azure public cloud. On the **Options** tab in the SSMS connection window, you will also need to explicitly enter the databaseName value in the **Database** field to successfully connect.

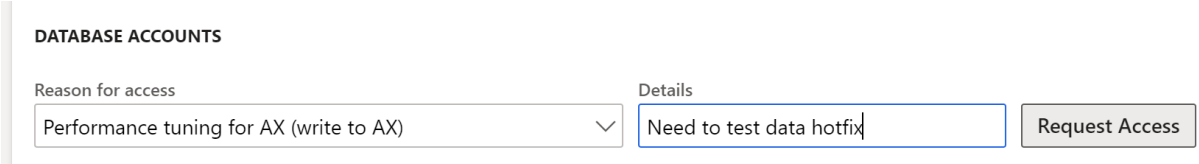
Self-service environments

The self-service environment type has never had Remote Desktop Protocol (RDP) access or static database accounts. However, it is still possible to access the database.

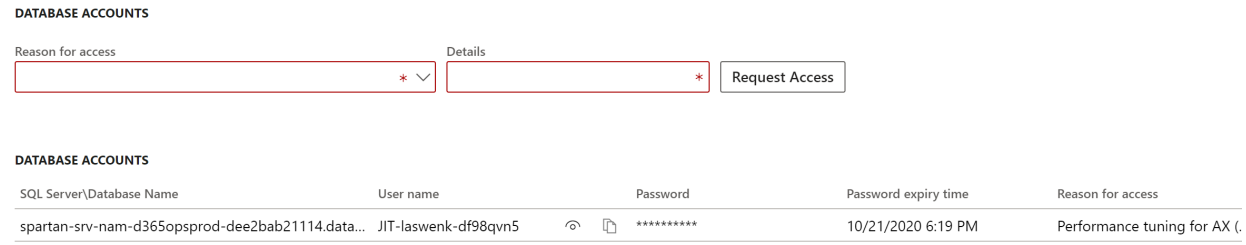
From the environment details page for your sandbox environment, select **Maintain** > **Enable access**, and then

in the dialog box, add the IP address of your source environment. This entry will not expire, however it will be lost after the database is replaced by a database movement operation, such as database refresh or database import.

You also need to enter which type of access you require in the **Database Accounts** section. The available options include read or read-write access. Enter a short reason description and then select **Request access**.



When the page is refreshed, the database account will be shown with its expiry time.



You can now use tools like SQL Server Management Studio (SSMS) to connect to the database, using the accounts from LCS and the IP address you enabled.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Refresh for training purposes

2/18/2021 • 4 minutes to read • Edit Online

Database movement operations are a suite of self-service actions that can be used as part of data application lifecycle management (DataALM). This tutorial shows how to use the refresh database operation in a training scenario.

In this tutorial, you will learn how to:

- Prepare the target environment.
- Run the refresh.
- Reconfigure the target environment.
- Enable selected users.

As an example of this scenario, a customer who has already gone live with the application wants to load a recent copy of production transactions into the user acceptance testing (UAT) environment. In this way, the customer can support training of new employees and evaluate configuration changes without affecting the live environment.

Prerequisites

To do a refresh database operation, your production environment must be deployed, or you must have a minimum of two standard UAT environments.

Notify users about the pending downtime

Before you start the bulk of the work, notify users of the target environment that the environment will be offline for a period. You can notify users either manually via Microsoft Dynamics Lifecycle Services (LCS) or programmatically by using RESTful application programming interface (API) calls.

Manually send a broadcast message

To notify users manually via LCS, follow these steps.

1. In LCS, open the **Environment details** page for the target environment.
2. Select **Maintain** > **Message online users**.
3. Select **Broadcast a new message for downtime**.
4. Select the valid from/valid to times in your local time zone.
5. Select **Post**.

Programmatically send a broadcast message

The following sample code can be used as shown in a console application, or it can be modified to make it work with other services that can be called on demand, such as Microsoft Azure Functions. Before this sample code will work, you must set up your application registration as described in Service endpoints overview.

```
[Serializable]
public class SysAddBroadcastMessageDataContract
{
    public SysAddBroadcastMessageRequest request { get; set; }
}
[Serializable]
public class SysAddBroadcastMessageRequest
{
    public DateTime FromDateTime { get; set; }
    public DateTime ToDateTime { get; set; }
}
public class Program
{
    public static AuthenticationResult getResult(AuthenticationContext ctx, string url)
    {
        return ctx.AcquireTokenAsync(url, "YOUR_APP_REGISTRATION_ID", new Uri("YOUR_REPLY_URL"), new
PlatformParameters(PromptBehavior.Always)).Result;
    }
    static void Main(string[] args)
    {
        SysAddBroadcastMessageRequest request = new SysAddBroadcastMessageRequest();
        request.FromDateTime = DateTime.UtcNow;
        request.ToDateTime = DateTime.UtcNow.AddHours(12);

        SysAddBroadcastMessageDataContract dc = new SysAddBroadcastMessageDataContract();
        dc.request = request;

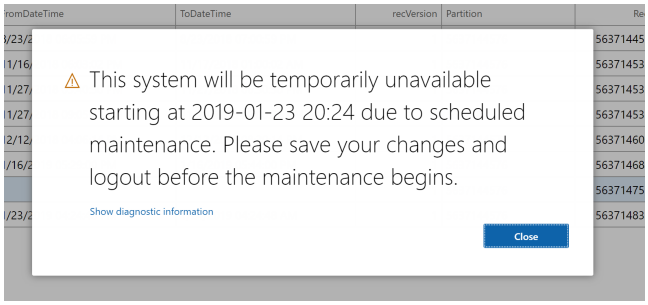
        AuthenticationContext ctx = new
AuthenticationContext("https://login.microsoftonline.com/YOUR_TENANT.COM");
        AuthenticationResult res = getResult(ctx,
"https://YOUR_SANDBOX_UAT.sandbox.operations.dynamics.com");

        HttpClient restfulCli = new HttpClient();
        restfulCli.DefaultRequestHeaders.Clear();
        restfulCli.BaseAddress = new Uri("https://YOUR_SANDBOX_UAT.sandbox.operations.dynamics.com/");
        restfulCli.DefaultRequestHeaders.Add("Authorization", res.CreateAuthorizationHeader());
        restfulCli.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue("application/json"));

        HttpRequestMessage requestMsg = new HttpRequestMessage(HttpMethod.Post,
string.Format("api/services/SysBroadcastMessageServices/SysBroadcastMessageService/AddMessage"));
        requestMsg.Content = new StringContent(JsonConvert.SerializeObject(dc));

        HttpResponseMessage responseMsg = restfulCli.SendAsync(requestMsg).Result;
        if(responseMsg.IsSuccessStatusCode)
        {
            Console.WriteLine("Wow I just notified the users programmatically!");
        }
    }
```

Both approaches, manual via LCS and programmatic via RESTful API calls, will show users that a period of downtime is pending.



Begin the refresh

Depending on the size of your source environment, it might make sense to begin the refresh process immediately. The larger the source database, the longer it will take to copy to your target Azure SQL Database instance. While the copy is in progress, the target environment will still be online. The downtime will start after the copy is completed.

This process can be done manually via LCS. For the latest steps, see Refresh database. In a future release of LCS, you will also be able to trigger this process via a RESTful API.

Reconfigure environment specific settings

After the refresh is completed, use the **Sign off** button in LCS to close out of the operation. You can then start to configure the environment-specific settings.

First, sign in to the environment by using the admin account that can be found on the **Environment details** page in LCS. Here are typical areas of reconfiguration. You might require additional reconfiguration, based on your setup and the independent software vendor (ISV) solutions that are installed.

- **System administration** > **Setup** > **Batch groups:** Add the various Application Object Server (AOS) instances to the batch server groups that you require.
- **System administration** > **Setup** > **Entity Store:** Refresh the various entities that you require for Microsoft Power BI reporting.
- **System administration** > **Setup** > **System parameters:** Reconnect the environment to the LCS Help configuration for task guides.
- **System administration** > **Setup** > **Email** > **Email parameters:** Enter the Simple Mail Transfer Protocol (SMTP) settings if you use email in your UAT environment.
- **System administration** > **Inquiries** > **Batch jobs:** Select the jobs that you want to run in your UAT environment, and update the status to **Waiting**.

To complete this reconfiguration more quickly, we recommend that you build a custom web service endpoint that can be called on demand after the refresh is completed. An example of this type of web service will be added in a future update of this topic.

Open the environment to users

When the system is configured as you require, you can enable selected users to let them access the environment. By default, all users except the admin and Microsoft service accounts are disabled.

Go to **System administration** > **Users** > **Users**, and enable the users that should have access to the UAT environment. If many users must be enabled, you can complete this task more quickly by using the Microsoft Excel Add-In.

Debug a copy of the production database

2/18/2021 • 4 minutes to read • Edit Online

Database movement operations are a suite of self-service actions that can be used as part of data application lifecycle management (DataALM). This tutorial shows how to debug specific data and transactions from a recent copy of production data.

In this tutorial, you will learn how to:

- Refresh the user acceptance testing (UAT) environment.
- Add the IP address of your developer environment to an approved list ("safe list").
- Update your developer environment so that it connects to the UAT database.
- Set a breakpoint, and start to debug the data.

As an example of this scenario, a customer who has already gone live wants to debug a recent copy of production transactions from the development environment. In this way, the customer will be able to debug specific transactions that are stuck, or develop new features and reports by using realistic datasets.

Prerequisites

To do a refresh operation, you must have your production environment deployed, or you must have a minimum of two standard UAT environments. To complete this tutorial, you must have a developer environment deployed.

> **IMPORTANT**
>
> For debugging, we highly recommend that you use a DevTest environment that runs the same code and business logic that are available in your UAT environment. If you use multiple branches in version control, we recommend that the DevTest environment that is used to debug recent UAT or production transactions be connected to the same branch that you use to build packages for UAT and, later, for production. In this way, you don't have to run a database synchronization between your DevTest environment and UAT database, because the schema will be compatible. Historically, this environment is known as a Hotfix/Support environment, because it's outside your usual code promotion path.

Refresh the UAT environment

This refresh operation overwrites the UAT environment with the latest copy of the production database. To complete this step, follow the instructions in Refresh for training purposes.

Enable database access

By default, all Sandbox Standard Acceptance Test environments use Microsoft Azure SQL Database as their database platform. The databases for these environments are protected by firewalls that restrict access to the Application Object Server (AOS) with which it was originally deployed.

To connect to the database, follow the instructions in Enable just-in-time access.

> **NOTE**
>
> Every time that a refresh is done, the firewall safe list is reset. You must add your DevTest environments back to this database when they are required in the future.

Update a OneBox DevTest environment to connect to the UAT database

In your developer environment, you must now update the web.config file to change the database connection. This step will let you run your local code and binaries that are configured against the database from UAT.

On your Services drive, go to the **AoSService\WebRoot** directory. (Typically, the Services drive is drive J or K.) Find the file that is named web.config, and *make a backup of it*. Then open the **web.config** file in Notepad or another editor, and find the following configurations:

- DataAccess.Database
- DataAccess.DBServer
- DataAccess.SqlPwd
- DataAccess.SqlUser

Update these configurations so that they use the values from the environment details page for the UAT environment in LCS.

```
<add key="DataAccess.Database" value="<example_axdb_fromAzure>" />
<add key="DataAccess.DbServer" value="<example_axdb_server.database.windows.net>" />
<add key="DataAccess.SqlPwd" value="<axdbadmin_password_from_LCS>" />
<add key="DataAccess.SqlUser" value="axdbadmin" />
```

Save the file. If you're operating in a cloud-hosted environment, run IISRESET. If you're on a Microsoft-managed developer machine and have limited permissions, make sure that Microsoft Visual Studio is closed.

Finally, open a web browser, go to the URL of your DevTest environment, and verify that you're pulling data from the UAT database.

Debug transactions in the DevTest environment

Now that your environment is correctly reconfigured, you can open Visual Studio and set a breakpoint in the application code that best meets your needs. Note that users in the UAT environment aren't affected while you debug in the DevTest environment.

Debugging batch

For scenarios where you need to debug batch jobs, on the debugging DevTest machine, you may need to restart the batch service before it will show up as an option to attach the debugger from Visual Studio. In addition, it may also be helpful to isolate this DevTest machine in to its own batch group to ensure that any jobs that you want to debug will run on the DevTest machine.

Best practices

Here are some common best practices that will help guarantee that your debugging experience is quick, reliable, and doesn't disrupt other users in your organization:

- Make sure that the version of the code and binaries in the DevTest environment exactly match the version in the UAT environment. Connect the DevTest environment to the same branch that you build packages for deployment from. Alternatively, connect it to a "HotfixSupport" branch that is kept up to date with the latest customizations that are released.
- Don't run a database synchronization from Visual Studio. Otherwise, you will affect the availability of the schema in the UAT database and might affect users in the UAT environment.
- For the best experience, use a developer environment that was deployed in the same datacenter as the UAT environment.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Export a copy of the standard user acceptance testing (UAT) database

2/18/2021 • 12 minutes to read • Edit Online

Database movement operations are a suite of self-service actions that can be used as part of data application lifecycle management (DataALM). This tutorial shows how to export all the data and transactions from a sandbox standard user acceptance testing (UAT) environment.

In this tutorial, you will learn how to:

- Refresh the UAT environment.
- Run the export to the Asset library in Microsoft Dynamics Lifecycle Services (LCS).
- Download the database backup.
- Import the database, and prepare it so that it can be used in a developer environment.

As an example of this scenario, a customer who has already gone live wants to load a recent copy of production transactions into the development environment. In this way, the customer will be able to debug specific transactions, or develop new features and reports by using realistic datasets.

> **IMPORTANT**
>
> Database copy to a build environment is not supported. Learn more about build environments.

Known limitations

Because of recent restrictions by the Microsoft Azure SQL Database platform, we don't recommend that you export your database if it's larger than 200 gigabytes (GB). If you must export a larger database, we recommend that you use the legacy documentation until SQL Database can support larger exports. Note that this recommendation applies to export operations, not refresh operations. Refresh operations can support databases that are up to 4 terabytes (TB) in size.

Prerequisites

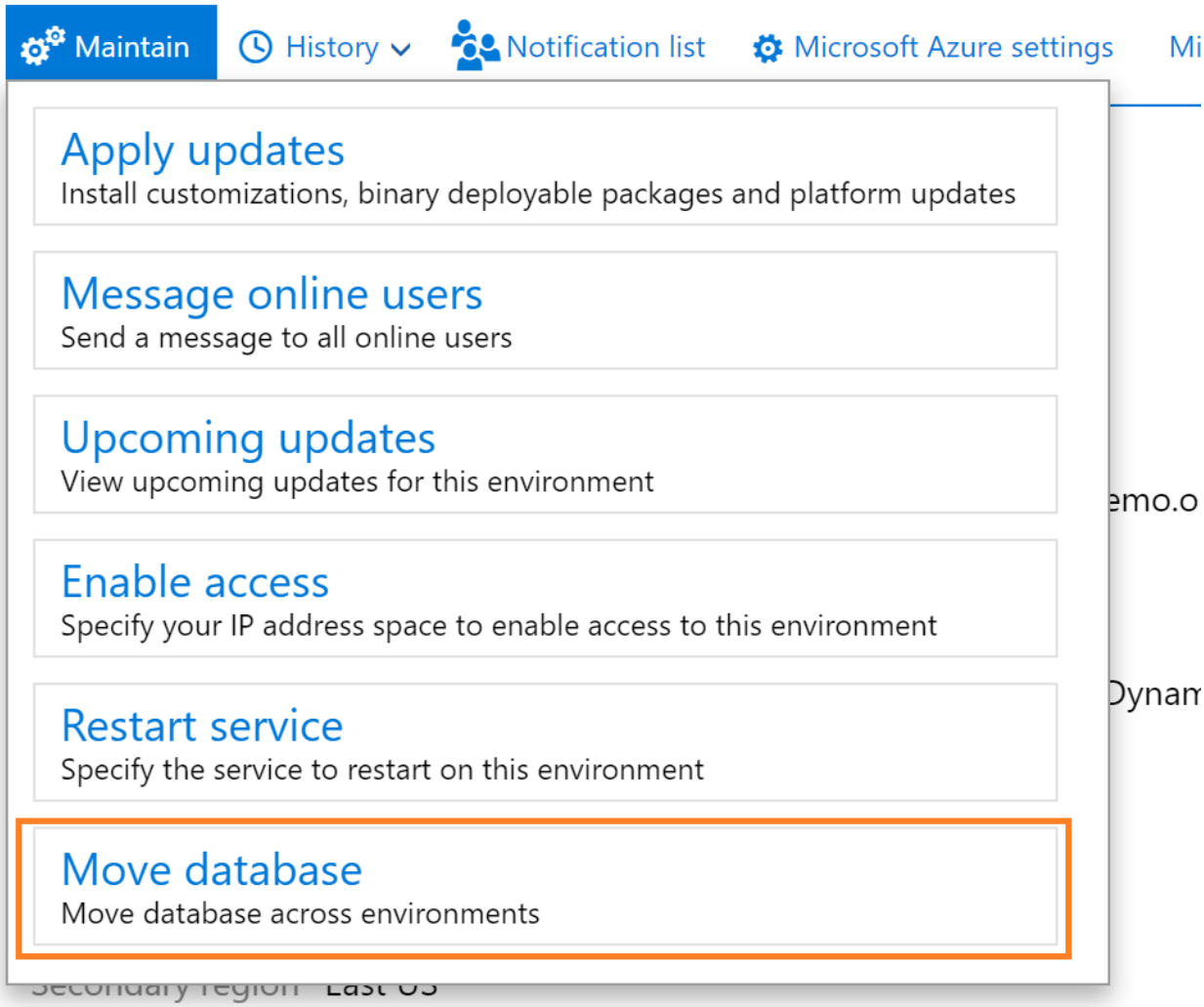
To do a refresh operation, you must have your production environment deployed, or you must have a minimum of two standard UAT environments. To complete this tutorial, you must have a developer environment deployed.

Refresh the UAT environment

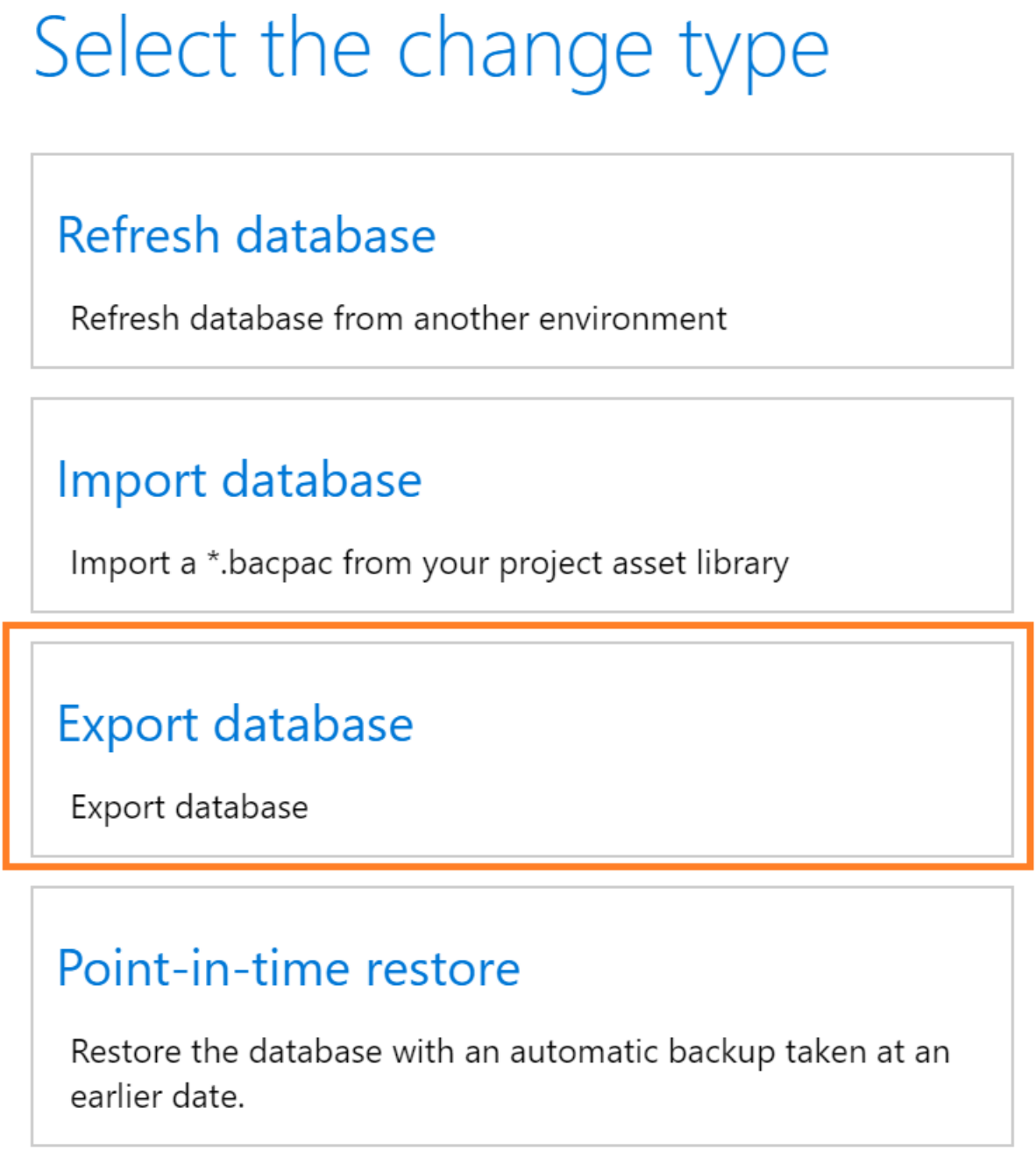
This refresh operation overwrites the UAT environment with the latest copy of the production database. To complete this step, follow the instructions in Refresh for training purposes.

Back up to the Asset Library

From your sandbox **Environment Details** page, click the **Maintain** menu, and then select **Move database**.

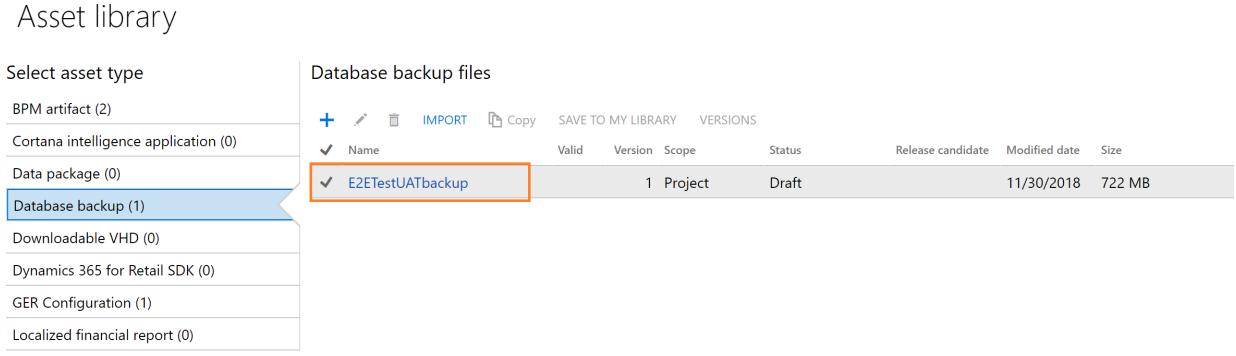


A slider pane will open on the page where you can use the **Export database** action.



The environment will be unavailable for other servicing operations, such as Sandbox refresh or package deployment during this time. The source environment will be usable from a Dynamics user perspective.

After the export operation completes successfully, sign out of the servicing operation on your **Environment details** page. You can then see the asset in your **Asset Library** in the **Database backups** section.



The **.bacpac** files are stored here and can be manually downloaded to your Tier 1 developer environments for import. In the future, Microsoft will provide APIs to trigger the export action, as well as list the available backup files in your asset library. This includes the secured URL for automatically downloading a backup asset file or copying it directly to your secure blob storage using Microsoft Azure Storage SDKs.

Import the database

After you've downloaded a database backup (.bacpac) file, you can begin the manual import operation on your Tier 1 environment. When you import the database, we recommend that you follow these guidelines:

- Keep a copy of the existing AxDB database, so that you can revert to it later if you must.
- Import the new database under a new name, such as **AxDB_fromProd**.

To ensure the best performance, copy the *.bacpac file to the local computer that you're importing from. Download sqlpackage .NET Core for Windows from Get sqlpackage .NET Core for Windows. Open a **Command Prompt** window, and run the following commands from the sqlpackage .NET Core folder.

```
SqlPackage.exe /a:import /sf:D:\Exportedbacpac\my.bacpac /tsn:localhost /tdn:<target database name>
/p:CommandTimeout=1200
```

Here is an explanation of the parameters:

- **tsn (target server name)** – The name of the Microsoft SQL Server instance to import into.
- **tdn (target database name)** – The name of the database to import into. The database should **not** already exist.
- **sf (source file)** – The path and name of the file to import from.

> **NOTE**
> During import, the user name and password aren't required. By default, SQL Server uses Microsoft Windows authentication for the user who is currently signed in.

Update the database

Run the following SQL script against the imported database. This script adds back the users that you deleted from the source database and correctly links them to the SQL logins for this SQL Server instance. The script also turns change tracking back on. Remember to edit the final **ALTER DATABASE** statement so that it uses the name of your database.

```
CREATE USER axdeployuser FROM LOGIN axdeployuser
EXEC sp_addrolemember 'db_owner', 'axdeployuser'

CREATE USER axdbadmin FROM LOGIN axdbadmin
EXEC sp_addrolemember 'db_owner', 'axdbadmin'

CREATE USER axmrruntimeuser FROM LOGIN axmrruntimeuser
EXEC sp_addrolemember 'db_datareader', 'axmrruntimeuser'
EXEC sp_addrolemember 'db_datawriter', 'axmrruntimeuser'

CREATE USER axretaildatasyncuser FROM LOGIN axretaildatasyncuser
EXEC sp_addrolemember 'DataSyncUsersRole', 'axretaildatasyncuser'

CREATE USER axretailruntimeuser FROM LOGIN axretailruntimeuser
EXEC sp_addrolemember 'UsersRole', 'axretailruntimeuser'
EXEC sp_addrolemember 'ReportUsersRole', 'axretailruntimeuser'

CREATE USER axdeployextuser FROM LOGIN axdeployextuser
EXEC sp_addrolemember 'DeployExtensibilityRole', 'axdeployextuser'

CREATE USER [NT AUTHORITY\NETWORK SERVICE] FROM LOGIN [NT AUTHORITY\NETWORK SERVICE]
EXEC sp_addrolemember 'db_owner', 'NT AUTHORITY\NETWORK SERVICE'

UPDATE T1
SET T1.storageproviderid = 0
    , T1.accessinformation = ''
    , T1.modifiedby = 'Admin'
    , T1.modifieddatetime = getdate()
FROM docuvalue T1
WHERE T1.storageproviderid = 1 --Azure storage

DROP PROCEDURE IF EXISTS SP_ConfigureTablesForChangeTracking
DROP PROCEDURE IF EXISTS SP_ConfigureTablesForChangeTracking_V2
GO
-- Begin Refresh Retail FullText Catalogs
DECLARE @RFTXNAME NVARCHAR(MAX);
DECLARE @RFTXSQL NVARCHAR(MAX);
DECLARE retail_ftx CURSOR FOR
SELECT OBJECT_SCHEMA_NAME(object_id) + '.' + OBJECT_NAME(object_id) fullname FROM SYS.FULLTEXT_INDEXES
    WHERE FULLTEXT_CATALOG_ID = (SELECT TOP 1 FULLTEXT_CATALOG_ID FROM SYS.FULLTEXT_CATALOGS WHERE NAME =
'COMMERCEFULLTEXTCATALOG');
OPEN retail_ftx;
FETCH NEXT FROM retail_ftx INTO @RFTXNAME;

BEGIN TRY
    WHILE @@FETCH_STATUS = 0
    BEGIN
        PRINT 'Refreshing Full Text Index ' + @RFTXNAME;
        EXEC SP_FULLTEXT_TABLE @RFTXNAME, 'activate';
        SET @RFTXSQL = 'ALTER FULLTEXT INDEX ON ' + @RFTXNAME + ' START FULL POPULATION';
        EXEC SP_EXECUTESQL @RFTXSQL;
        FETCH NEXT FROM retail_ftx INTO @RFTXNAME;
    END
END TRY
BEGIN CATCH
    PRINT error_message()
END CATCH

CLOSE retail_ftx;
DEALLOCATE retail_ftx;
-- End Refresh Retail FullText Catalogs
```

Turn on change tracking

If change tracking was turned on in the source database, be sure to turn it on in the newly provisioned database in the target environment. To turn on change tracking, use the **ALTER DATABASE** command.

```
ALTER DATABASE [your database name] SET CHANGE_TRACKING = ON (CHANGE_RETENTION = 6 DAYS, AUTO_CLEANUP = ON);
```

To help guarantee that the current version of the store procedure that is related to change tracking is used in the new database, you must turn change tracking on or off for a data entity in Data management. You can choose any entity. This step is required in order to trigger a refresh of the store procedure.

Start to use the new database

To switch environments and use the new database, first stop the following services:

- World Wide Web Publishing Service
- Microsoft Dynamics 365 Unified Operations: Batch Management Service
- Management Reporter 2012 Process Service

After these services have been stopped, rename the AxDB database **AxDB_orig**, rename your newly imported database **AxDB**, and then restart the three services.

To switch back to the original database, reverse this process. In other words, stop the services, rename the databases, and then restart the services.

Post steps for Commerce environments

If you are using Commerce channels, when importing a database to a developer environment, which was originally exported from a self-service sandbox, the following additional steps must be performed on the destination developer environment. Without completing these steps, Commerce channels will not function.

1. To restore Commerce channels functionality, apply the latest Microsoft service update or quality update, which will create the channel database.
2. To restore any previously deployed channel database extensions, re-apply the corresponding Retail self-service deployable package.

Reprovision the target environment

> **IMPORTANT**
>
> Some environment-specific records are not included in automated database movement operations and require additional steps. These include the following:
>
> - Commerce self-service installer references
> - Commerce Scale Unit channel database configuration records

If you copy a database between environments, Commerce capabilities in the destination environment will not be fully functional until you perform the following additional steps.

Initialize Commerce Scale Units

If you are moving a database to a sandbox UAT or production environment, you must Initialize Commerce Scale Unit after the database movement operation is complete. The Commerce Scale Unit's association from the source environment will not copy over to the destination environment.

Synchronize Commerce self-service installers

To be able to access Commerce self-service installers in HQ, you must Synchronize self-service installers after the database movement operation is complete.

IMPORTANT

The Environment re-provisioning step has now been fully automated as part of database movement operations, and no longer needs to be run manually. The Environment re-provisioning tool is still available in the Asset Library and may be used in certain situations to mitigate error conditions.

To run the Environment re-provisioning tool on the destination environment, run the following steps:

1. In your project's **Asset Library**, in the **Software deployable packages** section, select **Import**.
2. From the list of shared assets, select the **Environment Reprovisioning Tool**.
3. On the **Environment details** page for your destination environment, select **Maintain** > **Apply updates**.
4. Select the **Environment Reprovisioning** tool that you uploaded earlier, and then select **Apply** to apply the package.
5. Monitor the progress of the package deployment.

For more information about how to apply a deployable package, see Create deployable packages of models. For more information about how to manually apply a deployable package, see Install deployable packages from the command line.

Re-activate POS devices

If you use point of sale (POS) devices, after you import a database you must activate the POS devices again. Previously activated devices in the destination environment will no longer function. For more information, see Point of sale device activation.

Reset the Financial Reporting database

If you're using Financial Reporting, you must reset the Financial Reporting database by following the steps in Reset the Financial reporting data mart. (Financial Reporting was previously named Management Reporter.)

Reenter data from encrypted and environment-specific fields in the target database

In the client, enter the values that you documented for the encrypted and environment-specific fields. The following fields are affected. The field names are given in *Table.Field* format.

FIELD NAME	WHERE TO SET THE VALUE
CreditCardAccountSetup.SecureMerchantProperties	Select **Accounts receivable** > **Payments setup** > **Payment services**.
ExchangeRateProviderConfigurationDetails.Value	Select **General ledger** > **Currencies** > **Configure exchange rate providers**.
FiscalEstablishment_BR.ConsumerEFDocCsc	Select **Organization administration** > **Fiscal establishments** > **Fiscal establishments**.
FiscalEstablishmentStaging.CSC	This field is used by the Data Import/Export Framework (DIXF).
HcmPersonIdentificationNumber.PersonIdentificationNumber	Select **Human resources** > **Workers** > **Workers**. On the **Worker** tab, in the **Personal information** group, select **Identification numbers**.

FIELD NAME	WHERE TO SET THE VALUE
HcmWorkerActionHire.PersonIdentificationNumber	This field has been obsolete since Microsoft Dynamics AX 7.0 (February 2016). It was previously in the **All worker actions** form (**Human resources** > **Workers** > **Actions** > **All worker actions**).
SysEmailSMTPPassword.Password	Select **System administration** > **Email** > **Email parameters**.
SysOAuthUserTokens.EncryptedAccessToken	This field is used internally by Application Object Server (AOS). It can be ignored.
SysOAuthUserTokens.EncryptedRefreshToken	This field is used internally by AOS. It can be ignored.

Community tools

Are you looking for more tools to help you import backup files into your developer environments? Here are some other sources of information:

- D365fo.Tools provides many valuable tools that have been created by the community.
- Community-provided open source projects on GitHub.

Known issues

The export database is in a "Preparation failed" state

If the automation from LCS times out, the state of the export database is changed to **Preparation failed**. The export operation to export to the Asset library is still running in SQL Database. To resolve this issue, you can use the **Resume** button to reconnect the process with SQL Database. The process should then be successfully completed.

The export database takes a very long time

The Azure SQL team recently announced that the Import/Export application programming interface (API) that LCS uses has variable execution times for any database that is over 200 GB in size. If you encounter this issue, you can either connect your DevTest environment directly to the UAT database or follow the legacy documentation. We don't recommend that you export databases for backup purposes, because the point-in-time restore functionality is available and included with your environment.

The Lifecycle Services team is working directly with the Azure SQL team to increase the performance of the Import/Export API and will make improvements in upcoming releases of LCS.

I can't download Management Studio installation files

When you try to download the Microsoft SQL Server Management Studio installer, you might receive the following error message:

> Your current security settings do not allow this file to be downloaded.

To work around this issue, follow these steps to enable file downloads.

1. In your web browser, open **Internet options**.
2. On the **Security** tab, select the **Internet** zone, and then select **Custom level**.
3. Scroll to **Downloads**, and then, under **File download**, select the **Enable** option.

Database synchronization fails

When you sync the database against the newly imported database from Microsoft Visual Studio, the synchronization might fail, and you might receive the following error message:

> Failed to open SQL connection syncengine.exe exited with code -1.

In this case, the following message is also logged under event ID 140 in the Windows application log:

> Object Server Database Synchronizer: The internal system table version number stored in the database is higher than the version supported by the kernel (141/138). Use a newer Microsoft Dynamics kernel, or start Microsoft Dynamics using the -REPAIR command line parameter to enforce synchronization.

This issue can occur when the platform build number of the current environment is lower than the platform build number of the source environment. To resolve the issue, follow one of these steps, depending on your circumstances:

- Use the **Updates** tiles on the environment page in LCS to upgrade the platform in the current environment so that it matches the platform in the source environment.
- Run the following query to adjust the expected version in the database.

```
UPDATE SQLSYSTEMVARIABLES

SET VALUE = 138

WHERE PARM = 'SYSTABVERSION'
```

> **NOTE**
>
> The value **138** in this query is taken from the event log message, where version 138 was expected in this particular environment.

Performance

The following guidelines can help you achieve optimal performance:

- Always import the .bacpac file locally on the computer that runs the SQL Server instance. Don't import it from Management Studio on a remote machine.
- In a one-box environment that is hosted in Azure, put the .bacpac file on drive D when you import it. (A one-box environment is also known as a Tier 1 environment.) For more information about the temporary drive on Azure virtual machines (VMs), see the Understanding the temporary drive on Windows Azure Virtual Machines blog post.
- Grant the account that runs the SQL Server Windows service Instance File Initialization rights. In this way, you can help improve the speed of the import process and the speed of a restore from a *.bak file. For a developer environment, you can easily make sure that the account that runs the SQL Server service has these rights by setting SQL Server to run as the axlocaladmin account.

Golden configuration promotion

2/18/2021 • 11 minutes to read • Edit Online

Database movement operations are a suite of self-service actions that can be used as part of data application lifecycle management (DataALM). "Golden configuration" refers to a common practice among customers and partners in the Microsoft Dynamics ecosystem, where a developer environment is used as a configuration store. In this way, implementation projects can store finalized global and company-specific settings in a database that can later become a baseline for Conference Room Pilots, mock go-lives, and go-lives. This tutorial shows how to prepare a golden configuration database and hydrate a target user acceptance testing (UAT) environment.

In this tutorial, you will learn how to:

- Prepare the golden configuration database for Microsoft Azure SQL Database.
- Run the import to the target UAT environment.
- Copy the UAT environment into a production environment.

As an example of this scenario, a customer who hasn't gone live is instead preparing for a Conference Room Pilot, mock go-live, or go-live. This scenario supports promoting a baseline golden database from a developer environment to a UAT environment and eventually to production.

Prerequisites

To complete this tutorial, you must have a developer environment that is deployed with a database that is curated as a golden configuration database. You must also have at least one standard UAT environment deployed and, optionally, a production environment.

The developer environment must run the same *application version* as the target UAT environment. In addition, the *platform version* of the developer environment must be earlier than or the same as the platform version in the target UAT environment.

Before you begin

Supported SQL Server collation

The only supported collation databases in the cloud is **SQL_Latin1_General_CP1_CI_AS**. Make sure that your Microsoft SQL Server and database collations in development environments are set to this value. Also make sure that any configuration environments that are published to sandbox have this collation.

Document the values of encrypted fields

Encrypted and environment-specific values can't be imported into a new environment. After you've completed the import, you must reenter some data from your source environment in your target environment.

Because of a technical limitation that is related to the certificate that is used for data encryption, values that are stored in encrypted fields in a database will be unreadable after that database is imported into a new environment. Therefore, after an import, you must manually delete and reenter values that are stored in encrypted fields. New values that are entered in encrypted fields after an import will be readable. The following fields are affected. The field names are given in *Table.Field* format.

FIELD NAME	WHERE TO SET THE VALUE

FIELD NAME	WHERE TO SET THE VALUE
CreditCardAccountSetup.SecureMerchantProperties	Select **Accounts receivable** > **Payments setup** > **Payment services**.
ExchangeRateProviderConfigurationDetails.Value	Select **General ledger** > **Currencies** > **Configure exchange rate providers**.
FiscalEstablishment_BR.ConsumerEFDocCsc	Select **Organization administration** > **Fiscal establishments** > **Fiscal establishments**.
FiscalEstablishmentStaging.CSC	This field is used by the Data Import/Export Framework (DIXF).
HcmPersonIdentificationNumber.PersonIdentificationNumber	Select **Human resources** > **Workers** > **Workers**. On the **Worker** tab, in the **Personal information** group, select **Identification numbers**.
HcmWorkerActionHire.PersonIdentificationNumber	This field has been obsolete since Microsoft Dynamics AX 7.0 (February 2016). It previously appeared on the **All worker actions** page (**Human resources** > **Workers** > **Actions** > **All worker actions**).
SysEmailSMPTPassword.Password	Select **System administration** > **Email** > **Email parameters**.
SysOAuthUserTokens.EncryptedAccessToken	This field is used internally by Application Object Server (AOS). It can be ignored.
SysOAuthUserTokens.EncryptedRefreshToken	This field is used internally by AOS. It can be ignored.

If you're running Commerce components, document encrypted and environment-specific values

The values on the following pages are either environment-specific or encrypted in the database. Therefore, all the imported values will be incorrect.

- Payments services (**Accounts receivable** > **Payments setup** > **Payments services**)
- Hardware profiles (**Retail and commerce** > **Channel setup** > **POS setup** > **POS profiles** > **Hardware profiles**)

Create a copy of the source database

Back up the source database using SSMS. Right-click the source database, and select **Tasks** > **Backup option**. After this is completed, right-click the **Databases** folder in the SSMS navigation window, and select **Restore database**. Choose the backup that you just made, but give the target database a new name such as AXDB_CopyForExport.

Prepare the database

Run the following script against the AxDB_CopyForExport database that you created in the previous section. This script makes the following changes:

- Set the **SysGlobalConfiguration** flag to inform the application that the database is Azure-based.
- Remove a reference to tempDB in the XU_DisableEnableNonClusteredIndexes procedure. References to tempDB aren't allowed in an Azure SQL database. The database synchronization process will re-create the reference later.

- Drop users, because Microsoft Windows users are forbidden in Azure SQL databases. Other users must be re-created later, so that they're correctly linked to the appropriate sign-in on the target server.
- Clear encrypted hardware profile merchant properties.

A successful export and import of the database requires all these changes.

```
update sysglobalconfiguration
set value = 'SQLAZURE'
where name = 'BACKENDDB'

update sysglobalconfiguration
set value = 1
where name = 'TEMPTABLEINAXDB'

drop procedure if exists XU_DisableEnableNonClusteredIndexes
drop procedure if exists SP_ConfigureTablesForChangeTracking
drop procedure if exists SP_ConfigureTablesForChangeTracking_V2
drop schema [NT AUTHORITY\NETWORK SERVICE]
drop user [NT AUTHORITY\NETWORK SERVICE]
drop user axdbadmin
drop user axdeployuser
drop user axmrruntimeuser
drop user axretaildatasyncuser
drop user axretailruntimeuser
drop user axdeployextuser

--Tidy up the batch server config from the previous environment
DELETE FROM SYSSERVERCONFIG

--Tidy up server sessions from the previous environment
DELETE FROM SYSSERVERSESSIONS

--Tidy up printers from the previous environment
DELETE FROM SYSCORPNETPRINTERS

--Tidy up client sessions from the previous environment
DELETE FROM SYSCLIENTSESSIONS

--Tidy up batch sessions from the previous environment
DELETE FROM BATCHSERVERCONFIG

--Tidy up batch server to batch group relation table
DELETE FROM BATCHSERVERGROUP

-- Clear encrypted hardware profile merchant properties
update dbo.RETAILHARDWAREPROFILE set SECUREMERCHANTPROPERTIES = null where SECUREMERCHANTPROPERTIES is not
null
```

Export the database from SQL Server

Open a **Command Prompt** window, and run the following commands.

> **IMPORTANT**
>
> The 140 folder reflects the current version. You must use the version that is available in your sandbox environment. Therefore, you might have to install the latest version of Microsoft SQL Server Management Studio in your development environment.

```
cd C:\Program Files (x86)\Microsoft SQL Server\140\DAC\bin\
SqlPackage.exe /a:export /ssn:localhost /sdn:<database to export> /tf:D:\Exportedbacpac\my.bacpac
/p:CommandTimeout=1200 /p:VerifyFullTextDocumentTypesSupported=false
```

Here is an explanation of the parameters:

- **ssn (source server name)** – The name of the SQL Server to export from. For the purposes of this topic, the name should always be **localhost**.
- **sdn (source database name)** – The name of the database to export.
- **tf (target file)** – The path and name of the file to export to. The folder should already exist, but the export process will create the file.

Import the database

Upload the .bacpac file that was created in the previous step to the **Database backup** section in your LCS project's Asset Library. Then begin the import. The target UAT environment's databases will be overwritten by the golden configuration database.

> **NOTE**
> Certain elements are not copied as part of the import database step. In the golden configuration scenario, this would impact things such as Email Addresses and Print Management setup. These settings ideally should be populated as part of the master data migration in the steps below, and should not be part of the golden configuration database.

To import a database that is prepared from a developer environment to a standard user acceptance test (UAT), or a database previously exported from a UAT environment, follow the steps outlined below:

1. Go to your target sandbox **Environment Details** page, and select the **Maintain** > **Move database** menu option.
2. Select **Import database** and choose your source database backup (.bacpac format) file from the Asset Library.
3. Note the warnings. Review the list of data elements that are cleaned up from the backup file.
4. The import operation will begin immediately.

To import a database to a developer environment after you've downloaded a database backup (.bacpac) file, you can begin the manual import operation on your Tier 1 environment. When you import the database, we recommend that you follow these guidelines:

- Keep a copy of the existing AxDB database, so that you can revert to it later if needed.
- Import the new database under a new name, such as **AxDB_fromProd**.

To ensure the best performance, copy the *.bacpac file to the local computer that you're importing from. Download sqlpackage .NET Core for Windows from Get sqlpackage .NET Core for Windows. Open a **Command Prompt** window, and run the following commands from the sqlpackage .NET Core folder.

```
SqlPackage.exe /a:import /sf:D:\Exportedbacpac\my.bacpac /tsn:localhost /tdn:<target database name>
/p:CommandTimeout=1200
```

Here is an explanation of the parameters:

- **tsn (target server name)** – The name of the Microsoft SQL Server instance to import into.
- **tdn (target database name)** – The name of the database to import into. The database should **not** already

exist.

- **sf (source file)** – The path and name of the file to import from.

> **NOTE**
> During import, the user name and password aren't required. By default, SQL Server uses Microsoft Windows authentication for the user who is currently signed in.

For information about how to complete the manual import operations into a Tier 1 environment, see Import the database.

Perform master data migration

Now that the UAT environment is hydrated with the golden configuration, you can begin to migrate master data. You can do this data migration by using data entities. We recommend that you complete your data migration activities before you copy the UAT environment to production, because you will have access to the database in the UAT environment for troubleshooting.

> **IMPORTANT**
> Document attachments are not copied from UAT to Production in the next step. If your go live requires attachments, you will want to import those in the Production environment directly.

Copy the sandbox database to production

When you're ready to do a mock go-live or actual go-live, you can copy the UAT environment to production. This process is often referred to as *cutover*. We recommend that you do a cutover more than one time before your actual go-live. In this way, you can get detailed time estimates for each step of the process.

Determine the **Environment type** of your production environment and follow the relevant steps accordingly.

Microsoft-managed

1. In LCS, on the project home page, select **Service requests**.
2. On the **Service requests** page, select **Add**, and then select **Sandbox to Production**.
3. In the **Sandbox to Production** dialog box, follow these steps:
    a. In the **Source environment name** field, select the sandbox environment to copy the database from.
    b. Set the **Preferred downtime start date** and **Preferred downtime end date** fields. The end date must be at least four hours after the start date. To help ensure that resources are available to run the request, it's recommended that you submit your request at least 24 hours before your preferred downtime window.
    c. Select the check boxes at the bottom to agree to the terms.

Self-service

1. In LCS, open the **Full details** for the production environment to load the **Environment page**.
2. In the **Maintain** menu, select **Move database**.
3. In the options of operations select **Refresh database**.
4. In the **Source environment** chose the sandbox where your golden configuration is. Note the important instructions found on the Refresh database page for this kind of operation.
5. Select the check box to confirm that you understand this operation will overwrite the production database. The operation starts immediately after submitting the request.

> **IMPORTANT**
> Every database refresh will create a new database that will reset the **Point-in-time-restore** chain of restore points.

Reconfigure environment specific settings

After the refresh is completed, use the **Sign off** button in LCS to close out of the operation. You then can start to configure the environment-specific settings.

First, sign in to the environment by using the admin account that can be found on the **Environment details** page in LCS. Here are some typical areas of reconfiguration. You might require additional reconfiguration, based on your setup and the independent software vendor (ISV) solutions that are installed:

- **System administration** > **Setup** > **Batch groups:** Add the various AOS instances to the batch server groups that you require.
- **System administration** > **Setup** > **Entity Store:** Update the various entities that you require for Microsoft Power BI reporting.
- **System administration** > **Setup** > **System parameters:** Reconnect the environment to the LCS Help configuration for task guides.
- **System administration** > **Setup** > **Email** > **Email parameters:** Enter the Simple Mail Transfer Protocol (SMTP) settings if you use email in your UAT environment.
- **System administration** > **Setup** > **Integration configuration** > **Azure storage account connection string:** Enter the storage account string.
- **System administration** > **Setup** > **System Parameters:** On the **Document connections** tab enter the Azure Key and Application Secret.
- **System administration** > **Inquiries** > **Batch jobs:** Select the jobs that you want to run in your UAT environment, and update the status to **Waiting**.

> **NOTE**
> As a best practice, all mission-critical batch jobs that will run with recurrence should be created and run by the admin account. The admin should be a generic user such as `erp@customer.com`. It should not be linked to a specific employee's Azure Active Directory (Azure AD) account, because that account might be disabled later if the employee leaves the company.

Open the environment to users

When the system is configured as you require, you can enable selected users to access the environment. By default, all users except the admin and Microsoft service accounts are disabled.

Go to **System administration** > **Users** > **Users**, and enable the users that should have access to the Production environment. If many users must be enabled, you can complete this task more quickly by using the Microsoft Excel Add-In.

Community tools

Are you looking for more tools to help you prepare backup files from your developer environments? Here are some other sources of information:

- D365fo.Tools provides many valuable tools that are created by the community.
- Community-provided open source projects on GitHub.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Destructive testing

2/18/2021 • 4 minutes to read • Edit Online

Database movement operations are a suite of self-service actions that can be used as part of data application lifecycle management (DataALM). In some situations, destructive testing must be done on an environment. In this context, destructive testing means that the environment is rendered no longer useful for continued testing. Destructive testing is typical in an implementation lifecycle during Conference Room Pilots. This tutorial shows how to use database movement operations to facilitate destructive testing.

In this tutorial, you will learn two approaches:

- Use a database backup asset.
- Use point-in-time restore.

As an example of this scenario, a customer wants to do a Conference Room Pilot and wants to start with an environment that has no transactions (that is, no sales orders or purchase orders). The customer will be traveling from physical warehouse to physical warehouse throughout the geographic region to do the same pilot, and wants the environment to be "reset" before each pilot is done.

Prerequisites

To complete this tutorial, you must have a standard user acceptance testing (UAT) environment deployed in your project.

Using a database backup

If you've prepared a database backup (.bacpac) file that is already at the starting point for the test, the easiest approach is to upload the backup file to the **Database backup** section in your LCS project's Asset Library. It can then be imported to your target environment as described here.

To import a database that is prepared from a developer environment to a standard user acceptance test (UAT), or a database previously exported from a UAT environment, follow the steps outlined below:

1. Go to your target sandbox **Environment Details** page, and select the **Maintain** > **Move database** menu option.
2. Select **Import database** and choose your source database backup (.bacpac format) file from the Asset Library.
3. Note the warnings. Review the list of data elements that are cleaned up from the backup file.
4. The import operation will begin immediately.

To import a database to a developer environment after you've downloaded a database backup (.bacpac) file, you can begin the manual import operation on your Tier 1 environment. When you import the database, we recommend that you follow these guidelines:

- Keep a copy of the existing AxDB database, so that you can revert to it later if needed.
- Import the new database under a new name, such as **AxDB_fromProd**.

To ensure the best performance, copy the *.bacpac file to the local computer that you're importing from. Download sqlpackage .NET Core for Windows from Get sqlpackage .NET Core for Windows. Open a **Command Prompt** window, and run the following commands from the sqlpackage .NET Core folder.

```
SqlPackage.exe /a:import /sf:D:\Exportedbacpac\my.bacpac /tsn:localhost /tdn:<target database name>
/p:CommandTimeout=1200
```

Here is an explanation of the parameters:

- **tsn (target server name)** – The name of the Microsoft SQL Server instance to import into.
- **tdn (target database name)** – The name of the database to import into. The database should **not** already exist.
- **sf (source file)** – The path and name of the file to import from.

> **NOTE**
> During import, the user name and password aren't required. By default, SQL Server uses Microsoft Windows authentication for the user who is currently signed in.

For information about how to complete the manual import operations into a Tier 1 environment, see Import the database.

Database backup pros and cons

The advantage of using backup file assets is that you can keep importing the same file to get back to the starting point for the test.

The disadvantage is that if many configurations (for example, batch jobs) must be set after the import is completed but before users can begin, more effort will be required before each destructive testing session.

Using point-in-time restore

If you didn't start with a database backup (.bacpac) file but instead have the UAT environment in a known good state, you can just record the date and time in your time zone. You can then begin the destructive testing. Then, when the testing is completed, you can restore the environment to the previous time by using the following steps.

To restore the database of a standard user acceptance test (UAT) environment to a previous point-in-time, follow the steps outlined below:

1. Go to your target sandbox **Environment Details** page, and select the **Maintain** > **Move database** menu option.
2. Select the **Point-in-time restore** option and choose a point-in-time.
3. Note the warnings. Review the list of data elements that are not copied over from the previous point-in-time.
4. The restore operation will begin immediately.

Point-in-time restore pros and cons

The advantage of using point-in-time-restore is that you can avoid dealing with database backup (.bacpac) files and can reduce the time between destructive testing sessions.

The disadvantage is that, because of current limitations of point-in-time restore, you must record a new restore date and time in your time zone after the restore is completed. Because point-in-time restore always creates a new database, the original date and time that were used won't be available as a restore point on the new database.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Database Movement API

2/18/2021 • 2 minutes to read • Edit Online

The Database Movement application programming interface (API) is a RESTful endpoint that is used to manage the data lifecycle of Microsoft Dynamics 365 environments. It provides a versioned set of capabilities that you can currently use to copy databases between environments, and to list and download database backups. More supported actions will be added in later releases.

What is supported by the Database Movement API?

The Database Movement API exposes RESTful endpoints for the following Dynamics 365 services:

- Dynamics 365 Finance
- Dynamics 365 Supply Chain Management
- Dynamics 365 Commerce

Next steps

- Learn how to set up authentication.
- Review the API reference.

Versioning and support

2/18/2021 • 2 minutes to read • Edit Online

This topic provides an overview of the versioning and breaking change policies for the Database Movement application programming interface (API).

Support and deprecation information

As new versions of the REST APIs are released, earlier versions will be retired. Microsoft will declare a version deprecated at least six months before it retires an API endpoint.

By incrementing the version number of the API (for example, from v1 to v2), Microsoft announces that the lowest version (in this example, v1) is immediately deprecated and will no longer be supported six months after the announcement. However, Microsoft might make exceptions to this policy for service health and security issues.

When an API is marked as deprecated, a date value will be entered in the **VersionEOL** (Version end of life) field. Therefore, you can proactively monitor this field and plan for upcoming changes.

Compatible and breaking changes

Microsoft will provide details of API changes in the private preview group. If the changes are non-breaking in nature, the API version number will remain the same. If the changes are breaking in nature, Microsoft will increment the API version number.

Here are some examples of breaking changes:

- The URL or fundamental request/response is changed.
- A declared property is removed or renamed, or its type is changed.
- The API or API parameters are removed or renamed.
- A required request parameter is added.

Here are some examples of non-breaking changes:

- Properties are added that are nullable or have a default value.
- A member is added to an enumeration.
- Paging is introduced to existing collections.
- Error codes are changed.
- The order of properties in requests or responses is changed.

Example of VersionEOL in a response contract

The following example shows a response contract in JavaScript Object Notation (JSON) format. All response contracts contain the **VersionEOL** property of which has a default value of DateMax() from the Microsoft .NET Framework. Your applications can monitor the value of this field on responses from Microsoft to get an immediate alert when Microsoft has deprecated a specific endpoint or a whole API version.

```
{
    "IsSuccess": true,
    "OperationActivityId": "55eb4327-9346-4c7b-82bd-fe8ef15112c6",
    "ErrorMessage": null,
    "VersionEOL": "9999-12-31T23:59:59.9999999"
}
```

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Database movement API - Authentication

2/18/2021 • 3 minutes to read • Edit Online

This topic provides overview information about how to authenticate with the Database Movement application programming interface (API).

Fundamentals

To call the Database Movement API, your application must acquire an access token from the Microsoft identity platform. The access token contains information about your application and the permission that it has to call resources in Microsoft Dynamics Lifecycle Services (LCS).

Access token

Access tokens that are issued by the Microsoft identity platform are base64–encoded JavaScript Object Notation (JSON) Web Tokens (JWTs). They contain information (claims) that the Database Movement API and other web APIs that are secured by the Microsoft identity platform use to validate the caller and make sure that the caller has the correct permissions to perform the operation that they are requesting. During calls, you can treat access tokens as opaque. You should always transmit access tokens over a secure channel, such as Transport Layer Security (TLS) and Hypertext Transfer Protocol Secure (HTTPS).

Here is an example of an access token that is issued by the Microsoft identity platform.

```
EwAoA8l6BAAU7p9QDpi/D7xJLwsTgCg3TskyTaQAAXu71AU9f4aS4rOK5xoO/SU5HZKSXtCsDe0Pj7uSc5Ug008qTI+a9M1tBeKoTs7tHzhJ
NSKgk7pm5e8d3oGWXX5shyOG3cKSqgfwuNDnmmPDNDivwmi9kmKqWIC9OQRf8InpYXH7NdUYNwN+jljffvNTewdZz42VPrvqoMH7hSxiG7A1
h8leOv4F3Ek/oeJX6U8nnL9nJ5pHLVuPWD0aNnTPTJD8Y4oQTp5zLhDIIfaJCaGcQperULVF7K6yX8MhHxIBwek418rKIp11om0SWBXOYSGO
M0rNNN59qNiKwLNK+MPUf7ObcRBN5I5vg8jB7IMoz66jrNmT2uiWCyI8MmYDZgAACPoaZ9REyqke+AE1/x1ZX0w7OamUexKF8YGZiw+cDpT/
BP1GsONnwI4a8M7HsBtDgZPRd6/Hfqlq3HE2xLuhYX8bAc1MUr0gP9KuH6HDQNlIV4KaRZWxyRo1wmKHOF5G5wTHrtxg8tnXylMc1PKOtaXI
U4JJZ1l4x/7FwhPmg9M86PBPWr5zwUj2CVXC7wWlL/6M89Mlh8yXESMO3AIuAmEMKjqauPrgi9hAdI2oqnLZWCRL9gcHBida1y0DTXQhcwMv
1ORrk65VFHtVgYAegrxu3NDoJiDyVaPZxDwTYRGjPII3va8GALAMVy5xou2ikzRvJjW7Gm3XoaqJCTCExN4m5i/Dqc81Gr4uT7OaeypYTUjn
wCh7aMhsOTDJehefzjXhlkn//2eik+NivKx/BTJBEdT6MR97Wh/ns/VcK7QTmbjwbU2cwLngT7Ylq+uzhx54R9JMaSLhnw+/nIrcVkG77Hi3
neShKeZmnl5DC9PuwIbtNvVge3Q+V0ws2zsL3z7ndz4tTMYFdvR/XbrnbEErTDLWrV6Lc3JHQMs0bYUyTBg5dThwCiuZ1evaT6BlMMLuSCVx
dBGzXTBcvGwihFzZbyNoX+52DS5x+RbIEvd6KWOpQ6Ni+1GAawHDdNUiQTQFXRxLSHfc9fh7hE4qcD7PqHGsykYj7A0XqHCjbKKgWSkcAg==
```

To call the Database Movement API, you attach the access token as a bearer token to the authorization header in your HTTP request. Here is an example.

```
HTTP/1.1
Authorization: Bearer EwAoA8l6BAAU ... 7PqHGsykYj7A0XqHCjbKKgWSkcAg==
Host: lcsapi.lcs.dynamics.com
GET https://lcsapi.lcs.dynamics.com/databasemovement/v1/databases
```

Register a new application by using the Azure portal

1. Sign in to the Microsoft Azure portal by using a work or school account, or a personal Microsoft account.
2. If your account gives you access to more than one tenant, select your account in the upper-right corner, and set your portal session to the Azure Active Directory (Azure AD) tenant that you want.
3. In the left pane, select the **Azure Active Directory** service, and then select **App registrations > New registration**.
4. When the **Register an application** page appears, enter your application's registration information:

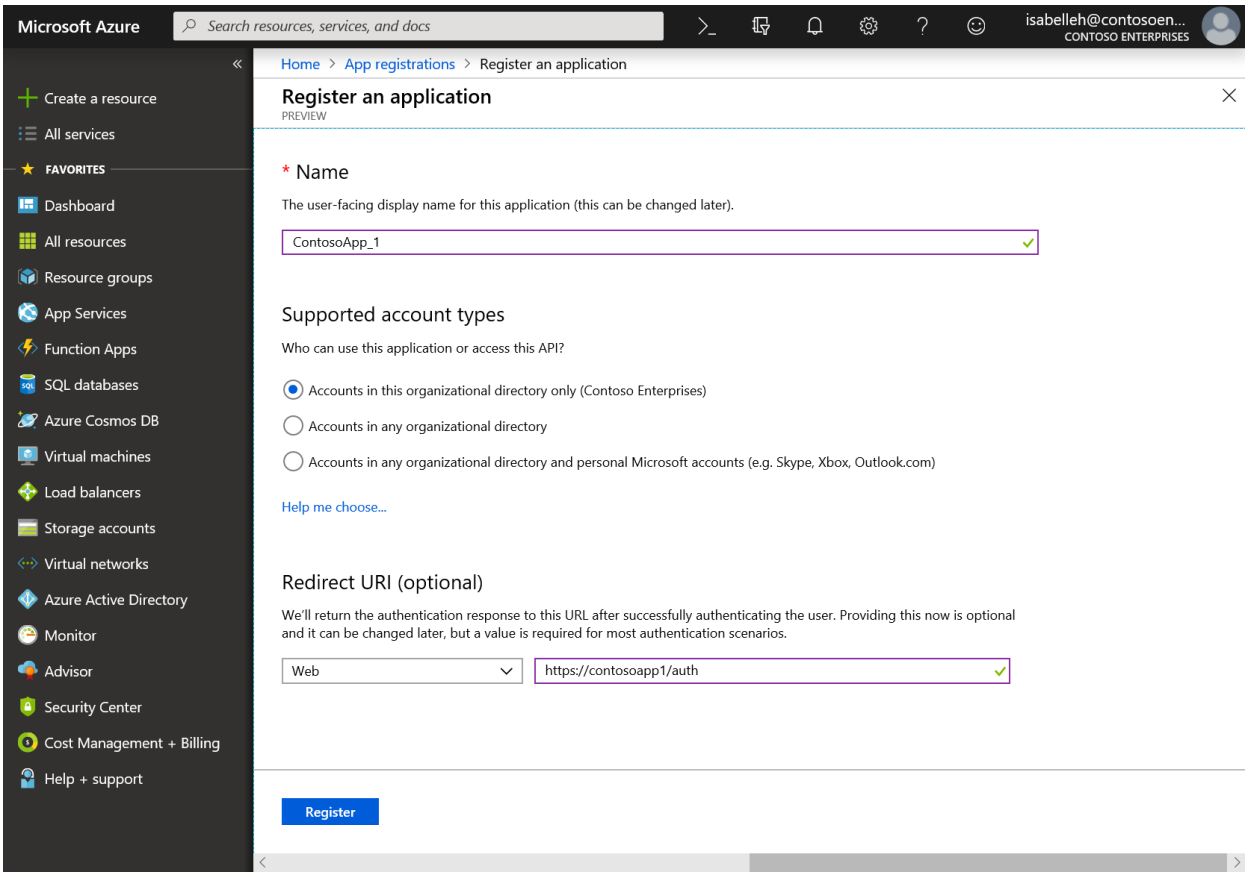
- **Name** – Enter a meaningful application name that will be shown to users of the app.
- **Supported account types** – Select the types of accounts that your app should support.

SUPPORTED ACCOUNT TYPES	DESCRIPTION
Accounts in this organizational directory only	Select this option if you're building a line-of-business app. This option isn't available unless you're registering the app in a directory. This option is mapped to **Azure AD only single-tenant**. This option is the default option unless you're registering the app outside a directory. In that case, the default option is **Azure AD multi-tenant and personal Microsoft accounts**.
Accounts in any organizational directory	Select this option to target all business and educational customers. This option is mapped to **Azure AD only multi-tenant**. If you registered the app as **Azure AD only single-tenant**, you can use the **Authentication** blade to update it to **Azure AD only multi-tenant** and then back to **Azure AD only single-tenant**.
Accounts in any organizational directory and personal Microsoft accounts	Select this option to target the widest set of customers. This option is mapped to **Azure AD multi-tenant and personal Microsoft accounts**. If you registered the app as **Azure AD multi-tenant and personal Microsoft accounts**, you can't change this setting in the user interface (UI). Instead, you must use the application manifest editor to change the supported account types.

- **Redirect URI (optional)** – Select the type of app that you're building: **Web** or **Public client (mobile & desktop)**. Then enter the redirect URI (or reply URL) for the app.
  - For web apps, provide the base URL of the app. For example, `http://localhost:31544` might be the URL for a web app that runs on your local machine. Users then use this URL to sign in to a web client app.
  - For public client apps, provide the URI that Azure AD uses to return token responses. Enter a value that is specific to your app, such as `myapp://auth`.

  To see specific examples for web apps or native apps, see the quick start guides from Azure AD.

5. Under **API permissions**, select **Add a permission**. Then, on the **APIs my organization uses** tab, search for **Dynamics Lifecycle services**, and add the **user_impersonation** permission to your app.
6. Select **Register**.



Azure AD assigns a unique application ID (client ID) to your app, and you're taken to the **Overview** page for your app. To add more capabilities to your app, you can select other configuration options, such as options for branding, and for certificates and secrets.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Throttling

2/18/2021 • 2 minutes to read • Edit Online

This topic provides an overview of throttling for the Database Movement application programming interface (API).

Rate limits

To help maintain the reliability of the service and reduce costs, throttling will be turned on for the Database Movement API. Throttling helps protect against malicious and excessive use of the RESTful endpoints. Database movement operations are some of the most time-consuming and CPU-intensive tasks that can be run from Microsoft Dynamics Lifecycle Services (LCS), and Microsoft might change the current call limits later.

Current limits

Currently, the Database Movement API has a global call limit of **three executions per 24-hour timeframe** for all actions that trigger a new operation in LCS. These operations include database refresh operations.

If you exceed the limit, you won't be able to start a new operation and will receive an error that resembles the following example.

```
{
    "IsSuccess": false,
    "OperationActivityId": "55eb4327-9346-4c7b-82bd-fe8ef15112c6",
    "ErrorMessage": "Maximum allowed API operations are 3 from 2019-09-30T04:01:01.9999999",
    "VersionEOL": "9999-12-31T23:59:59.9999999"
}
```

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

API v1 reference overview

2/18/2021 • 2 minutes to read • Edit Online

Welcome to the Database Movement application programming interfaces (API) reference for the version 1 (v1) endpoint. Use the table of contents to the left to view details of each endpoint that is available.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

List database backups

2/18/2021 • 2 minutes to read • Edit Online

You can retrieve a list of database backups from the Project asset library.

Permissions

One of the following permissions is required to call this application programming interface (API). For more information about permissions and how to select them, see Authentication.

PERMISSION TYPE	PERMISSIONS (FROM LEAST PRIVILEGED TO MOST PRIVILEGED)
Delegated (work or school account)	user_impersonation

HTTP request

```
GET /databasemovement/v1/databases/project/{projectId}
```

Request headers

HEADER	VALUE
Authorization	Bearer {token} (required)
'x-ms-version'	'2017-09-15' (required)
Content-Type	application/json

Request body

Don't supply a request body for this method.

Response

The response is always a **200 OK** response, unless you aren't correctly authenticated. Be sure to use the **IsSuccess** property to evaluate the success or failure of the action.

Example

```
GET /databasemovement/v1/databases/project/12345
```

```
{

    "DatabaseAssets": [
        {
            "Id": "4a2c52d4-49ca-4606-94a9-92b3a6b42985",
            "ProjectId": 12345,
            "OrganizationId": 1,
            "Name": "LanesBackupRecord",
            "FileName": "RandomFile.bacpac",
            "FileDescription": "",
            "FileLocation": "https://scrubbed.blob.core.windows.net/product-ax7productname/e6244b15-5112-
4d1d-a422-49c63496ab6d/AX7ProductName-12-17-83c6e642-676a-4048-b413-6e284f5d1f55-e6244b15-5112-4d1d-a422-
49c63496ab6d?sv=2015-12-11&sr=b&sig=rO3zmAZ3zM6s%2FV%2BeihBA2LMVvqMsxbtsnbauvd8keYo%3D&se=2019-09-
28T15%3A18%3A05Z&sp=r",
            "ModifiedDateTime": "2019-09-27T15:17:35.867",
            "CreatedDateTime": "2019-09-27T15:17:35.867",
            "CreatedByName": null,
            "ModifiedByName": null
        }
    ],
    "IsSuccess": true,
    "OperationActivityId": "55eb4327-9346-4c7b-82bd-fe8ef15112c6",
    "ErrorMessage": null,
    "VersionEOL": "9999-12-31T23:59:59.9999999"
}
```

Create a database refresh

2/18/2021 • 2 minutes to read • Edit Online

You can create a database refresh between two environments. Note that the same validation rules from the details page in Microsoft Dynamics Lifecycle Services (LCS) apply to the application programming interface (API).

Permissions

One of the following permissions is required to call this API. For more information about permissions and how to select them, see Authentication.

PERMISSION TYPE	PERMISSIONS (FROM LEAST PRIVILEGED TO MOST PRIVILEGED)
Delegated (work or school account)	user_impersonation

HTTP request

```
POST
/databasemovement/v1/refresh/project/{projectId}/source/{sourceEnvironmentId}/target/{targetEnvironmentId}
```

Request headers

HEADER	VALUE
Authorization	Bearer {token} (required)
'x-ms-version'	'2017-09-15' (required)
Content-Type	application/json

Request body

Don't supply a request body for this method.

Response

The response is always a **200 OK** response, unless you aren't correctly authenticated. Be sure to use the **IsSuccess** property to evaluate the success or failure of the action.

Example

```
POST /databasemovement/v1/refresh/project/12345/source/5362377c-bc37-4f92-b30e-fe0c1e664cc0/target/6a90b45f-1764-4077-b924-3f4671540237
```

```
{
    "IsSuccess": true,
    "OperationActivityId": "55eb4327-9346-4c7b-82bd-fe8ef15112c6",
    "ErrorMessage": null,
    "VersionEOL": "9999-12-31T23:59:59.9999999"
}
```

Create a database export

2/18/2021 • 2 minutes to read • Edit Online

You can create a database export from a sandbox environment to the project's asset library. Note that the same validation rules from the details page in Microsoft Dynamics Lifecycle Services (LCS) apply to the application programming interface (API).

Permissions

One of the following permissions is required to call this API. For more information about permissions and how to select them, see Authentication.

PERMISSION TYPE	PERMISSIONS (FROM LEAST PRIVILEGED TO MOST PRIVILEGED)
Delegated (work or school account)	user_impersonation

HTTP request

```
POST /databasemovement/v1/export/project/{projectId}/environment/{environmentId}/backupName/{backupName}
```

Request headers

HEADER	VALUE
Authorization	Bearer {token} (required)
Content-Type	application/json

Request body

Don't supply a request body for this method.

Response

The response is always a **200 OK** response, unless you aren't correctly authenticated. Be sure to use the **IsSuccess** property to evaluate the success or failure of the action.

Example

```
POST /databasemovement/v1/export/project/12345/environment/5362377c-bc37-4f92-b30e-
fe0c1e664cc0/backupName/TestBackupViaAPI
```

```
{
    "IsSuccess": true,
    "OperationActivityId": "55eb4327-9346-4c7b-82bd-fe8ef15112c6",
    "ErrorMessage": null,
    "VersionEOL": "9999-12-31T23:59:59.9999999"
}
```

Get status

2/18/2021 • 2 minutes to read • Edit Online

You can get the status of an ongoing operation.

Permissions

One of the following permissions is required to call this application programming interface (API). For more information about permissions and how to select them, see Authentication.

PERMISSION TYPE	PERMISSIONS (FROM LEAST PRIVILEGED TO MOST PRIVILEGED)
Delegated (work or school account)	user_impersonation

HTTP request

```
GET
/databasemovement/v1/fetchstatus/project/{projectId}/environment/{environmentId}/operationactivity/{operationactivityId}
```

Request headers

HEADER	VALUE
Authorization	Bearer {token} (required)
'x-ms-version'	'2017-09-15' (required)
Content-Type	application/json

Request body

Don't supply a request body for this method.

Response

The response is always a **200 OK** response, unless you aren't correctly authenticated. Be sure to use the **IsSuccess** property to evaluate the success or failure of the action.

Example

```
GET /databasemovement/v1/fetchstatus/project/12345/environment/5362377c-bc37-4f92-b30e-fe0c1e664cc0/operationactivity/55eb4327-9346-4c7b-82bd-fe8ef15112c6
```

```
{
    "IsSuccess": true,
    "OperationActivityId": "6a90b45f-1764-4077-b924-3f4671540237",
    "ErrorMessage": null,
    "VersionEOL": "9999-12-31T23:59:59.9999999",
    "ProjectId": 12345,
    "EnvironmentId": "5362377c-bc37-4f92-b30e-fe0c1e664cc0",
    "ActivityId": "55eb4327-9346-4c7b-82bd-fe8ef15112c6",
    "CompletionDate": null,
    "OperationStatus": "InProgress"
}
```


OperationStatus property

STATUS	DESCRIPTION
NotStarted	The action hasn't yet been started.
InProgress	The action is in progress.
Completed	The action was successfully completed.
Failed	The action was halted.
SignedOff	The action was successfully completed and signed off.
Aborted	The action was canceled without automated cleanup.
RollbackInProgress	Reversal of the action is in progress.
RollbackFailed	Reversal of the action was halted.
RollbackCompleted	Reversal of the action was successfully completed.

Start and stop environments

2/18/2021 • 2 minutes to read • Edit Online

You can start and stop environments through Microsoft Dynamics Lifecycle Services (LCS) via the Database Movement API. Using these APIs will ensure the LCS environment status is synced with the actual environment.

Note that the same validation rules from the details page in LCS apply to the API.

NOTE

- Only **Customer-managed** environments are supported. Self-service environments do not have the same concept of stop and start and are not supported by this API. Microsoft-managed environments are not supported.
- These APIs will trigger/invoke the operation. A successful response only indicates that the trigger was successful.
- For **stop**, non-success will be returned if the environment is already undergoing another operation or if the environment is already stopped.
- For **start**, non-success will be returned if the environment is already undergoing another operation but will return success if the environment is already started.

Permissions

One of the following permissions is required to call this API. For more information about permissions and how to select them, see Authentication.

PERMISSION TYPE	PERMISSIONS (FROM LEAST PRIVILEGED TO MOST PRIVILEGED)
Delegated (work or school account)	user_impersonation

HTTP request

Use the following POST method to send an HTTP request to stop or start an environment.

Stop an environment

```
POST /environment/v1/stop/project/{projectId}/environment/{environmentId}
```

Start an environment

```
POST /environment/v1/start/project/{projectId}/environment/{environmentId}
```

Request headers

Use the following header value in the HTTP request header.

HEADER	VALUE
Authorization	Bearer {token} (required)

HEADER	VALUE
'x-ms-version'	'2017-09-15' (required)
Content-Type	application/json

Request body

Don't supply a request body for this method.

Response

The response is always a **200 OK** response, unless you aren't correctly authenticated. Be sure to use the **IsSuccess** property to evaluate the success or failure of the action.

Example

Request to stop an environment

```
POST /environment/v1/stop/project/{projectId}/environment/{environmentId}
```

Successful response

```
{
    "IsSuccess": true,
    "OperationActivityId": "55eb4327-9346-4c7b-82bd-fe8ef15112c6",
    "ErrorMessage": null,
    "VersionEOL": "9999-12-31T23:59:59.9999999"
}
```

Demo data overview

2/18/2021 • 5 minutes to read • Edit Online

This topic provides an overview of the demo data that is available.

Demo data is the base data set that is released for implementation support and demonstration purposes. The current demo data set supports the following verticals:

- Commerce
- Distribution
- Service Industries
- Public Sector
- Discrete & Process Manufacturing

The demo data set supports 40 languages across 16 countries or regions. It also supports various implementation scenarios. The following table lists the legal entities that are included in the demo data set.

LEGAL ENTITY	DESCRIPTION
BRMF	Contoso Entertainment System Brazil
CNMF	Contoso Entertainment China
DAT	Default Company
DEMF	Contoso Entertainment System Germany
FRRT	Contoso Retail FR
FRSI	Contoso Consulting FR
GBSI	Contoso Consulting GB
GLCO	Contoso Group
GLMF	Contoso Entertainment System
GLRT	Contoso Retail
GLSI	Contoso Consulting
INMF	Contoso India
ITCO	Contoso Italy
JPMF	Contoso Entertainment Japan
MXMF	Contoso Entertainment System Mexico

LEGAL ENTITY	DESCRIPTION
MYMF	Contoso Entertainment System MYMF
RUMF	Contoso Entertainment System Russia
RURT	Contoso Retail RUS
US01	Contoso US01
USMF	Contoso Entertainment System USA
USP2	Contoso Orange Juice
USPI	Contoso Process Industry
USRT	Contoso Retail USA
USSI	Contoso Consulting USA

Embedded analytics

Demo data has been updated in five companies to provide better reports on the new embedded analytics within workspace. Filter the embedded analytics to the following legal entities for the improved report data:

LEGAL ENTITY	DESCRIPTION
DEMF	Contoso Entertainment System Germany
GLMF	Contoso Entertainment
USMF	Contoso Entertainment System USA
USRT	Contoso Retail USA
USSI	Contoso Consulting USA

Reports from the Cash overview Power BI content are displayed in the **Cash overview** and **Bank management** workspaces.

To view the Cash flow forecasting reports with data, you must first run the forecast calculation process using the **Calculate cash flow forecasts** function from the **Cash and bank management area**. This needs to be completed for each company included in the forecast. You then need to refresh the LedgerCovLiquidityMeasurement aggregate measure on the **Entity Store** page.

For demonstration purposes, you can add cash flow forecasting demo data using the **Generate data** page from the **Demo data** module. This script will insert data into the cash flow forecasting tables to quickly populate information necessary for reports.

The credit and collections analytics can be viewed on the **Manage customer credit and collections** workspace. To view the analytics you need to refresh the CustCollectionsBIMeasurements aggregate measure on the **Entity Store** page.

The vendor payments analytics can be viewed on the **Vendor payments** workspace. To view the analytics, you

need to refresh the VendPaymentBIMeasure aggregate measure on the **Entity Store** page.

The Purchase performance analytics can be viewed on the **Purchase spend analysis** page from the Procurement and sourcing module. To view the analytics, you need to refresh the Purchase cube aggregate measure on the **Entity Store** page.

The Sales and profitability analytics can be viewed on the **Sales and profitability performance** page from the Sales and marketing module. To view the analytics, you need to refresh the Sales cube aggregate measure on the **Entity Store** page.

The production performance analytics can be viewed on the **Production performance** page from the Production control module. To view the analytics, you need to refresh the Production cube aggregate measure on the **Entity Store** page.

For demonstration purposes, you can add production performance demo data using the **Generate data** page from the Demo data module. This script will generate production orders and with associated feedback journals to populate the production performance reports with data.

The warehouse performance analytics can be viewed on the **Warehouse performance** page from the Warehouse management module. To view the analytics, you need to refresh the Warehouse aggregate measure on the **Entity Store** page.

For demonstration purposes, you can add warehouse performance demo data using the **Generate data** page from the Demo data module. This script will generate sales orders and warehouse work to populate the warehouse performance reports with data.

The demo data module is only available if you have the Demo data suite model deployed on the environment.

Vendor collaboration

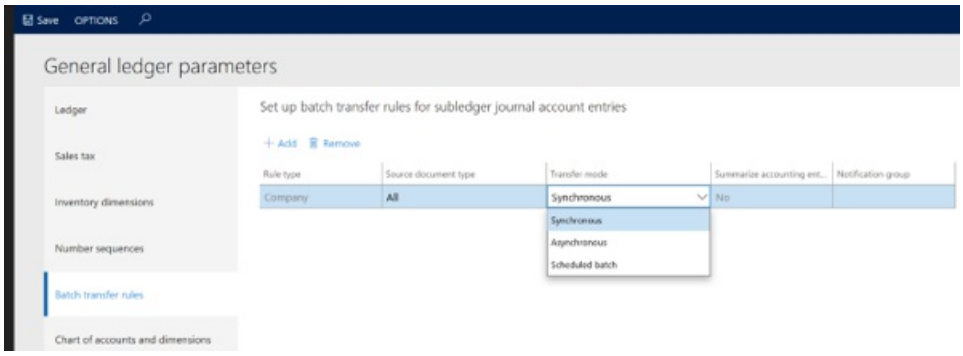
In the USMF demo company, there are three purchase orders for vendor US-104 to use for demonstration purposes. You can log in as user ErinH, who is a contact person who has access to vendor collaboration for US-104.

Purchase order approval

In the USMF demo company, there are two purchase orders for INGA to approve. You can log in as user INGA to see the purchase orders awaiting approval.

Batch transfer rules for subledger journals

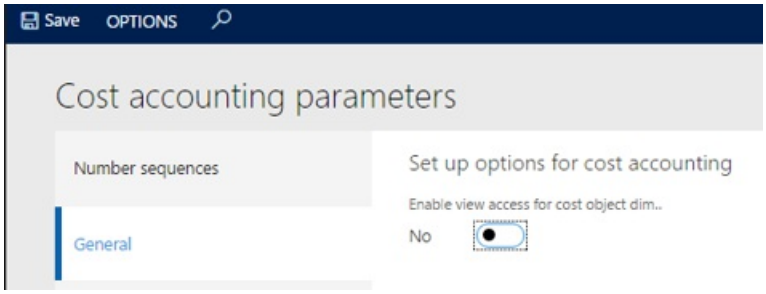
The batch transfer rules for subledger journal account entries have been changed to **Scheduled batch** to reflect a best practice. The batches are configured to run every 10 minutes. It is important to understand that accounting entries for all source documents will not be reflected in General ledger until the batch process has run. If you have requirements to see the immediate effect in General ledger, set the **Transfer mode** to **Synchronous** on the **Batch transfer rules** page within **General ledger** parameters. While Synchronous works well for product demos and environments with low transaction volumes, it can cause performance issues in larger transaction volume environments.



Cost accounting

Three Cost accounting ledgers are created in demo data. The Cost accounting ledger USP2 provides an E2E demo experience based on data from legal entity USP2. The Cost control unit consists of 2 Cost object dimensions (Cost centers and Product groups). Actual cost, Budget cost and Statistical measures are transferred for all 12 fiscal periods of year 2017. Overhead calculation has also been performed for all fiscal periods of year 2017.

Access level security is configured but not enabled. This is enabled in the **Cost accounting parameters** page.



After Access level security has been enabled, you can assign an employee to the role Cost object controller. You can log in as the employee and access the **Cost control** workspace. The employee can now see their Cost center performance and drill into details of how these were calculated.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Generate demo data by using data packages

2/18/2021 • 16 minutes to read • Edit Online

In previous releases, demo data was delivered as a database. In Microsoft Dynamics 365 for Finance and Operations, Enterprise edition 7.3, a subset of demo data has been released as data packages. These packages are available in the Shared asset library in Microsoft Dynamics Lifecycle Services (LCS). The packages are designed so that they can be loaded into an empty environment.

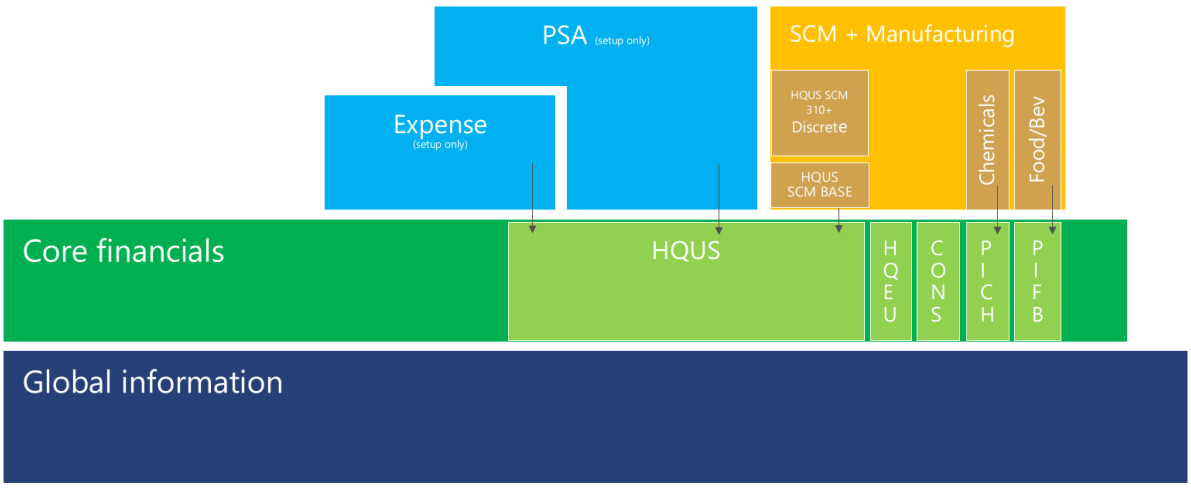
Here are some of the benefits of using data packages instead of a database to deliver demo data:

- The download times are significantly faster.
- You can import just the data packages that you require.
- You can edit the spreadsheets to customize the data for your customers.
- Updated demo data can be provided very quickly through LCS.

> **NOTE**
> The demo data packages aren't yet a full replacement for the demo database that is currently provided.

How the packages are organized

The demo data packages are designed to be layered on top of each other, as shown in the following illustration.



However, the global information for one demo scenario might have completely different requirements than the global information for another demo scenario. For example, the dimensions for one scenario will interfere with the dimensions for another scenario. In this case, a separate global information package will be created, and only packages that are related to that global information can be layered on top of the package.

For example, there is currently a commercial system and shared package as well as a separate public sector system and shared package that can't be used together.

System and Shared package

The base package, System and Shared, is the foundation for all other packages. This package creates legal entities, loads the global address book, and adds other shared information. It must be loaded first to support all the remaining packages. The package is named **100-System and Shared.zip**.

After the System and Shared package is loaded, you will see one or more of the following legal entities.

LEGAL ENTITY	DESCRIPTION
HQUS	The US-based headquarters for your demo company. This company is based on the original USMF data, but it has been changed to remove the manufacturing focus in the name. It includes setup information that is intended for US companies.
HQEU	The non-US-based headquarters for your demo company. This company is based on the original DEMF data, but it has been changed to remove the manufacturing focus in the name. It includes setup information that is intended for non-US companies.
CONS	A small consolidations company.
PICH	A process industries company that is focused on chemicals.
PIFB	A process industries company that is focused on food and beverages.

Financials

The Financials data packages contain data for the general ledger, bank, accounts payable, tax, accounts receivable, fixed assets, and budgeting for a single company. The names of these data packages consist of **200-Financials** followed by the legal entity that the packages are intended for. For example, the Financials data package for the HQUS legal entity is named **200-Financials-HQUS.zip**.

At least two financial companies are required for cross-company tasks such as centralized payments. To facilitate cross-company tasks, all customers and vendors have been added to each legal entity. The CONS legal entity is required if you want to do consolidations.

The Financials data packages also have five inventory products to support the creation of invoices that can move through the accounts receivable and accounts payable processes. These items use a minimum of inventory and product functionality to support those processes. However, you no longer have to set up products when you want to demonstrate only Financials functionality. More complete products will be added when you import the Supply chain data packages.

Expense management

The expense management data packages contain data for expense management and aren't specific to project management. The names of these data packages consist of **225-Expense** followed by the legal entity that the packages are intended for. For example, the Expense management data package for the HQUS legal entity is named **225-Expense management HQUS.zip**.

Project management and accounting

The Project management and accounting data packages contain data for project accounting and expense management. The names of these data packages consist of **250-Project management and accounting** followed by the legal entity that the packages are intended for. For example, the Project management and accounting data package for the HQUS legal entity is named **250-Project management and accounting-HQUS.zip**.

Supply chain

The Supply chain data packages contain data for inventory management, product information, procurement and sourcing, sales and marketing, quality management, warehouse management, transportation management,

production control, process manufacturing, costing, and master planning for a single company. Because of the large number of entities, the Supply chain packages for some companies have been split into two packages. You must load both packages to complete the supply chain scenarios. However, you can load these packages as two separate projects.

These names of these data packages consist of **300-Supply chain** followed by the legal entity that the packages are intended for. For example, the Supply chain data package for the PICH legal entity is named **300-Supply chain-PICH.zip**. The supply chain package for the HQUS legal entity is split into packages that are named **300-Supply chain 1 of 2-HQUS.zip** and **300-Supply chain 2 of 2-HQUS.zip**.

Demo data package releases

The demo data packages will be released through LCS and will be specific to a release. Note that the contents of a given package are subject to change as we add more demo scenarios and tune the packages. Additional packages will also be released as we add more module areas and industry-specific scenarios.

Package names will include a release identifier. For example, for Finance and Operations 7.3, **Demo data-7.3-** will precede the package name that uses the previously described naming conventions. For example, the full name of the Financials package for the HQUS legal entity for Finance and Operations 7.3 will be **Demo data-7.3-200-Financials-HQUS.zip**.

Before you load the packages

Before you load the data packages, you must manually follow these steps.

1. If you want to sign in as a specific user, change the user's email address to the sign-in address that you want to use. You can make this change in the **User information** data entity spreadsheet or, after you load data, on the **Users** page (**System administration** > **Users**).
2. Start the **Ready to post** batch scheduler. This batch job automatically posts transactions. You must start the scheduler in every legal entity where data should be processed. Follow the steps in the "The Ready to post process" section later in this topic.
3. If you aren't using the en-us locale, you may need to alter the source data format to match the format the packages were built on. Once you've loaded a data entity, click into the source data format column which should be specified as value = Excel. From the next page, again select Excel. From within the source data formats page, the bottom fast-tab will be regional settings. Change the language locale to en-us if it isn't specified as en-us. After loading packages, you can change it back to its original non en-us value.

Load the packages

The data packages must be loaded into a specific legal entity in a specific order. The number before the name of the package gives you guidance about the order that the data must be loaded in. For example, you must import **100-System and Shared.zip** before you can load the Financials package for the HQUS legal entity, **200-Financials-HQUS.zip**. Then, to add Supply chain data to the HQUS legal entity, you can load **300-Supply chain 1 of 2-HQUS.zip** and **300-Supply chain 2 of 2-HQUS.zip**.

Follow these steps to load the packages.

1. Start with an empty instance where no data is loaded.
2. Open the **Data management** workspace.
3. Select the **Import** tile to create an import job.
4. Enter a name for the job. For example, enter **Import shared information**.
5. Select **Add file**.
6. Select **Upload and add**, and browse to the data package that you want to import. You must start with the System and Shared data package.

7. Select the data package, and wait for the data to be loaded.
8. After the data is loaded, close the dialog box, and then select **Import**.
9. Repeat steps 5 through 8 for every additional package that you want to load. Be sure to switch to the legal entity that the data package is intended for.

Loading package combinations

The following packages can be loaded. When you import any package except the System and Shared package, you must be in the legal entity that is listed in the package name. The System and Shared package can be loaded from any legal entity. However, it's typically loaded from the default company, DAT.

Commercial Data

DESCRIPTION	NOTES
100 - System and Shared	Load this package before you any other package.
200 - Financials - HQUS	You can load this package alone or together with another Financials package.
200 - Financials - HQEU	You can load this package alone or together with another Financials package.
200 - Financials - CONS	You can load this package alone or together with another Financials package.
200 - Financials - PICH	You can load this package alone or together with another Financials package.
200 - Financials - PIFB	You can load this package alone or together with another Financials package.
225 - Expense Management - HQUS	Follow the prerequisite step before starting the import. Load this package after the HQUS Financials package.
250 - Project management - HQUS	Load this package after the HQUS Financials or Expense package. If you want to also use Supply chain, you must import this package first.
300 - Supply chain 1 of 2 (base) - HQUS	Load this package after the HQUS Financials package. If you want to also use Project management, you must import the Project management package first before you import this package.
310 - Supply chain 2 of 2 (Discrete) - HQUS	Load this package after the HQUS Supply chain base package.
300 - Supply chain - PIFB	Load this package after the PICH Supply chain package.
300 - Supply chain - PICH	Load this package after the PIFB Supply chain package.
900 - Financial transactions - HQUS	Load this package after the HQUS Financials package.
900 - Financial transactions - HQEU	You can load this package alone or together with another Financials package.

Public Sector data

DESCRIPTION	NOTES
100 - Public Sector System and Shared	Load this package before you any other package.
200 - PSUS Financials	You can load this package alone.
900 - PSUS Financial transactions	Load this package after the PSUS Financials package.

To load the data correctly, you need to load one package at a time, import it, and then load the next one once the import is complete. We are considering additional methods for loading the demo data to improve the process.

Troubleshooting and known issues

> **NOTE**
>
> We discovered an issue with the Number sequence references entity that causes a random failure during import although the data in the packages is correct. If you see an error during the import of number sequence references, follow these steps to process the failed records.
>
> 1. Click on the name of the entity (**number sequence references**) to display a form that lists all of the records in the data package.
> 2. Click on **Copy data to target**.
> 3. Change the **Run for** value to **Criteria** and **Change Rows with previous errors** to **Yes**.
> 4. Click **Ok** and then click **Run** on the form that appears.
> 5. Repeat these steps until all records import without error.

> **NOTE**
>
> There is currently an issue that some data entities have the same name, which can cause an import failure for **document types** and **date intervals** in the **200 - Financials** packages. If you see an error during the import of these entities extract them from the .zip file provided and manually import, making sure to use the **document types** pointing to **DocuTypeEntity** and **date intervals** pointing to **LedgerDateIntervalEntity**. Once these are imported you can retry the failed records from the **200 - Financials** packages.

After you load the packages

Check **Ready to post** if anything needs to be posted. In some cases individual demo scripts may recommend some setup be done prior to loading this data.

In some cases, there might be data that you want to add because of a special scenario or a missing entity. Add that data after you've finished loading the data packages. You might also want to manually post additional transactions or add your own data packages to enhance the demo experience.

After you load the data packages, you must also manually follow these steps.

1. Start the workflow jobs. Select **System administration** > **Workflow infrastructure configuration**, and then select **OK**.
2. Set up policy precedence rules. Select **Procurement and sourcing** > **Setup** > **Policies** > **Purchasing policies**, and then select **Parameters**. Then select **Companies**, and move it to the right column.
3. Setup policy precedence prior to importing Expense packages. Select **Expense management** > **Setup** > **Policies** > **Expense report**, and then select **Parameters**. then select **Companies**, and move it to the right column.
4. After you load the Project management and accounting packages, you must run the **Resource capacity roll-up** batch job. You can run this job from the **Synchronize resource capacity roll-ups** page (**Project management and accounting** > **Periodic** > **Capacity synchronization** > **Synchronize resource**

capacity roll-ups). Specify an end date that lets you schedule resources a long time in the future. After the batch job is run, automatic generation of team functionality will be enabled in the project's work breakdown structure (WBS).

5. Add Print management settings for each module.

Scenario scripts

Scripts have been provided for many of the scenarios that the demo data supports. You can find these scripts in Demo data scripts.

Transactions and automatic posting

Many scenarios for demo data require that transactions be processed after they are imported. You can process transactions by using the Ready to post feature. This feature includes both a page that lets you define the transactions that should be posted, and an entity that lets you import the definitions and automatically run them.

The following transaction types are supported when demo data is posted.

DOCUMENT	ENTITY DOCUMENT ID	DATE FILTER	ID FILTERS	OTHER FILTERS
Budget registry update	BudgetRegistryUpdate	Default date	Budget entry number	Not in use, Status = Draft
Costing version	CostingVersion	Not applicable	Version ID	Version Activation blocked = No
Customer payment journal	CustomerPaymentJournal	Transaction date	Journal number	Not posted, not workflow, not system blocked
Daily journal	GeneralJournal	Transaction date	Journal number	Not posted, not workflow, not system blocked
Fixed assets journal	FixedAssetsJournal	Transaction date	Journal number	Not posted, not workflow, not system blocked
Free text invoice	FreeTextInvoice	Invoice date	Not applicable	
Inventory adjustment journal	InventoryAdjustmentJournal	Transaction date	Journal number	Not posted
Invoice journal	InvoiceJournal	Transaction date	Journal number	Not posted, not workflow, not system blocked
Price calculation	PriceCalculation	Not applicable	Version ID	Version Activation blocked = No
Purchase order	PurchaseOrder	Delivery date	Purchase order ID	Able to confirm/PR/Vendor confirm/invoice

DOCUMENT	ENTITY DOCUMENT ID	DATE FILTER	ID FILTERS	OTHER FILTERS
Sales order	SalesOrder	Delivery date	Sales order ID	Able to confirm/packing slip/invoice
Trade agreement	TradeAgreement	Not applicable	Price/discount journal number	
Vendor invoice	VendorInvoice	Posting date	Invoice number	Approved, not in use, not yet
Vendor payment journal	VendorPaymentJournal	Transaction date	Journal number	Not posted, not workflow, not system blocked

For journals that support date ranges, the Ready to post process examines all the journal lines for a date that falls in the specified range. If any line in the journal falls in the date range, the search is stopped, and the whole batch is posted.

The Ready to post process

The Ready to post feature uses a batch to monitor the list of transaction types that you want to post. When the monitor detects a transaction of the type that you want to post, it uses the transaction type to generate a batch that posts those transactions. The batch that is created is the same type of batch that is created when you use the user interface for that transaction type. When the transaction batch is completed, the Ready to post monitor updates the list with the results of the processing. It also adds links to the batch and the original transaction.

Use the Ready to post page to process transactions

1. Select **System administration** > **Periodic tasks** > **Batch job ready to post** to open the **Ready to post** page.
2. Select **Create posting monitor**, and set up the batch parameters so that a recurring batch is running. You must complete this step only one time for each legal entity to start the batch process for posting.
3. Select **New**, and enter a name for the demo data job. The job name must be unique across all companies.
4. Select **Add line** to add a transaction type.
5. Select the transaction target. For journals, the target is **Post**. For other transactions, the target depends on the transaction type.
6. Specify a start date and an end date (that is, a date range) to limit the transactions that will be processed (when available).
7. Specify a "from" document and a "to" document (that is, a document range) to limit the transactions that will be processed (when available).
8. Select **Add line** to add additional transaction types. You can use the same type on multiple lines.
9. Select **Mark ready to post**. The batch status is changed from **Open** to **Ready**, and the posting monitor starts to process each line.
10. If you want to process a document immediately, select **Process documents**. The batch status is changed to **Scheduled**, and a batch is started without using the posting monitor.

When the batch is running, the status is changed to **In Progress**.

When the batch is completed, the status is changed to **Successful** or **Error**, depending on the results. The posting results are shown at the bottom of the page.

Use the Ready to post entity to process transactions

An entity that is named **Demo data posting** lets you import a list of document types that you want to post. The

entity will create a demo data job on the **Ready to post** page. If you've started the posting monitor, the transactions are automatically posted after you import the data by using the entity.

The following columns appear in the **Ready to post** entity.

COLUMN	PURPOSE
DemoDataJob	The unique ID of the demo data job to run. Use the same ID for every line that belongs to a single job.
LineNum	The order that the tasks will be run in.
DataProjectId	A link to the data project that contains the **Ready to post** entity. This information is for export only.
DemoDataJobStatus	The status of your demo data project. This information is for export only.
Document	The document type to process.
DocumentTarget	The process to run. For journals, the target must be **Post**. For transactions such as sales orders, the target will match the options that appear on the page when you add the task.
EndDate	An optional end date that limits the transactions that are processed.
FromDocument	An optional "from" document that limits the transactions that are processed.
ProcessOnImport	If you change the value to **Yes**, the demo data job will be set to **Ready**, and the process monitor will pick it up automatically. No action is required.
StartDate	An optional start date that limits the transactions that are processed.
ToDocument	An optional "to" document that limits the transactions that are processed.

Insert the **Ready to post** entity at the end of your data project, after all the transaction entities. In the data project, specify a sequence number that is larger than the sequence numbers that are used for the transactions entities.

If you have a mixture of transactions, some of which should be processed whereas others should not be processed, you must use date and document ranges to limit the transactions that are processed. If you can't use the ranges, you must use a separate data package for the unposted transactions.

Deploy a demo environment

2/18/2021 • 2 minutes to read • Edit Online

This topic explains how to deploy a demo environment on Microsoft Azure using Microsoft Dynamics Lifecycle Services (LCS). This topic applies to deploying a demo environment for:

- Dynamics 365 Finance
- Dynamics 365 Supply Chain Management
- Dynamics 365 Commerce

Prerequisites

Before you begin your deployment, the following prerequisites must be in place:

- Verify that you have an Azure subscription, and that you are a co-administrator on it.
- Verify that you have access to an LCS project and permissions to deploy an environment.
- Verify that you've connected your Azure subscription to your LCS project by using the information in the Complete the Azure Resource Manager (ARM) onboarding process topic.

Deploy a demo environment

Use this procedure to deploy a demo environment on Azure using LCS.

1. In LCS, open your project, and then, in the **Environments** section, click the plus sign (**+**).
2. Select the Azure environment topology, and then select **Demo**.
3. Select a topology.
   - For Finance and Operations, select the most recent Azure Resource Manager (ARM) topology for Finance and Operations.
   - For Commerce, select **Dynamics 365 for Commerce - Demo**.
4. In the **Deploy environment** dialog box, enter the name of the environment. This name should be unique in the Azure subscription. To make environments easy to identify, consider forming an acronym using the user's name and the topology.
5. Select the size of the virtual machine (VM). You must use **Ev3-series sizes** for Finance and Operations workloads. We recommend **Ev3**. If you experience allocation failures, see the Azure troubleshooting guide.
6. Set the **Instances** field to 1.

   > **NOTE**
   > The size of the VM and the number of instances affect the cost of your subscription. For more information, see Azure pricing.

7. Click **Advanced settings** to add customizations to your deployment. For the demo environment, we recommend that you keep the default settings.
8. Agree to the licensing and pricing terms, and then click **Next**.

9. In the **Confirm** message box, click **Deploy**.
10. Open the **Cloud hosted environments** page to view the status of the deployment. After the deployment is successfully completed, the environment will be ready.

Log on to your demo environment

To log on to your demo environment, do the following.

1. In LCS, open the **Cloud-hosted environments** page, and select the demo environment that you just deployed.
2. Scroll to the right and in the **Environment details** pane, under **Cloud services**, click the appropriate link:
   - **Log on to Finance and Operations**
   - **Log on to Commerce**

Cloud deployment overview

2/18/2021 • 15 minutes to read • Edit Online

Working with Microsoft to deploy Finance and Operations apps in the cloud requires that you understand the environment and subscription that you are deploying to, who can perform which tasks, and the data and customizations that you need to manage. We recommend that you sign up for the Full Microsoft FastTrack for Dynamics 365 to help speed your deployment and implementation - it's a program that provides training and consulting to help you realize business value faster. For more information, see Microsoft FastTrack. If you choose to use the Essentials FastTrack program instead, you will be using the Implementation Project Methodology in Lifecycle Services (LCS) to help you manage your implementation project.

Customer lifecycle, subscriptions, and environment types

Microsoft assumes that all customers will follow a lifecycle similar to the following for all cloud deployments, and therefore need different environment topologies at each phase.

- Evaluate
- Develop customizations, if needed.
- Curate a "golden configuration" environment that contains only module configurations without master or transactional data. This is to be the baseline for your data migration testing and eventual go live.
- Install and test customizations and partner solutions on a tier-1 sandbox (Development or test environment).
- Test customizations, partner solutions and data configuration on a tier-2 sandbox environment.
- Deploy customizations and data configurations to a production environment with high availability.

At some phases of a project, you may have all of the environments live at once. For more information, about the default licenses and tiers that are available, see the Dynamics 365 Licensing Guide.

You may notice the terms cloud hosted or Microsoft subscriptions. A *cloud hosted subscription* means that the customer or partner brings their own Azure subscription and deploys Finance and Operations apps to it, for evaluation and development purposes only. The customer or partner pays for the resources deployed to their Azure subscription based on the Azure price list. A *Microsoft subscription* means that the customer purchases Finance and Operations licenses, which will then allow them to deploy environments to an Azure subscription which is managed by Microsoft, therefore, the customer has no separate Azure billing.

With each Enterprise offer, two environments are included by default:

- One Tier 2 sandbox (multi-box environment) for user acceptance testing (UAT).
- One production environment with high availability (HA).

Additional environments may be purchased as add-ons. For information about licensing and what is included in Microsoft Dynamics 365, see the Dynamics 365 Licensing Guide.

Here's how the lifecycle maps to the available environments. If you already have environments deployed in your Lifecycle Services project, you can find the Environment Type and Environment Sub type on each environment's details page.

LIFECYCLE PHASE	ENVIRONMENT TIER	SUBSCRIPTION	ENVIRONMENT TYPES	ENVIRONMENT SUB-TYPE
Evaluation and analysis	Tier 1 Sandbox	Cloud hosted	Customer Managed	Demo

LIFECYCLE PHASE	ENVIRONMENT TIER	SUBSCRIPTION	ENVIRONMENT TYPES	ENVIRONMENT SUB-TYPE
Customize	Tier 1 Sandbox	Cloud hosted or VHD	Customer Managed	Develop
Golden configuration	Tier 1 Sandbox	Cloud hosted	Customer Managed	Develop
User acceptance testing (UAT)	Tiers 2-5 Sandbox	Microsoft	Microsoft Managed or Self-service	Not applicable
Go live	Production	Microsoft	Microsoft Managed or Self-service	Not applicable

Tiers 2-5 can be purchased to increase performance of the environment. The higher the tier, the more compute and database capacity is reserved for your use. For more information about Self-service environment types, check out the Self-service deployment overview.

Environment lifecycle operations

Users with the Environment Administrator or Project Owner roles in Lifecycle Services can perform various lifecycle operations on their environments. These operations often involve downtime on the environment until the task is finished. Each of these operations are located under or next to the **Maintain** button on each environment details page.

LIFECYCLE OPERATION	DESCRIPTION	LEARN MORE
Apply software	Install Microsoft updates, ISV solutions, or your own customization packages.	Apply updates to cloud environments
Enable access	Allow list your IP for Remote Desktop or database access	See the Remote Desktop section later in this topic
Restart services	Ability to restart components of your environment	Restart environment services
Move database	Full data lifecycle management	Database movement operations
Maintenance mode	Ability to change configuration with only admin access	Maintenance mode
Upgrade	Upgrade code and data from 7.x to the latest version	Process for moving to the latest update
Deallocate	Ability to turn off an environment not being used, or to troubleshoot a failed action	Not applicable
Start	Ability to turn on an environment for use	Not applicable
Delete	Ability to delete an environment previously deallocated	Not applicable

Security and compliance

Finance and Operations is PA-DSS 3.1 certified which means that all communications between components are secured out-of-the-box.

All Finance and Operations front-end virtual machines in Microsoft Azure are configured during deployment to only accept TLS 1.2.

IMPORTANT

Customers who have administrator access to Microsoft-managed sandboxes, including any add-on sandboxes purchased, must follow these guidelines:

- By default, automatic Windows update is enabled for all Tier 1 - 5 sandboxes and should NOT be disabled. This ensures that any time that Microsoft pushes security or critical infrastructure updates to your environment, your environment receives the latest set of updates and is updated each month with the operating system fixes that Microsoft releases.
- Admin passwords on these environments should NOT be changed. Environments that have admin passwords changed will be flagged by Microsoft. Microsoft reserves the right to, and will reset the admin password.
- Adding new user accounts to any Microsoft managed VM is NOT permitted. Microsoft reserves the right to, and will remove the newly added user accounts without providing notice.

> Finance and Operations is not covered by a FedRAMP ATO at this time. If Finance and Operations is provisioned in the United States, all customer data at rest is stored in data centers located in the United States, as described in International availability of Dynamics 365. Finance and Operations does not support any other Dynamics 365 US Government or Microsoft 365 GCC compliance attributes (for example, access by US screened personnel, and support for CJIS and IRS 1075).

Remote Desktop

Microsoft-managed environments

WARNING

Microsoft will be removing the use of Remote Desktop by customers and partners. Each environment will first have administrator access removed, but still allow non-administrator access to the virtual machines. After this, all access will be removed. For each step of this phased removal, an email notification will be sent to the Notification list setup for each environment. All Remote Desktop access will be removed by November 2020.

Customers are required to complete additional setup to connect to virtual machines (VMs) through Microsoft Remote Desktop (RDP). This additional setup applies to all Microsoft-managed environments, including Tier 1 through Tier 5 sandboxes and add-ons. In order to connect to Tier 1 through Tier 5 sandbox environments, you must explicitly enable access (safe list) from your organization's IP address space. This can be done by a Lifecycle Services (LCS) user who has access to the **Environment** page (**Maintain** > **Enable Access**) where they can enter the IP address space that will be used to connect to the virtual machines through Remote Desktop. Access rules are either a single IP address (example: 10.10.10.10) or an IP address range (example: 192.168.1.0/24). You may add multiple entries at once as a semi-colon(;) separated list (example: 10.10.10.10;20.20.20.20;192.168.1.0/24). These entries are used to configure the Azure Network Security Group that is associated with your environment's virtual network. For more information, see Security rules.

IMPORTANT

Customers need to ensure that RDP endpoints are secured through explicit IP safe list rules as mentioned above. The IP safe list rules must adhere to the following conditions.

- IP safe list rules must NOT use asterisk/zero.
- Wide IP address ranges must NOT be used.
- IP address ranges must restrict to the customer's CORPNET.
- If computers outside the customer's CORPNET (such as a home office) are used to connect to sandbox environments, only the specific IP addresses of the computers used to connect to the sandbox environments must be added.
- Azure Datacenter IP address ranges must NOT be added.
- Public IP addresses, such as a coffee shop location, must NOT be added.
- IP safe list rules should be removed when not in use. Periodic review of environment IP safe list rules is recommended.

> Microsoft will run periodic tests on the Microsoft Managed environments validating that the environments are sufficiently restricted. Microsoft reserves the right to and will remove any IP Address safe list rules that violate the above guidelines, immediately without providing notice.

Partner/Customer managed environments

By default, Remote Desktop is enabled for all non-Microsoft managed environments. We recommend that customers restrict access to any environments that belong to their subscriptions. This can be done by configuring Network Security Group rules on the environments directly in Azure Portal.

Windows Remoting (WinRM)

Windows Remoting (WinRM) is disabled on all environments. Although you can enable WinRM on environments that belong to your subscriptions through Azure Portal, we strongly recommend that you do not do this.

WARNING

Exceptions to enable WinRM will not be granted for any Microsoft-managed environments.

Availability

The guaranteed uptime for Finance and Operations apps is 99.9%. Planned downtime occurs once a month and lasts no longer than eight hours. Because the work completed during the downtime doesn't always take eight hours, we will always communicate the estimated amount of time that your environments will be down. For more information, see Get support for Finance and Operations apps or Lifecycle Services (LCS).

High-availability features

To ensure service availability, all production environments are protected by using default Azure high availability (HA) features. HA functionality provides ways to avoid downtime caused by the failure of a single node within a datacenter, and DR features protect against outages broadly impacting an entire datacenter. Azure availability sets are used to prevent single-point-of-failure events. For more information about Azure availability sets, see Use availability zones to protect from datacenter level failures. High availability for databases is supported through Azure SQL. For more information, see Overview of business continuity with Azure SQL Database.

Disaster recovery features

Production environments are configured with Azure disaster recovery support that includes the following:

- Azure SQL active-geo replication is configured for the Finance and Operations database of the production

environment. For more information about SQL replication, see Compare geo-replication with failover groups.

- Geo-redundant copies of Azure blob storage (containing document attachments) in other Azure regions. For more information, see Azure Storage redundancy.
- Same secondary region for the Azure SQL and Azure blog storage replication.

Only primary data stores are supported by replication. The Financial reporting services and Entity store database use transformed data from the primary database and must be generated after the recovery site has been set up and the Finance and Operations service has started.

Service availability in Azure Regions

Finance and Operations apps can be deployed into a subset of Microsoft Azure datacenters using Dynamics Lifecycle Services (LCS). Azure is generally available in datacenters and geographical locations around the world. With Finance and Operations apps, customers can specify the region or datacenter where their customer data will be stored. Microsoft may replicate data to other regions for data durability, but we will not replicate or move customer data outside the geographical location. For more details, see the Service description white paper.

> **IMPORTANT**
>
> Regardless of where customer data is stored, Microsoft does not control or limit the locations from which customers or their end-users may access it. For more information, see International availability of Dynamics 365.

Upcoming changes to region availability

Dynamics 365 solutions consist of a collection of multiple services. Looking across Dynamics 365 applications, the Power Platform and the Azure services that they both depend on, the required matrix of services is quite large and growing. We have locked on a strategy of selecting a subset of data center regions across the globe to simplify ensuring that we have availability of the full portfolio of required services. Our plan is to optimize to have minimal latency between the component services of a solution and as a result, we are focused on having the full portfolio of services available in each of the designated data centers.

Additionally, the Finance and Operations architecture is being enhanced to build on self-service for greater elasticity, stronger reliability, and more seamless maintenance. Customers gain material efficiency by having deeper self-service deployments in fewer data centers. This transition also benefits from selecting a subset of Azure regions. To that effect, the regional availability of Finance and Operations apps will now be **limited to East US, West US, and Central US in North America** for all new projects. For a list of the latest supported regions, see International availability of Dynamics 365.

Support for East US2, West US2, West Central US, North Central US, and South Central US will continue to be available for projects and environments that currently have their data stored in those regions on Microsoft-managed environments.

> **NOTE**
>
> Microsoft will work with customers to move them to an appropriate data center beginning October 19, 2020. This will happen in a phased approach. Select customers will receive advance notification before we migrate them to a supported region.

If there are other customer workloads that are not part of the Dynamics 365 or Power Platform family that also require proximity to the Dynamics 365 and Power Platform services, Microsoft will work with customers to coordinate a plan for the overall migration. For more information, see Cloud deployment overview: Frequently asked questions.

Frequently asked questions

Why does the status display 'Maintenance' on my environment in LCS?

To provide the best experience and performance, Microsoft performs maintenance operations on your environment. During some of these maintenance operations, your environment status may display one of the following statuses:

- Preparing for maintenance
- Prepared for maintenance
- Maintenance in progress

While your environment is in this state and until the status returns to 'Deployed', you will not be able to perform any lifecycle operations, such as package applications. There will be no impact to Finance and Operations apps. Users can continue with normal operations without any service interruption. You will receive an email notification before any maintenance operation puts your environment in this state.

How do I connect to the SQL database on my Sandbox environment?

To connect to the SQL database in your Sandbox environment, follow the steps in Enable just-in-time access.

How do I access a development instance?

For information about how to access development instances, configure on-premises development VMs, and find configurations settings for developers and administrators, see Deploy and access development environments.

How do I deploy a demo environment?

A demo environment includes only Microsoft demo data. You can use a demo environment to explore default features and functionality. For more information, see Deploy a demo environment.

How do I move my customizations between environments?

To move customizations from a development to a sandbox or production environment, see Create deployable packages of models

Can I bring my own domain name?

You can bring your own domain name if it is running Azure Active Directory (AAD), and the administrator of your AAD instance has enabled the Finance and Operations apps within their AAD. This is usually done through the office email, after you buy a license. When you click the link to accept the offer, AAD is set up for you.

Can I add guest AAD accounts as users?

You can add guest AAD accounts if you have correctly configured them within Azure Active Directory, and enabled the Finance and Operations apps within your AAD.

Why am I no longer able to see the Private AOS machines in one or more of my Tier 2 through Tier 5 Sandbox environments?

The Private AOS VMs were part of your environment configuration as they were needed to secure communication between the AOS and BI machines in the past. With recent updates, all communication between AOS and BI machines are secure directly and no longer need the intermediary Private AOS machines. Therefore, we are in the process of rolling out removing the Private AOS machines. As we are removing the machines in batches, you may notice that only some of your environments have the Private AOS machines removed. This change will not impact functionality or security in any way and will be transparent to you.

Why am I no longer able to Remote Desktop into one or more of my Tier 1 through Tier 5 Microsoft managed Sandbox environments?

Microsoft managed Tier 1 through Tier 5 sandbox environments require Remote Desktop management endpoints to be restricted to specific IP Address sets (safe list). Microsoft regularly validates that the environments are sufficiently restricted. Microsoft reserves the right to immediately remove any IP Address safe list rules that violate the above guidelines without notice. You may not be able to Remote Desktop into your

environment for one of these reasons:

- Your current IP address is not in the safe list.
- Your IP has changed from the IP address listed in the safe list.
- Microsoft deleted the rule containing your IP address from the safe list because it violated a guideline.

To regain access to the environment, you will need to add the IP address of the computer from which you are connecting to. To do this, complete the steps Remote Desktop section earlier in this topic.

When will the availability of reduced regions go into effect for new onboarding?

Beginning August 1, 2020, new projects for Finance and Operations will be onboarded to the following regions:

- East US
- West US
- Central US

My environments are currently in the regions that will be deprecated. How will this change affect me?

We will deprecate support for the following regions only for new projects that will be onboarded on or after August 1, 2020:

- East US2
- West US2
- West Central US
- North Central US
- South Central US

This will not affect any environments that have their data stored in the deprecated regions before August 2020. In the near future there is a transition plan to move customers in the deprecated regions into the reduced regions.

I'm unable to redeploy an environment after deleting it, the environment slot is missing.

This is due to the license expiring, which means that you no longer have the minimum required licenses to obtain an environment slot. Please review your subscription status and then reactivate the expired license to enable the redeployment.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Self-service deployment overview

2/18/2021 • 3 minutes to read • Edit Online

> **IMPORTANT**
>
> Functionality noted in this topic will be made available to users based on the geographic location recognized by Microsoft Azure.

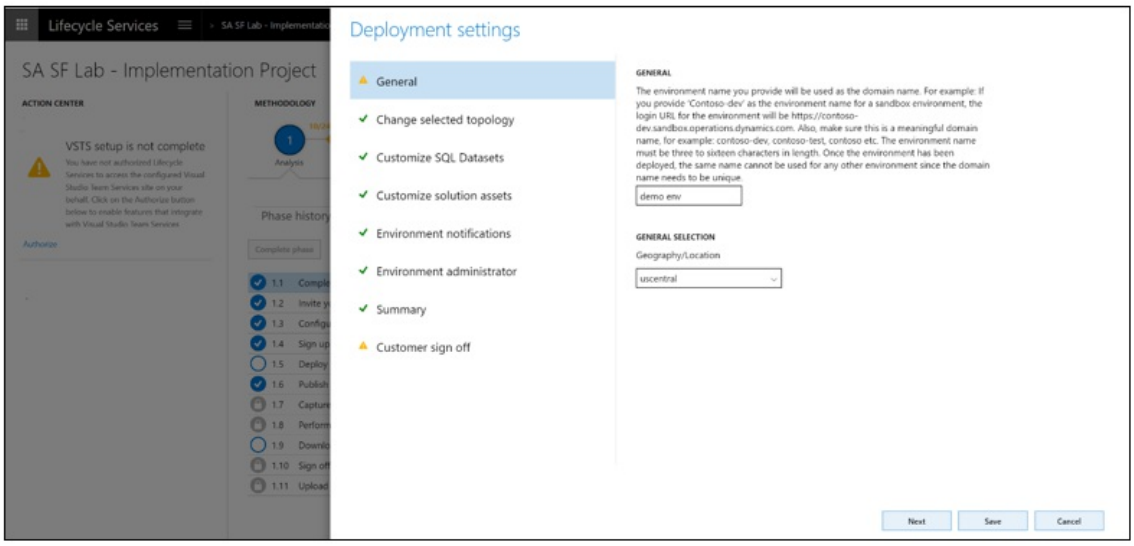
Self-service deployment is available for cloud environments. Self-service deployment enables easier deployment and significantly reduced deployment times.

> **IMPORTANT**
>
> The functionality for this feature will be released incrementally, based on your Microsoft Azure country/region. However, this functionality is currently available only for **new customers** who are in the process of signing up for Finance and Operations apps. There is no change in existing environments for current customers.
>
> Note that not all new customers will see this functionality. However, the number of new customers who have access to it will gradually increase.

What's new or changed

Customers using the self-service capabilities will see the following changes in their Lifecycle Services (LCS) experience.

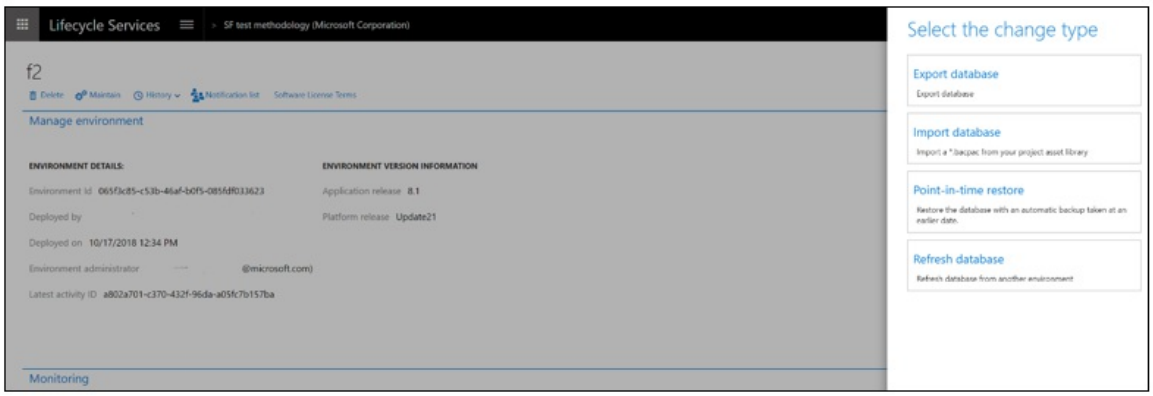
- Deployment is self-service and can be completed within an average time of 30 minutes. There are no longer lead times and wait times for deployment. You can control when you deploy, and verify that the environment is deployed. This experience is the same as the current experience. For more information, see Self-service deployment FAQ.



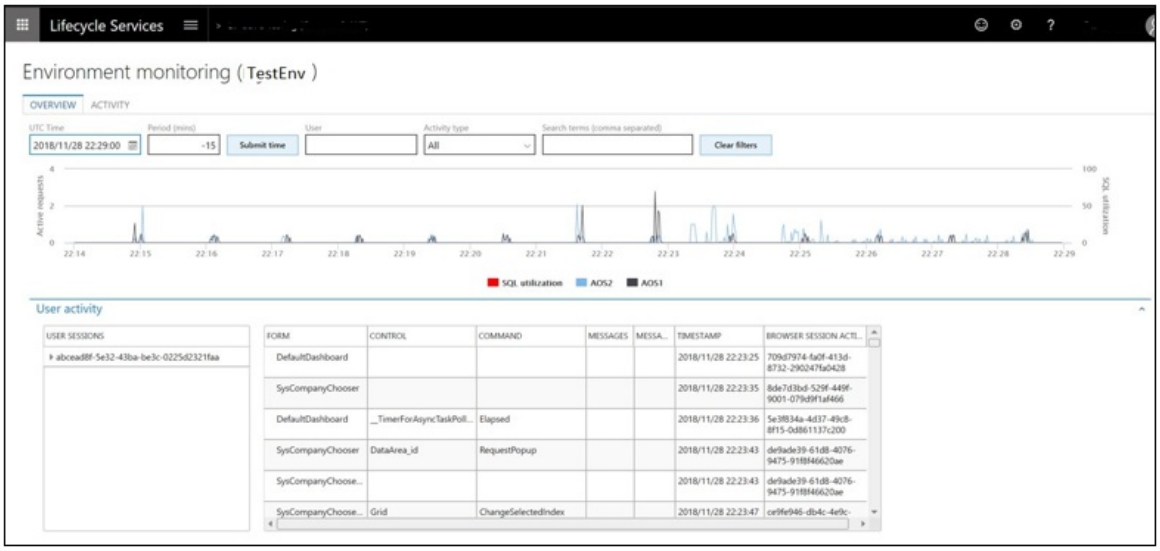
- You will no longer have remote desktop access to the Tier 2+ sandbox environments. All operations that need remote desktop access are now available as self-service actions. The following image shows some of the operations in the environment's **Maintain** > **Move database menu** option. For more information, see Maintenance operations for deployments.

IMPORTANT

Remote desktop access will be restricted only to environments deployed using the self-service deployment. There is no change to existing environments or existing customers.



- The diagnostics capabilities will remain the same, which enables troubleshooting without remote desktop access. For more information, see Troubleshoot environments deployed through self-service deployment.



- You will not have SQL Server access on Tier 2+. You will continue to have SQL database access using just-in-time access.
- You will need to provide a combined deployable package for customizations. That is, all custom extension packages, including ISV packages, must be deployed as a single software deployable package. You will not be able to deploy one module at a time. This was always a recommended best practice and is now enforced.
- The document preview experience has been improved to deliver greater fidelity with the printed output. Before this change, documents viewed on screen were displayed using an HTML viewer. Although the HTML format supported interactive functions like embedded drill-thru links and collapsible sections, this was not a true representation of the document rendered by the service. With the new embedded PDF Viewer, customers have access to a preview that's consistent with the printed documents. For more information, see Preview PDF documents with an embedded viewer.
- Custom fonts are no longer supported for document reports rendered using the built-in SSRS framework. Finance and Operations apps include access to hundreds of standard, business-ready fonts available for documents rendered by the cloud-hosted service. This portfolio will continue to grow as the service expands into new regions and industries. However, the service no longer supports the installation of custom fonts in customer environments. Requests to expand the collection of fonts supported by the

service will be considered on a case-by-case basis.

- The service no longer supports business logic defined using Visual Basic script embedded in SQL Server Reporting Services (SSRS) reports. Visual Basic expressions defined in Tablix controls used to format and evaluate data at runtime will continue to be fully supported. However, the service will ignore instructions defined in Visual Basic script functions. This change was necessary to improve the security and reliability of the service.
- Sub reports are no longer supported in document reports defined using the SSRS development tools. Application solutions that include sub reports will need to be recreated or replaced with solutions that take advantage of other reporting options supported by the service.

  > **IMPORTANT**
  > Support for sub report items has been re-introduced with the Platform update 36 release. Customers dependent on solutions that include properly formatted sub report items can transition to self-service deployments running on Platform update 36 or later.

> **NOTE**
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Deploy a new environment

2/18/2021 • 2 minutes to read • Edit Online

IMPORTANT

Functionality noted in this topic will be made available to users based on the geographic location recognized by Microsoft Azure.

This topic walks through the process of deploying sandbox (Tier 2 and above) and production environments with the self-service deployment experience. Refer to the following procedure to deploy these environments.

1. Select **Configure** on the project dashboard page.
2. Select the **Application** and **Platform** version for the environment that you want to deploy.
3. Provide a **unique name** for the environment.
4. Select the **region** where you want this environment to be deployed.
5. Choose whether you want to load **demo data** in your environment or if you want an **empty database**.
6. Select the **BPM library** that will be set as the Getting started library in the product.
7. Select from a list of available **AOT packages** (customization packages) on the Software Deployable tabs in the Asset Library if you want to apply customizations. Only packages generated from a build environment on version 8.1 and above should be selected. Applying a package from an incompatible version will have an adverse effect on the environment.
8. Specify **two user email addresses** that will receive **notifications** related to this environment. These users are in addition to the users who are already on the project team (such as an ISV or a partner).
9. Select the **email address** of the **user** that will be set as the **system administrator** in the product.
10. After you validate the configurations, click **Submit** to trigger the deployment.
11. If you plan to use channels, you must also Initialize Retail Cloud Scale Unit.

The environment deployment starts immediately and could take anywhere between **1-2 hours** to complete.

To closely monitor the deployment progress, you can view the **Environment details** page. The environment state should change to either **Deploying** or **Deployed/Failed**.

If the deployment **succeeds**, the environment state will be **Deployed**, and the user set as the environment administrator will be able to sign in to the environment.

If the deployment **fails**, then create a Microsoft support ticket and provide the **Last Activity ID** (available under **Manage environment** on the **Environment details** page) in the support ticket.

Delete an environment

You can delete an environment that is in the deployed state directly through the environment details page. To delete an environment, go to the environment details page and click the **Delete** button on the action bar. A confirmation dialog box will display asking you to enter the name of the environment that you want to delete. After you enter the environment name and select **Yes**, the deletion operation will start. When the deletion operation is complete, the **Configure** button for this environment will be enabled on the project dashboard if you want to redeploy the environment.

NOTE

To redeploy an environment, you will need to delete the environment and then deploy it again using the steps listed above.

IMPORTANT

For existing customers using channel functionality in the cloud, to ensure continued and uninterrupted support for your business, we require that you Initialize Retail Cloud Scale Unit no later than January 31, 2020. There is no action required for customers who exclusively use Store Scale Unit. Contact your Microsoft FastTrack Solution Architect if you require an extension.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Self-service deployment FAQ

2/18/2021 • 5 minutes to read • Edit Online

IMPORTANT

Functionality noted in this topic will be made available to users based on the geographic location recognized by Microsoft Azure.

This topic provides answers to some frequently asked questions about self-service deployment. Refer to the known issues list if your scenario is not listed here.

Why do I see only application version 8.1.1 and Platform update 21 and above when I try to deploy my sandbox environment using self-service deployment?

Currently, self-service deployment supports only application version 8.1.1 with Platform update 21 and above.

My development environment is on application version 8.1. Am I still able to move my customization to the sandbox environment?

Yes. Application version 8.1.1 is fully backward compatible with application 8.1.

What is the minimum supported application and platform version when trying to use the self-service deployment?

Application version 8.1 with Platform update 20 is the minimum supported version.

What do I do if deployment fails?

1. Delete the deployment.
2. If you want to reuse the same environment name, wait 5 minutes and try again. Otherwise, retry deploying with a new name.
3. If deployment fails again, log a Support ticket.

NOTE

Microsoft will add automatic retry logic in an upcoming release and make logs available.

What if my deployment fails with an “environment already exists” error?

You may be trying to reuse the environment name of a deployment that you just deleted. Wait 5–10 minutes before retrying the deployment.

I don't have Remote Desktop access to my sandbox environment. How do I perform critical actions that require Remote Desktop access

today?

Although you will no longer have Microsoft Remote Desktop access, you can continue to operate your Tier 2+ sandbox environments, because Microsoft is providing self-service capabilities and tools that you can use to perform the common critical actions that are described here.

Access the Azure SQL database

You can access the Microsoft Azure SQL database by following these steps.

1. From LCS, add a safe list of the IP address of the machine that you will use to connect to the Azure SQL database using SQL Management Studio.
2. Use LCS to request access to see the database credentials. You must provide a reason for requesting access.

As soon as you submit the request, it's automatically approved. Within a minute or two, you will be able to see the database access credentials on the LCS environment details page. You can use the credentials to connect to the SQL database.

> **NOTE**
>
> The credentials are valid for eight hours, and then they will expire. After the credentials expire, you will have to request access again.

Access log files

You can view and download all log files from the **Activity** tab on the LCS environment monitoring page.

Use perfmon tools

Although you can no longer use Remote Desktop and then use perfmon.exe to diagnose performance issues, you can monitor critical health metrics for CPU and memory consumption through LCS. In addition, you can run predefined queries on demand, and you can run predefined actions to mitigate performance issues on your Tier 2+ environments.

Access self-service logs

All logs that are related to self-service operations that are performed on the environment through LCS are available for download from LCS. These self-service operations include deployment, servicing, and database movement. You can download the log folders from the LCS environment history page.

Turn Maintenance mode on/off

Starting in the January 2019 release, you will be able to turn Maintenance mode in your environment on and off through a self-service action in LCS.

Restart services

Starting in the January 2019 release, you will able to restart services through a self-service action in LCS.

Configure the Regression suite automation tool

Microsoft is working on tooling that will let you update certificate thumbprints in the wif.config file without having to use Remote Desktop. This tooling is scheduled for release in February 2019. If you must use the Regression suite automation tool before then, log a support request.

I must perform one of the critical actions that are listed earlier in this topic, but the self-service feature isn't yet available. How do I get help?

Log a support ticket, and Microsoft will help you perform the action on your environment.

I don't have Remote Desktop access to my sandbox environment, and the critical action that I must perform isn't listed in this topic. How do I get help?

If your critical action isn't listed earlier in this topic, add a comment to this topic or log a documentation bug, and Microsoft will address your requirement.

For my Microsoft-managed environments, I have external components that have dependencies on an explicit outbound IP safe list. How can I ensure my service is not impacted after the move to self-service deployment?

With self-service migrations, we are changing the outbound IP addresses in regions where your environments are hosted. New outbound IP addresses are available so you can add them in preparation for the upcoming self-service migrations or post migrations.

- If none of your external components have dependencies on an explicit inclusion list of IPs or special handling of outbound IP addresses for routing or firewall, no action is required.
- If any of your external components have special handling for the outbound IP addresses to communicate to the AOS, add the new outbound IP addresses where the existing ones appear. Don't replace the existing IP addresses. You can find the new outbound IP addresses in the following list. For example, an outbound IP address may be explicitly included in a firewall outside your AOS, or an external service may have an allowed list that contains the outbound IP address for your AOS.

The inbound IP address to the AOS is dynamic. This can, and will, change over time as infrastructure changes occur.

> **NOTE**
> The outbound IP address from the AOS will remain static for the duration of an individual AOS session, which is currently listed to end in June 2021.

REGION	IP PREFIX
West US	52.250.195.128/26
East US	52.255.218.64/26
Central US	13.86.98.128/26
West EUR	51.105.159.192/26
West EUR-2	20.61.88.128/26
North EUR	52.155.160.192/26
UK West	51.137.139.0/26
UK South	51.11.26.192/26
Australia East	20.40.190.0/26

REGION	IP PREFIX
Australia SouthEast	20.40.165.192/26
Canada Central	20.151.60.0/26
Canada East	52.155.27.128/26
Brazil South	191.234.130.0/26
East Asia	52.229.231.64/26
South East Asia	20.44.247.0/26
Japan East	20.48.77.192/26
Japan West	20.39.179.192/26

Is there a potential impact on the environment's certificates?

Yes, if you are migrating from the previous non self-service deployment, your environment's certificate may be renewed due to infrastructure differences. Determine if there is any dependence on the certificates in your solution/integration and perform the needed actions after the migration.

Known issues with self-service deployment

2/18/2021 • 3 minutes to read • Edit Online

> **IMPORTANT**
> Functionality noted in this topic will be made available to users based on the geographic location recognized by Microsoft Azure.

This topic describes the known issues with self-service deployment.

Lifecycle Services (LCS)

Features not intended to be implemented

The following LCS features will not be implemented in self-service deployment.

- **System diagnostics** - All data and functionality provided by system diagnostics today will be available through other features in the product and LCS.
- **Service requests** - Service requests are being replaced with self-service actions.

Known issues in this release

Know issues are bugs that will be addressed in upcoming releases. Every 2 weeks there is a new release of LCS.

Finance and Operations apps

> **NOTE**
> Dynamics 365 Commerce is implemented in the modern deployment experience with the 10.0.10 release. For more information, see Create payment packaging for Application Explorer for self-service deployment.

Features not intended to be implemented

The following feature will not be implemented in self-service deployment.

- **Custom fonts** - Custom fonts are not supported. For more information, see Document Reporting Service in Dynamics 365 applications.
- **Customizations related to UI components on self service** - Customizations that do not use the standard Financial Reporting or SQL Server Reporting Services (SSRS) in Finance and Operations apps often take a dependency on UI components of the operating system where the AOS runs. Example dependencies include Windows fonts, web browsers such as Internet Explorer, or custom PDF rendering. We do not ensure the host operating system will include any support for font infrastructure, web browsers, or any general UI components. The host operating system will change when migrating to self-service infrastructure. If you have such dependencies and have additional questions, please contact Microsoft support.

Features no longer supported

The following feature is no longer supported with self-service deployment.

FTP

Customizations relying on FTP are not supported with self-service deployment and should consider the following actions.

- **Add retries** - This should be viewed as a short to medium term option. The current infrastructure design allows control and data calls to occasionally have matching IP. This design is subject to change.
- **Disabling the FTP requirement for control and data that is from the same IP** - Carefully evaluate the security implications for your situation in this case.
- **Remove the use of FTP** - For example, use Power Apps to pull the files in and make API calls into Finance and Operations apps to import the files using the Data Integration framework. For more information, see Choose a data integration strategy.
- **Use SFTP** - We do not recommend using static outbound IP addresses. For more information, see the Finance and Operations Static IP Addresses blog post.

Our general recommendation is to remove the use of FTP. Do not use a direct connection from Dynamics 365 to an SFTP. Instead, use a Logic App in between the two. With the Logic App you have two options:

- Use the native SFTP connector (as described in Monitor, create, and manage SFTP files in Azure Logic Apps), which still requires some port opening on the firewall to call the on-premises service. Consider that for Logic Apps, the list of IPs is much shorter than the entire region allow list and the limits and configuration in Power Automate.

- Use the "Local Filesystem" connector, as described in Outbound IP addresses, in combination with the on-premises data gateway. For more information, see Install on-premises data gateway for Azure Logic Apps. This solution completely removes the need for the IP allow list, which is deprecated in self-service deployment, while keeping a very high-level of security.

> **NOTE**
>
> For more information about deprecated features, see Removed or deprecated platform features and Removed or deprecated features from previous releases and Removed or deprecated features in previous releases.

Maintenance operations for deployments

2/18/2021 • 2 minutes to read • Edit Online

IMPORTANT

Functionality noted in this topic will be made available to users based on the geographic location recognized by Microsoft Azure.

This topic explains how to perform maintenance operations for an environment that was deployed by using the self-service deployment experience.

Restart services

You can use the restart services functionality to restart individual services that are associated with a Tier 2, Tier 3, Tier 4, or Tier 5 standard acceptance test (sandbox) environment that is deployed in a Microsoft subscription. The services that you can restart are **AOS service**, **DIXF (Data import export framework service)**, and **MR (Management Reporter service)**.

To restart a service, follow these steps.

1. In Microsoft Dynamics Lifecycle Services (LCS), on the environment details page, select **Maintain** > **Restart service**.
2. Select the service to restart, and then select **Confirm**.

   During the restart, the environment's status is updated to **Restarting service**, and you can't start any other maintenance operations. After the service has been restarted, the environment's status is returned to **Deployed**.

Maintenance mode

Finance and Operations apps includes a system-wide setting that is named maintenance mode. Maintenance mode gives system admins a safe way to make system changes that might affect system functionality. For example, configuration keys can be turned on or off. While maintenance mode is on, only the system admin and users who are assigned to the **Maintenance mode** user role can sign in to the system. By default, maintenance mode is turned off.

To turn maintenance mode on or off, follow these steps.

1. In LCS, on the environment details page, select **Maintain** > **Enable Maintenance mode**.

   The environment's status is changed from **Deployed** to **Servicing**. After maintenance mode has been turned on, the environment's status is updated to **In Maintenance**, and only the system admin can sign in.

2. After the system admin has finished making configuration changes, on the environment details page, select **Maintain** > **Disable Maintenance mode**.

   The environment's status is changed from **In Maintenance** to **Servicing**. After maintenance mode has been turned off, the environment's status is returned to **Deployed**. The environment history is updated to reflect the fact that the environment was put into maintenance mode.

Access database

Remote access is turned off for environments that were deployed by using the self-service deployment experience. During implementation, if you must connect to the database on your Tier 2, Tier 3, Tier 4 or Tier 5 standard acceptance test environments for troubleshooting purposes, access will be granted as it's required. The access won't be persistent.

To connect to a database, follow the instructions in Enable just-in-time access.

Update an environment

2/18/2021 • 6 minutes to read • Edit Online

IMPORTANT

Functionality noted in this topic will be made available to users based on the geographic location recognized by Microsoft Azure.

This topic walks through the process of applying updates to an environment that was deployed by using the self-service deployment experience.

IMPORTANT

In the next-generation infrastructure, updates are applied differently than they are applied in the current flow. *Whatever is provided in the package is applied to the environment, and it **overwrites** whatever is already present in that environment.* Therefore, you **must** create a single deployable package that contains all customizations and independent software vendor (ISV) solutions from your build environment. If the list of models in the environment differs from the list of models in in the package, you receive a warning before the update is applied. For information about how to create a single package, see Manage third-party models and runtime packages by using source control.

Supported package types

Because environments that are deployed by using the self-service deployment experience will be running on version 8.1 (October 2018) or a later version, the following package types are supported:

- **AOT deployable package** – A deployable package that is generated from application metadata and source code. This type of deployable package is created in a **build** environment.
- **Binary update package** – A deployable package that contains dynamic-link libraries (DLLs), and other binaries and metadata that the platform and application depend on. This type of package is released by Microsoft.

Development Application Lifecycle Management (ALM) flow

To apply code customizations to your sandbox and production environments, you must create a deployable code package from your build environment. For more information, see Create deployable packages of models.

Microsoft updates

To access Microsoft updates, go to the environment details page for your environment in Microsoft Dynamics Lifecycle Services (LCS). You will see a **single tile** that shows a cumulative binary update of all the application and platform fixes. To apply this update, select the package, and then select **Save Package** to save the Microsoft update to the project asset library.

Apply updates

Tier 1 sandbox/develop and test/demo/build environments

To apply updates to a Tier 1 sandbox/develop and test/demo/build environment that was deployed through LCS, follow the steps in Apply updates to cloud environments.

Tier 2 and above Standard Acceptance Test/sandbox environments

> **IMPORTANT**
> Package application causes system downtime. All relevant services will be stopped, and you won't be able to use your environments while the package is being applied.

Before you begin, verify that the deployable package has been uploaded to the project asset library in LCS, and that validations have succeeded.

After the package is in the project asset library, follow these steps to update your environment.

1. Open the environment details page for the environment where you want to apply the package.
2. Select **Maintain** > **Apply updates** to apply an update.
3. Enter a **unique name** for the update. You will use this name to identify the update that you want to move to the production environment.
4. Select the package to apply. Use the filter at the top to find your package. The list will include application and platform binary packages and application deployable packages that have passed validation from the asset library.
5. Select **Apply**.

   The status in the upper-right corner of the environment details page changes from **Queued** to **Servicing**. Then, when package application is completed, the status changes to **Deployed**.
6. After package application is completed, the environment history is updated. To view the environment history, select **History** > **Environment changes** on the environment details page.
7. You can also download the logs from the environment history page.
8. After the validations are completed, you can sign-off on the update from the environment history page by selecting either **Sign off** or **Sign off with issues**.

Production environment

To improve the reliability of updates that are applied to a production environment, a specific update that is done in the sandbox environment is designated ("marked") as a **release candidate** and moved to the production environment. This update contains both the base product and the customization. Therefore, both will be moved over when the selected update is applied to the production environment. This behavior differs from the current servicing behavior. Currently, you can move only the customization, but the base product **will not** change. In the case of a self-service deployment environment, the base product version **might change**, depending on the update that is marked as a release candidate in the sandbox environment.

> **IMPORTANT**
> Package application causes system downtime. All relevant services will be stopped, and you won't be able to use your environments while the package is being applied.

After you've successfully applied the update in the sandbox environment and are ready to move the update over to the production environment, follow these steps to mark an update as a release candidate.

1. Open the environment history page by selecting **History** > **Environment changes** on the environment details page.
2. Select the update to move over to the production environment.

3. In the details for the update, select **Mark as release candidate**.

   The **Is Release Candidate** option is set to **Yes**.

After you've marked an update as a release candidate, follow these steps to update your environment.

1. Open the environment details page for the production environment.
2. Select **Maintain > Update environment** to apply an update.
3. In the **Available sandboxes** list, select the source sandbox environment where the update was applied, validated, and marked as a release candidate.
4. In the grid, select the update to apply to the production environment. This grid shows only updates that have been marked as release candidates.
5. In the **Downtime start** field, select a date and time. The environment will be taken down for servicing at the specified time on the specified date. The **Downtime end** field is set to through hours from the start of the downtime.

   No lead time is required for this update.
6. Select **Schedule**. LCS runs validations to make sure that the selected update is applicable to the environment. To prevent downgrade of the environment, the update isn't allowed if its application version is lower than the current environment version. Additionally, customers are asked to confirm that they want the update to proceed.

   If the update is **successfully** scheduled, an email notification is sent to all project stakeholders.

   The status in the upper-right corner of the environment details page changes from **Queued** to **Servicing**. Then, when the update is completed, the status changes to **Deployed**.

   All stakeholders are notified of the progress of the operation.
7. After the update is completed, the environment history is updated. To view the environment history, select **History > Environment changes** on the environment details page.
8. You can also download the logs from the environment history page.
9. After the validations are completed, you can sign-off on the update from the environment history page by selecting either **Sign off** or **Sign off with issues**.

> **NOTE**
>
> If there is an on-going operation in the environment, or if the environment is already running on the same version or a later version, the scheduled update is canceled. When a scheduled update is canceled, an email notification is sent to all project stakeholders. Customers can also cancel an update by selecting **Cancel** on the environment details page. If customers want to reschedule or change an update, they can cancel the current operation and schedule a new one.

> **IMPORTANT**
>
> For environments that are deployed in the modern infrastructure stack, if servicing is unsuccessful, the environment is automatically rolled back. To learn why the operation was unsuccessful, you can download the logs from the environment history page.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Troubleshoot environments deployed through self-service deployment

2/18/2021 • 2 minutes to read • Edit Online

IMPORTANT

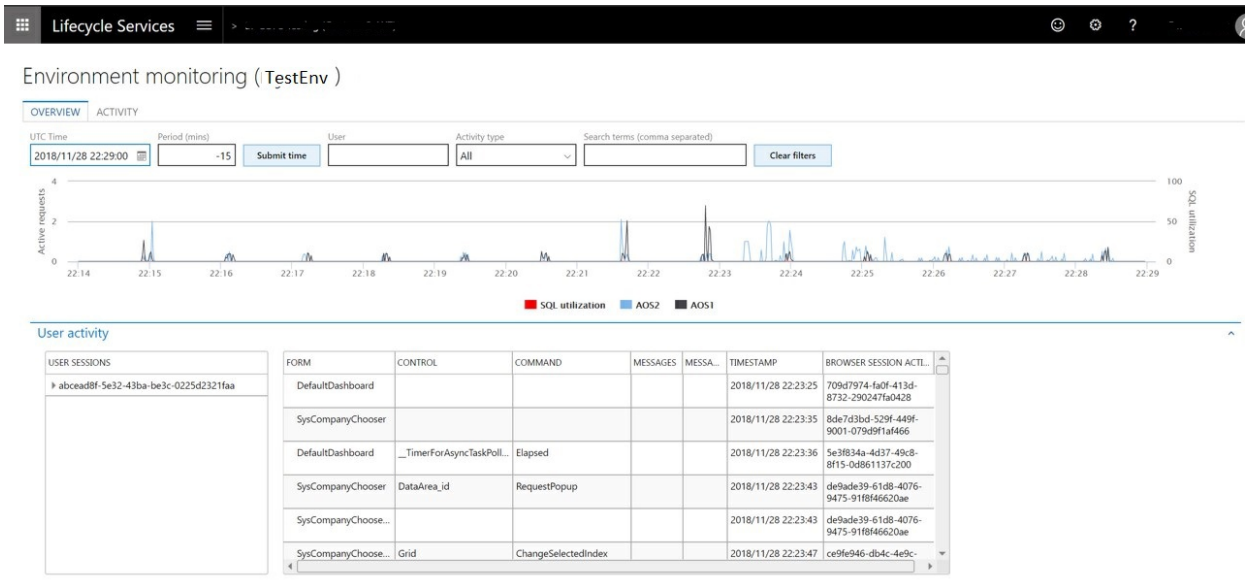
Functionality noted in this topic will be made available to users based on the geographic location recognized by Microsoft Azure.

This topic explains how you can troubleshoot issues on an environment that was deployed using the self-service deployment experience. When a user reports an issue, you can use various tools in Lifecycle Services (LCS) for troubleshooting. The rich set of telemetry data helps you build a storyboard view that shows what that user and other users were doing when the issue was reported.

To open the **Environment Monitoring** dashboard, follow the steps listed below:

1. Open LCS and navigate to the appropriate project.
2. In the **Environments** section, select the environment that you want to view, and then click **Full details**.
3. On the **Environment details** page, click **Environment monitoring** to open the Monitoring and diagnostics portal.

On the Environment Monitoring dashboard, you will see two tabs: **Overview** and **Activity**.



Overview tab

The **Overview** tab provides a storyboard view that shows how the environment was being used during a specific period. You can use the filters on this page to narrow the information logs. Here are some of the filters that are available:

- **Time duration**: Go back 60 minutes from the selected date and time.
- **User**: View a specific user's activities.
- **Search terms**: Create a search that is based on the issue that is being investigated.

In addition, you will also see two sections:

- The **User interaction** chart shows a user's activities on various machines in the environment and the SQL utilization trend.
- The **User activity** grid shows the various activities that users performed, based on their session timestamp. The active sessions display on the left side of the grid. For each session, the Form:Control:Command and the corresponding timestamp show when the action was taken. You can trace the exact steps that the user was taking in the information presented in this grid.

> **IMPORTANT**
> The Overview tab will also include the CPU and memory health metrics to help with diagnostics. This feature is currently not available but will be added soon.

Activity tab

The **Activity** tab shows a predefined set of queries for advanced troubleshooting. This gives you access to the raw information logs. You can then export the logs to do more advanced analysis. The following types of queries are available:

- User-related errors
- Slow queries
- Deadlocks
- Crashes

> **NOTE**
> The data shown on the **Overview** and **Activity** tabs is only retained for 30 days.

> **IMPORTANT**
> The Environment monitoring will also include advanced SQL troubleshooting tools to diagnose and mitigate performance issues. This feature is currently not available but will be added soon.

> **NOTE**
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Complete the Azure Resource Manager (ARM) onboarding process

2/18/2021 • 4 minutes to read • Edit Online

This topic explains how to complete the Azure Resource Manager (ARM) onboarding process for your connectors.

To deploy Microsoft Azure Resource Manager (ARM) topologies, you must complete the ARM onboarding process for your connectors. To start the onboarding process, you must have the following items:

- The Azure subscription ID that you're deploying to
- Ownership of the Azure subscription, or access to the subscription owner, so that you can add contributor workflows, and the Upload Management certificate
- The tenant administrator, to work through the admin consent workflow

ARM onboarding process

You can consider ARM onboarding a two-step procedure, where each step has its own sub-procedures. You must complete all these procedures for every subscription that you add to the Microsoft Dynamics Lifecycle Services (LCS) project.

1. Authorize the LCS Deployment Service (DSU) to work on the Azure subscription.
    a. Authorize the workflow.
    b. Set the contributor workflow.
2. Enable the Azure subscription to deploy ARM resources.
    a. Enable the Azure connector, and add an LCS user.
    b. Optional: Upload the Management certificate.
    c. Configure deployment settings.

Authorize the LCS DSU to work on the Azure subscription

Complete the following procedures to authorize the LCS DSU to work on the Azure subscription.

Authorize the workflow

The administrator of the tenant must complete the following procedure.

1. In LCS, on the **Project** page, in the **Environments** section, click **Microsoft Azure settings**.
2. On the **Project settings** page, on the **Azure connectors** tab, in the **Organization list** group, click **Authorize** to start the ARM Contributor workflow. This workflow sets up permissions for the DSU, so that it can deploy to your subscription on your behalf.
3. On the **Grant admin consent** page, click **Authorize**. Then sign in by using the administrator account of the Azure subscription that you must connect to, and click **Accept**. The authorization is now shown as completed.

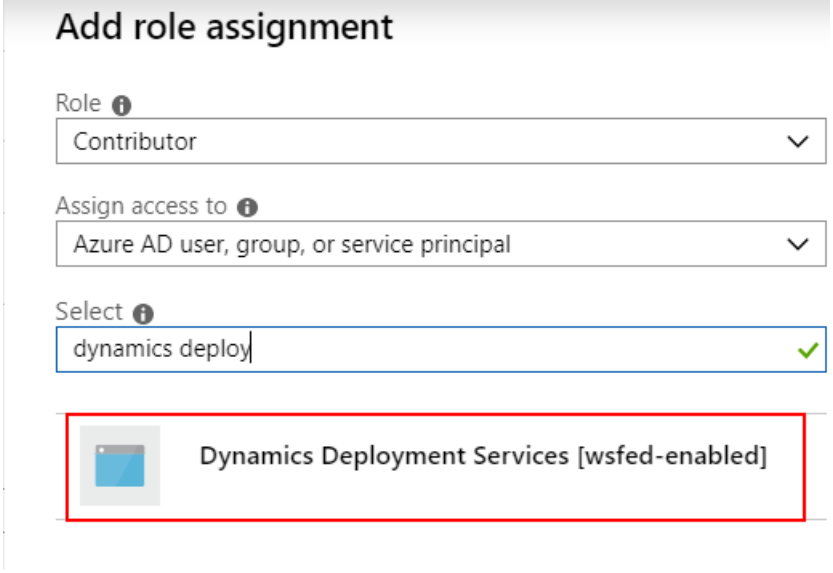
Set the contributor workflow

Follow these steps to assign the **Contributor** role to the **Dynamics Deployment Services [wsfed-enabled]** application.

1. In the Azure portal, on the **Subscription** tab, select the Azure subscription, and then click the **Access Control (IAM)** line item.

2. Click **Add**, select **Add role assignment**. In the dialog box, set **Role** to **Contributor** and set **Assign access to** to **Azure AD user, group, or service principal**. In the **Select** field, search for and select **Dynamics Deployment Services [wsfed-enabled]**. Select **Save**.

> **NOTE**
>
> Some Azure subscriptions have a **Users** section instead of an **Access control (IAM)** section. In this case, in the **Add users** dialog box, in the **Select** field, enter **Dynamics Deployment Services [wsfed-enabled]**, and then select **Select**.



3. On the **Role assignments** tab, the App is assigned as a **Contributor**.

> **NOTE**
>
> If Dynamics Deployment Services [wsfed-enabled] doesn't appear, the authorize process hasn't been completed, or it was completed on another Azure subscription.

Enable the Azure subscription to deploy ARM resources

Complete the following procedures to enable the Azure subscription to deploy ARM resources.

Enable the Azure connector and add an LCS user

Follow these steps to enable the Azure Connector and, as required, add an LCS user.

1. In LCS, on the **Project** page, in the **Environments** section, click **Microsoft Azure settings**.
2. On the **Project settings** page, on the **Azure connectors** tab, under **Azure connectors**, click **Add**.

> **NOTE**
>
> If you're enabling ARM for an existing connector, click **Edit**.

3. Enter the connector name, enter the Azure subscription ID to deploy to, and set the **Configure to use Azure Resource manager** option to **Yes**.
4. In the **Azure subscription AAD Tenant domain** field, enter the domain name of the Azure subscription account admin, and then click **Next**.
5. Authorize access to the subscription, either by adding the LCS user to the Azure subscription or by using the Management certificate. **Important:** If you're adding an LCS user, continue with step 6. If you must upload a Management certificate, don't complete steps 6 through 8 of this procedure. Instead, complete the next procedure, "Upload the Management certificate."

6. In the Azure portal, on the **Subscription** tab, select the Azure subscription, and then click the **Access Control (IAM)** line item.
7. In the **Access Control (IAM)** dialog box, click **Add**, select **Contributor**, and then click **OK**.
8. In the **Add users** dialog box, in the **Select** field, enter the LCS user, and then press Enter.

> **NOTE**
>
> You must specifically enter a user. You can't just add a group that the user is a member of. When the **Users** page opens, you can see that the user is assigned as a **Contributor**.

Upload the Management certificate

Complete this procedure only if you didn't complete steps 6 through 8 of the previous procedure, "Enable the Azure Connector and add an LCS user."

1. In LCS, on the **Microsoft Azure setup** page, click **Download**. Make a note of the location of the certificate file that is downloaded. You will use this information to upload the certificate to the Azure subscription.
2. In the Azure classic portal, in the left pane, click **Settings**.
3. Filter to the Azure subscription that is used, and then, on the **Management certificates** tab, click **Upload**.
4. Select the Management certificate that you downloaded in step 1, and then click **OK**.

Configure deployment settings

1. In LCS, on the **Project** page, in the **Environments** section, click **Microsoft Azure settings**.
2. On the **Microsoft Azure setup** page, select the region to deploy to, and then click **Connect**. The ARM onboarding flow is now completed. You should now see that the subscription has been added to the **Azure connectors** list. Additionally, a check mark should appear under **ARM Enabled**.

Azure ExpressRoute and Finance and Operations apps

2/18/2021 • 2 minutes to read • Edit Online

Customers can use Microsoft Azure ExpressRoute with Finance and Operations apps to connect to their on-premises infrastructure. This topic provides the information that you need to get started with ExpressRoute.

Microsoft Azure ExpressRoute lets you create dedicated, readily available, highly reliable, low latency connections between Azure datacenters and your on-premises locations. An ExpressRoute circuit is a logical connection between a customer's on-premises network and Microsoft cloud services through a connectivity provider. ExpressRoute is configured separately from Finance and Operations apps. To get an ExpressRoute circuit for your implementation, you must contact a network service provider directly. After ExpressRoute is configured, in addition to connecting to Finance and Operations apps, customers can connect to apps such as Microsoft 365 and supported Azure services, such as connecting to virtual machines and cloud services deployed in virtual networks. To learn more about other supported services, see ExpressRoute FAQ. Before purchasing an ExpressRoute circuit, you will need to know the following information:

- The datacenter that your Finance and Operations apps are located in.
- The region where you will be connecting from.

This information is necessary to determine whether a standard or premium offering of ExpressRoute is required.

Resources for getting started

- ExpressRoute service page
- ExpressRoute technical overview
- ExpressRoute partners and peering locations
- ExpressRoute pricing

NOTE

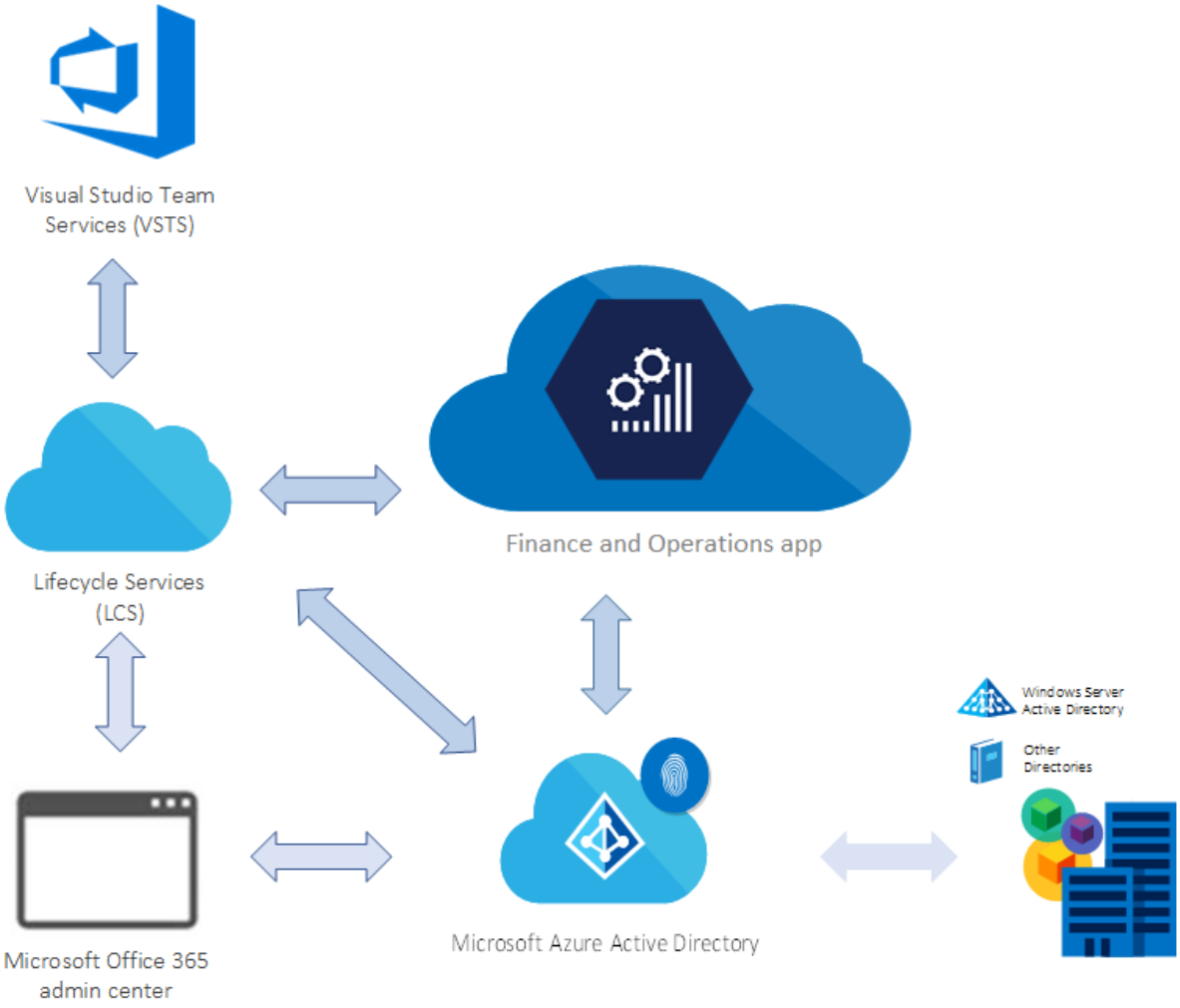
Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Finance and Operations application architecture

2/18/2021 • 4 minutes to read • Edit Online

The Finance and Operations application cloud architecture contains all the elements that are common to all Microsoft cloud offerings, as described in Subscriptions, licenses, accounts, and tenants for Microsoft's cloud offerings. Beyond this, it also includes services that automate software deployment and provisioning, operational monitoring and reporting, and seamless application lifecycle management.



The cloud architecture consists of these conceptual areas:

- **Subscription** – A subscription to Finance and Operations apps gives you an online cloud environment (or multiple environments) and experience.
- **Licenses** – Customers must purchase subscription licenses (SLs) for their organization, or for their affiliates' employees and on-site agents, vendors, or contractors who directly or indirectly access Finance and Operations apps. These apps are licensed through Microsoft Volume Licensing and the Microsoft Cloud Solution Provider (CSP) program. For more information, download the latest Microsoft Dynamics 365 Licensing Guide from Dynamics 365 pricing.
- **Tenant** – In Microsoft Azure Active Directory (AAD), a tenant represents an organization. It's a dedicated instance of the AAD service that an organization receives and owns when it creates a relationship with Microsoft (for example, by signing up for a Microsoft cloud service, such as Azure, Microsoft Intune, or Microsoft 365). Every AAD tenant is distinct and separate from other AAD tenants. For more information

about AAD tenants, see How to get an Azure Active Directory Tenant.

A tenant houses the company's user information. This information includes passwords, user profile data, permissions, and related information. The tenant also contains groups, applications, and other information that pertains to an organization and its security.

The tenant is created when customers sign up for their first subscription to any Microsoft online service, such as Microsoft 365, Microsoft Dynamics 365, or Azure. Any later subscriptions to the same online services or other online services can be grouped within the same tenant.

An organization can have multiple AAD tenants. If there are multiple tenants, make sure that any subscriptions for Finance and Operations apps are associated with the correct tenant.

- **Azure Active Directory (AAD)** – AAD is the multi-tenant, cloud-based directory and identity management service from Microsoft that combines core directory services, application access management, and identity protection in a single solution. For more information, see Azure Active Directory. Finance and Operations apps use AAD as the store for identity. Access to AAD is provided as part of a subscription to Finance and Operations apps.
- **Microsoft 365 admin center** – Microsoft 365 admin center is the subscription management portal that Microsoft 365 provides for administrators. It's used to provide management functions for users (AAD) and subscriptions. As part of these management functions, it provides information about service health. For more information, see About the Microsoft 365 admin center.

  > **NOTE**
  > You don't have to have an Microsoft 365 license to deploy Finance and Operations apps. However, you might require a license for specific Office integration scenarios. For more information, see Office integration overview.

- **Microsoft Dynamics Lifecycle Services (LCS)** – LCS is a collaboration portal that provides an environment and a set of regularly updated services that can help you manage the application lifecycle of your implementations. For more information, see Lifecycle Services resources. After you purchase and activate a subscription for a Finance and Operations app, an **Implementation project** workspace is provisioned in LCS when the tenant administrator signs in for the first time.

  > **NOTE**
  > An implementation project is an LCS project for the cloud service. As a Microsoft partner, you can also provision non-implementation LCS projects for your own purposes. For more information, see Lifecycle Services (LCS) for Finance and Operations apps partners.

- **Finance and Operations apps** – Finance and Operations apps are deployed through LCS. Various topologies are available: development/test/build, acceptance test, performance test, and high-availability production. For more information about the various topologies, download the latest Microsoft Dynamics 365 Licensing Guide from Dynamics 365 pricing.
- **Microsoft Azure DevOps** – Azure DevOps is used primarily for code version control, development, and to deploy a build environment. Azure DevOps is also used to track support incidents, such as work items in Azure DevOps that are submitted to Microsoft through Cloud-powered support, and to integrate the Business process modeler (BPM) library hierarchy into your Azure DevOps project as a hierarchy of work items. Azure DevOps is also used during code upgrade.

"Under the hood," Finance and Operations apps use many features of the Azure platform, such as Azure Storage, networking, monitoring, and Azure SQL Database, to name just a few. Shared services put into operation and orchestrate the application lifecycle of the environments for participants. Together, Azure functionality and LCS

offer a robust cloud service.

> **NOTE**
>
> Although many features of the Azure platform are used, you don't have to have an Azure subscription to deploy Finance and Operations apps in the Microsoft-managed cloud. You must have an Azure subscription only if you want to deploy Finance and Operations apps cloud-hosted environments in your own Azure subscription.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Apply updates to cloud environments

2/18/2021 • 9 minutes to read • Edit Online

This topic describes how you can use Microsoft Dynamics Lifecycle Services (LCS) to automatically apply updates to cloud environments.

> **IMPORTANT**
> Updates are applied using deployable packages. Applying updates causes system downtime. All relevant services will be stopped, and you won't be able to use your environments while the package is being applied. You should plan accordingly.

Supported environments

All environments deployed through Lifecycle Services are supported.

> **NOTE**
> If you have a build environment, you can only use LCS to apply Binary updates and Data upgrade packages. You can't use LCS to apply an Application Deployable package.

For other environments (listed below), you must use Remote Desktop Protocol (RDP) to connect to the environment and install from the command line. For information about manual package deployment, see Install deployable packages from the command line.

- Local development environments (Downloadable virtual hard disk [VHD])
- Multi-box dev/test environments in Microsoft Azure (Partner and trial projects)

Key concepts

Before you begin, you should understand *deployable packages*, *runbooks*, and the *AXInstaller*. A deployable package is a unit of deployment that can be applied in any environment. A deployable package can be a binary update to the platform or other runtime components, an updated application (AOT) package, or a new application (AOT) package. The AXInstaller creates a runbook that enables installing a package. For more details, see Packages, runbooks, and the AXUpdateInstaller in depth at the end of this topic.

Supported package types

- **AOT deployable package** – A deployable package that is generated from application metadata and source code. This deployable package is created in a development or build environment.
- **Application and Platform Binary update package** – A deployable package that contains dynamic-link libraries (DLLs) and other binaries and metadata that the platform and application depend on. This is a package released by Microsoft. This is available from the **All binary updates** tile from LCS.
- **Platform update package** – A deployable package that contains dynamic-link libraries (DLLs) and other binaries and metadata that the platform depend on. This is a package released by Microsoft. This is available from the **Platform binary updates** tile from LCS.
- **Commerce deployable package** – A combination of various packages that are generated after the Commerce code is combined.
- **Merged package** – A package that is created by combining one package of each type. For example, you can

merge one binary update package and one AOT package, or one AOT package and one Commerce deployable package. The packages are merged in the Asset library for the project in LCS.

> **NOTE**
>
> A binary package and a Commerce deployable package can't be included in the same merged package.
>
> For information about how to download an update from LCS and what you see in the tiles based on your environment version, see Download updates from Lifecycle Services (LCS).
>
> If your environment is on an application version 8.1 and later, then the **Platform Update package** does not apply to your environment. Starting with 8.1 and later releases, **Application and Platform Binary update package** is the one that applies since application and platform will be combined into a single cumulative package and will be released by Microsoft. Also note that you will no longer be applying granular X++ hotfixes and will get all application and platform updates together. This means that on the environment details page, clicking on **View detailed version information** will not have details on the granular hotfixes or KBs applied as there is no way to apply them.

Prerequisite steps

- **Make sure that the package that should be applied is valid.** When a package is uploaded to the Asset library, it isn't analyzed. If you select the package, the package status appears in the right pane as **Not Validated**. A package must pass validation before it can be applied in an environment by using the following procedures. The status of the package will be updated in the Asset library to indicate whether the package is valid. We require validation to help ensure that production environments aren't affected by packages that don't meet the guidelines.

  There are three types of validations:

  - Basic package format validations
  - Platform version checks
  - Types of packages
- **Make sure that the package is applied in a sandbox environment before it's applied in the production environment.** To help ensure that the production environment is always in a good state, we want to make sure that the package is tested in a sandbox environment before it's applied in the production environment. Therefore, before you request that the package be applied in your production environment, make sure that it has been applied in your sandbox environment by using the automated flows.
- **If you want to apply multiple packages, create a merged package that can be applied first in a sandbox environment and then in the production environment.** Application of a single package in an average environment requires about 5 hours of downtime. To avoid additional hours of downtime when you must apply multiple packages, you can create a single combined package that contains one package of each type. If you select a binary package and an application deployable package in the Asset library, a **Merge** button becomes available on the toolbar. By clicking this button, you can merge the two packages into a single package and therefore reduce the total downtime by half.
- **Make sure that the Application binary update package is applied to your dev/build environment before it is applied to your sandbox and production environment** - If the application binary package is applied directly to your Tier 2+ sandbox environment but is not applied on your dev/build environment, the next time you move an AOT package from dev/build box (which does not have the same application binaries as your sandbox environment) to sandbox, some of the application binaries will be overwritten with what is in your dev/build environment. This could result in a regression of the version of your sandbox environment.

Apply a package to a non-production environment by using LCS

Before you begin, verify that the deployable package has been uploaded to the Asset library in LCS.

1. For a binary update, upload the package directly to the Asset library. For information about how to download an update from LCS, see Download updates from Lifecycle Services (LCS). For an application (AOT) deployable package that results from an X++ hotfix, or from application customizations and extensions, create the deployable package in your development or build environment, and then upload it to the Asset library.
2. Open the **Environment details** view for the environment where you want to apply the update.
3. Click **Maintain** > **Apply updates** to apply an update.
4. Select the package to apply. Use the filter at the top to find your package.
5. Click **Apply**. Notice that the status in the upper-right corner of the **Environment details** view changes from **Queued** to **In Progress**, and an **Environment updates** section now shows the progress of the package. You can refresh the page to check the status.
6. Continue to refresh the page to see the status updates for the package application request. When the package has been applied, the environment status changes to **Deployed**, and the servicing status changes to **Completed**.

Apply a package to a production environment by using LCS

In a production environment, customers can schedule a downtime for when they want the update to be applied.

> **IMPORTANT**
> An important prerequisite for applying a package to a production environment is that the package must be successfully applied to at least one sandbox environment in the same project.

1. After the update is successfully applied in a sandbox environment, go to the project's asset library. On the **Asset library** page, select the **Software deployable package** tab, select the package that you want to move to production, and click **Release candidate**. This indicates that this package is ready for production deployment.
2. Open the **Environment details** view for the production environment where you want to apply the package.
3. Select **Maintain** > **Apply updates** to apply the package.
4. Select the package to apply in your production environment, and then click **Schedule** to submit a request to apply it.

   > **NOTE**
   > The list of packages includes only the packages that have been successfully signed off in the sandbox environment, and that have been marked as release candidates.

5. Specify the date and time to schedule the package application. Click **Submit**, and then click **OK** to confirm. Note that your environments will be unavailable to perform business while the package is being applied.
6. At the scheduled downtime, package deployment will start.
7. After the environment is serviced, you can monitor the status. The **Servicing status** field indicates the status of package application. Additionally, a progress indicator shows the number of steps that have

been run, out of the total number of steps that are available.

8. After the deployment is successfully completed, the **Servicing status** field is set to **Completed**.
9. If package application isn't successfully completed, Microsoft will investigate the issue. The **Servicing status** field will indicate that package application has failed. The environment will be rolled back to a good state.

Troubleshoot package deployment failures

If package deployment fails, see the Troubleshoot package application issues topic.

Applying updates and extensions

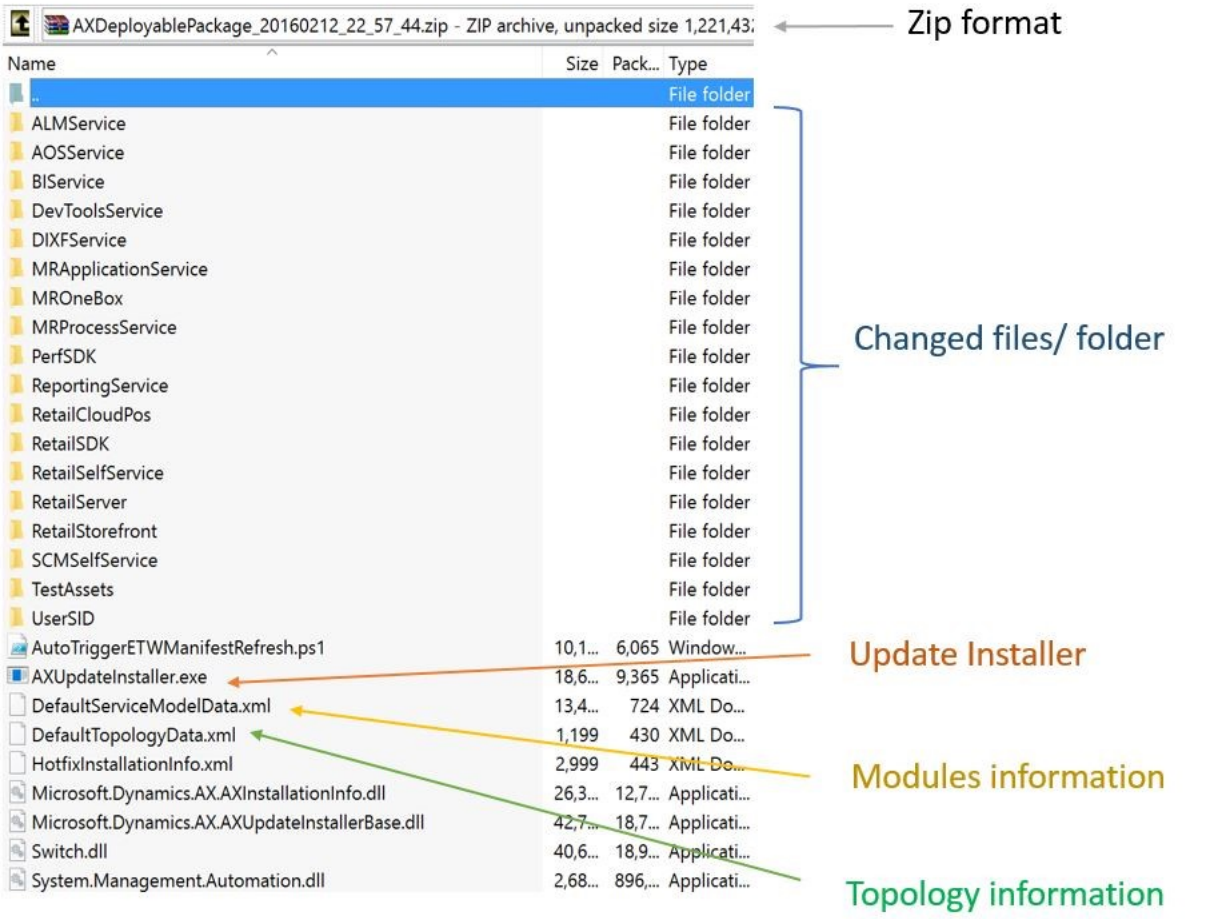
If you are updating a Tier-2 Sandbox or Production environment on application version 8.1.2.x or newer and have initialized Cloud Scale Unit, you will also need to update Commerce channel components. For more information, see Update Retail Cloud Scale Unit.

If you're using components (such as Modern POS), after you've applied updates and extensions in your environment, you must also update your in-store components. For more information, see Configure, install, and activate Modern POS (MPOS).

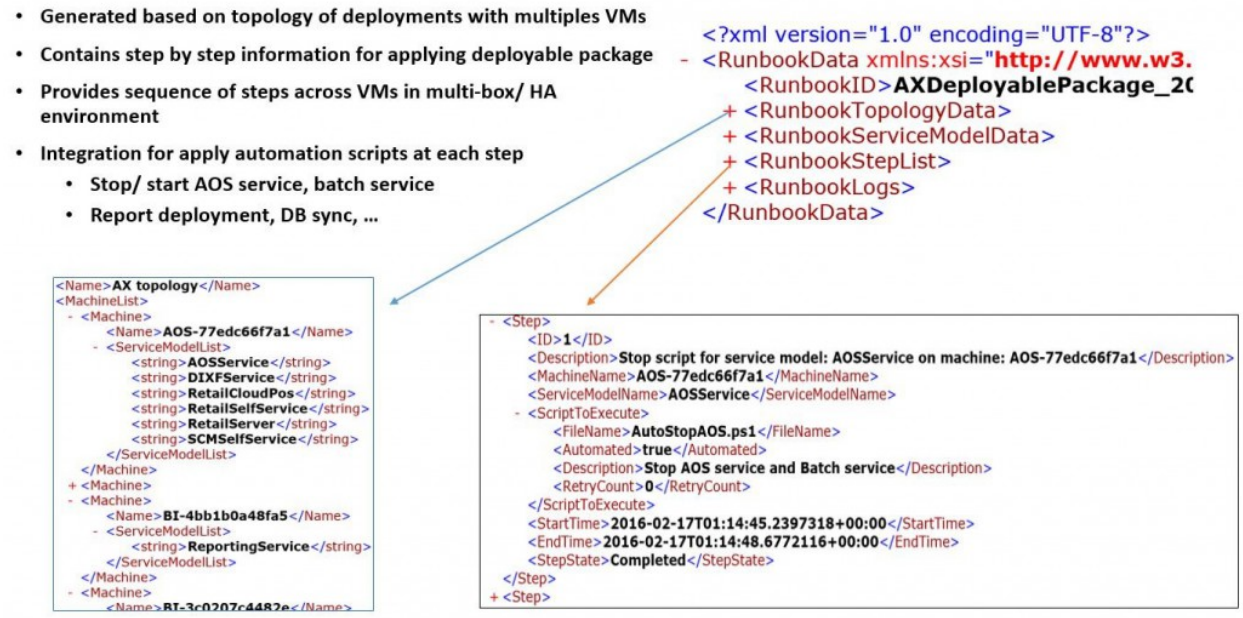
Packages, runbooks, and the AXUpdateInstaller in depth

Deployable packages, runbooks, and the AXUpdateInstaller are the tools you use to apply updates.

Deployable package – A deployable package is a unit of deployment that can be applied in an environment. A deployable package can be a binary update to the platform or other runtime components, an updated application (AOT) package, or a new application (AOT) package. Deployable packages downloaded from LCS or created in a development environment cannot be applied across product types. For example, a Finance deployable package cannot be applied in a Commerce app environment, and vice versa. If you have an existing customization for a Finance and Operations app that is compatible with the Commerce app, and you would like to apply it to a Commerce environment, you will need to re-package your source code in a Commerce development environment, and conversely if moving in the other direction.



Runbook – The deployment runbook is a series of steps that are generated in order to apply the deployable package to the target environment. Some steps are automated, and some steps are manual. AXUpdateInstaller lets you run these steps one at a time and in the correct order.



AXUpdateInstaller – When you create a customization package from Microsoft Visual Studio or a Microsoft binary update, the installer executable is bundled together with the deployable package. The installer generates the runbook for the specified topology. The installer can also run steps in order, according to the runbook for a specific topology.

Additional resources

Install deployable packages from the command line

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Apply updates and extensions to Commerce Scale Unit (cloud)

2/18/2021 • 2 minutes to read • Edit Online

If you are updating a Tier-2 sandbox or production environment on application version 8.1.2 or newer and have initialized Commerce Scale Unit (CSU), you will also need to update channel components. This topic shows how to apply updates and extensions to CSU.

Updates to CSU are cumulative. This means that any update that you apply will include all previously released changes. Applying a Dynamics 365 Commerce deployable package for extensions is also a cumulative process and will replace the previously deployed version of the extension.

Prerequisites

Before you proceed, you must first apply updates and extensions (if applicable) to the environment. For more information, see Apply updates to cloud environments.

To update CSU, complete the following steps for each:

1. On the **Environment details** page, go to **Environment features > Retail and Commerce**.
2. On the **Commerce deployment setup** page, select **Update**.
3. In the selection panel, select the version to update to.
4. You can choose to update to the newest service update to access the latest features, or you can update to the latest quality update to apply quality improvements for the currently deployed service update. For more information, see Download updates from Lifecycle Services (LCS).
5. You can choose to apply an extension at the same time.

To apply an extension to a CSU, complete the following steps:

1. On the **Commerce deployment setup** page, select **Apply Extension**.
2. In the selection panel, select the extension to apply.

> **NOTE**
> You must first upload the Commerce deployable package to the project asset library in Microsoft Dynamics Lifecycle Services (LCS) before you can select to deploy it on the **Commerce deployment setup page** in LCS.

Both **Apply updates** and **Apply extension** operations will involve a period of downtime that may last up to 1 hour, or in some cases up to 2 hours or more. For example, when updating non-US locations of CSU, large data volumes, or complex schema updates. For a realistic estimate of the downtime duration, note the downtime duration in Sandbox UAT for an equivalent update and dataset that you plan to use in your production environment. During this time, the following will occur:

- Cloud-hosted Commerce channels will not function (unless POS offline capability is enabled).
- POS devices that have the offline capability feature enabled will have reduced functionality.
- Any e-commerce clients that are dependent on CCSU will be disrupted.
- Channels hosted on CSUs will remain unaffected.
- Head office functionality will remain unaffected.

NOTE

Applying an extension and an update at the same time requires a single downtime, and can be an effective way of averting multiple downtimes.

View history

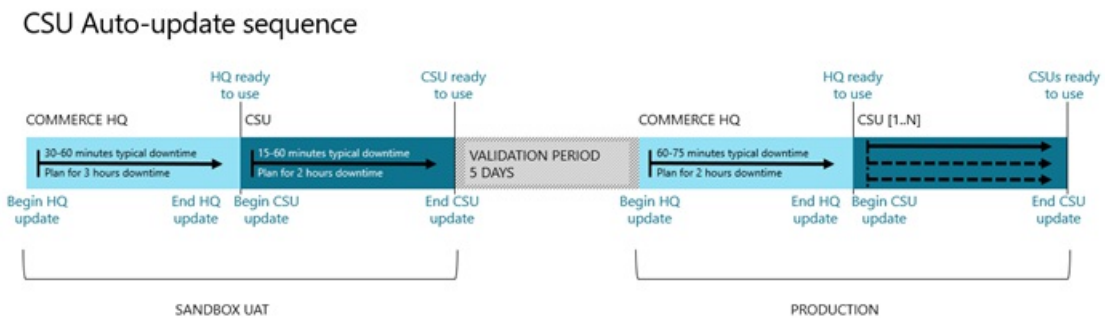
To view the history of recent operations on a Scale Unit, select **History** on the **Action** tab to open the **Scale Unit History** page. On this page, you can view recent operations such as initialize, service update, quality update, version, extension details, and other relevant information.

CSU auto-update sequence

NOTE

Auto-update for CSU is being gradually rolled out to all Commerce customers. If you are a LCS project owner or environment administrator, you'll recieve an email notification when CSU auto-update is rolled out to your LCS project.

When CSU is auto-updated by Microsoft, it takes place in the following sequence.



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Initialize Commerce Scale Unit (cloud)

2/18/2021 • 8 minutes to read • Edit Online

If you're using a Tier-2 sandbox or production environment that has application version 8.1.2.x or later, you must initialize Commerce Scale Unit (cloud) before you can use retail channel functionality either for point of sale (POS) operations or for e-Commerce operations that use Retail Server in the cloud. Initialization will deploy a Commerce Scale Unit (cloud).

This topic describes the steps for initializing Commerce Scale Unit (cloud).

> **IMPORTANT**
>
> For existing customers using retail channel functionality in the cloud, to ensure continued and uninterrupted support for your business, we require that you update your retail channels to use Commerce Scale Unit. New environments deployed without Commerce Scale Unit will no longer receive quality and service updates for cloud-hosted retail channel components. There is no action required for customers who exclusively use Commerce Scale Unit (self-hosted). Contact your Microsoft FastTrack solution architect if you require an extension.

Prerequisites

1. Deploy a Tier-2 sandbox or production environment that has application version 8.1.2.x or later.
2. If you require more than 1 RCSU per environment, in Microsoft Dynamics Lifecycle Services (LCS), create a support request, and enter **Access request for multiple Commerce Cloud Scale Units** and indicate the environment ID, number of CSUs, and corresponding datacenter regions. The request will be completed within five business days. If you do not require multiple CSUs per environment, you do not need to create a support request.
3. Ensure that Retail license configuration keys are enabled in your environment. For more information, see License codes and configuration keys report. You must have the following keys turned on to use Commerce Scale Unit.

- RetailBasic
- RetaileCommerce - If you plan to use E-Commerce for Dynamics 365 Commerce.
- RetailGiftCard - If you plan to use gift cards.
- RetailInvent - If you plan to use inventory.
- RetailModernPos - If you plan to use point of sale (POS).
- RetailReplenishment - If you plan to use replenishments.
- RetailScheduler
- RetailStores - If you plan to use POS.

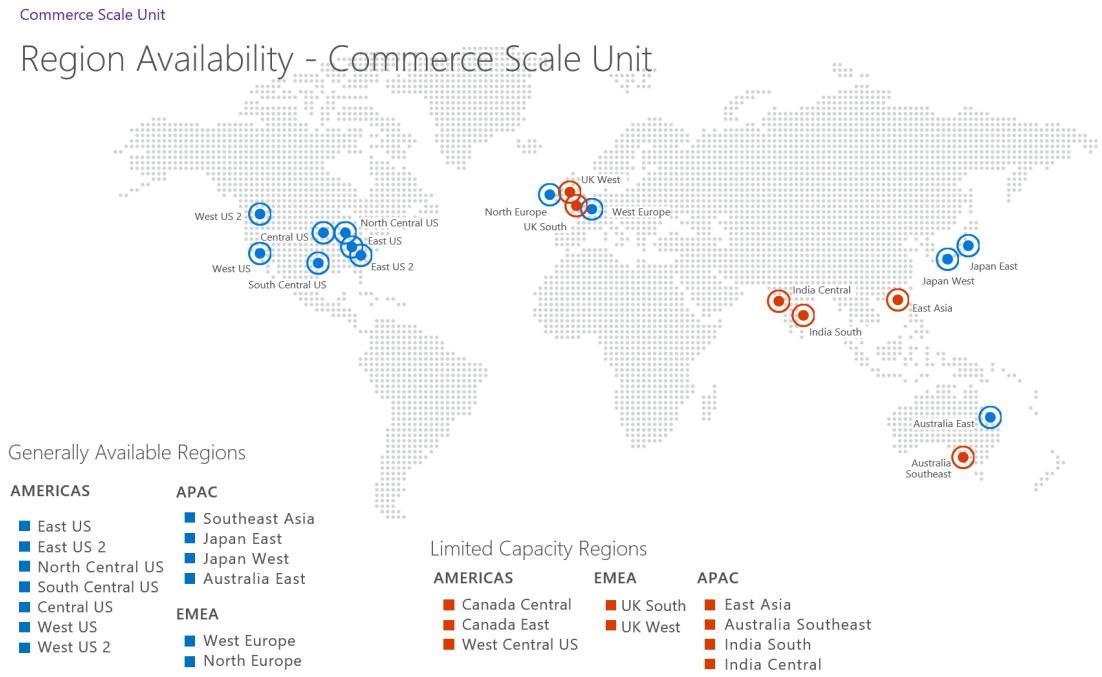
Region availability

Commerce Scale Unit is available for deployment in the following regions.

GLOBAL LOCATION	REGION	AVAILABILITY
AMERICAS	East US	Generally available
AMERICAS	East US 2	Generally available

GLOBAL LOCATION	REGION	AVAILABILITY
AMERICAS	North Central US	Generally available
AMERICAS	South Central US	Generally available
AMERICAS	Central US	Generally available
AMERICAS	West US	Generally available
AMERICAS	West US 2	Generally available
AMERICAS	Canada Central	Limited capacity
AMERICAS	Canada East	Limited capacity
AMERICAS	West Central US	Limited capacity
APAC	Australia East	Generally available
APAC	Southeast Asia	Generally available
APAC	Japan East	Generally available
APAC	Japan West	Generally available
APAC	Australia Southeast	Limited capacity
APAC	East Asia	Limited capacity
APAC	India South	Limited capacity
APAC	India Central	Limited capacity
EMEA	West Europe	Generally available
EMEA	North Europe	Generally available
EMEA	UK South	Limited capacity
EMEA	UK West	Limited capacity

Deployment capacity in Limited capacity regions are extremely constrained. Requests for deployment are evaluated on a case-by-case basis. If you have a compelling business need for deployment in Limited capacity regions, you can file a support request to be added to the waitlist.



Initialize Commerce Scale Unit as part of a new environment deployment

Please make sure the headquarters is available. This is required to register the scale unit with the headquarters during the initialization process. It is not recommended to initialize a scale unit when the headquarters is under servicing, as it may become unavailable during its servicing process.

1. Make sure the headquarters environment is available and not in Maintenance mode.
2. In LCS, on the environment details page, select **Environment features > Retail**.
3. On the Retail setup deployment page, select **Initialize**.
4. Select the version of the Commerce Scale Unit to initialize.
5. Select a region to initialize Commerce Scale Unit in.

Configure retail channels to use Commerce Scale Unit

1. After Commerce Scale Unit has been deployed, in the head office client go to **Retail and commerce > Retail Headquarters > Retail Scheduler setup > Channel database** to ensure that your retail channels are configured to use the database for this Commerce Scale Unit.
2. Go to each retail channel and select the Channel Profile for the corresponding Commerce Scale Unit.

Database refresh and Commerce Scale Units

Before you begin, make sure you are familiar with Steps to complete after a database refresh for environments that use Commerce functionality.

The scale unit channel database records (in the Channel Database form) cannot be moved across environments as part of database refresh. This is because the records represent environment specific configuration.

After database refresh, you can regenerate the scale unit's channel database record by issuing a re-deployment of your scale unit in LCS. Any deployment or servicing operation in the scale unit will attempt to register the scale unit with the headquarters, if the registration is detected as missing.

You can issue a re-deployment of the scale unit, without changing any components, by selecting to deploy the same version your scale unit is at already. This can be done in LCS by the following steps:

1. In LCS, on the environment details page, select **Environment features > Retail**.

2. On the setup deployment page, select the scale unit you would like to redeploy.
3. On the scale unit's operation menu, select **Update**.
4. On the slider, on the drop-down for **Select version**, pick the option **Specify a version**.
5. On the text box under **Specify a version**, type in the version shown for your scale unit, shown besides the **Current version** label.
6. Click on **Update** button.

You do not need to select **Update extensions**, even if you have applied extensions previously, since the last extension package applied to the scale unit is automatically picked when updating a scale unit.

If you have multiple scale units, you need to perform the operation above for each scale unit. You may perform these operations in parallel, if desired.

Deploy additional Commerce Scale Units (optional)

After you have initialized the first Commerce Scale Unit (CSU), if you require additional cloud scale units, enter a support request. In the support request, state the number of RCSUs needed, environment name, and desired regions.

For each additional RCSU that you deploy, it is also recommended that you create a separate channel database group for each RCSU. To do this, follow these steps:

1. In Commerce head office, go to **Retail and commerce > Retail Headquarters > Retail Scheduler setup > Channel database group**.
2. Create a new channel database group.
3. Go to the **Retail and commerce > Retail Headquarters > Retail Scheduler setup > Channel database** form and select the channel database that corresponds to the newly created RCSU.
4. Select **Edit** and select the new channel database group.
5. Select **Save**.
6. Select **Run Full data sync** for the selected channel database.

Additional considerations if you initialize cloud-hosted Commerce channel components in an existing environment

If you're already using cloud-hosted Commerce channel components in an environment, initialization of Commerce Scale Unit will help reduce the downtime when those components are updated. Additional planning is required before initialization of Commerce Scale Unit.

When you initialize your first Commerce Scale Unit in an environment that uses cloud-hosted Commerce channel components, the initialization process will migrate your channels associated to the cloud-hosted channel components to the first scale unit. Channels associated with Store Scale units are unaffected.

The migration process is transparent to the channels. After the scale unit initialization starts, the following operations are automatically performed:

1. A new Commerce Scale Unit will be created and associated with the environment.
2. The Commerce Scale Unit will be registered as an available Channel Database in the headquarters.
3. All channels mapped to the **Default** channel database in the headquarters will be updated to map to the new Commerce Scale Unit.
4. A Commerce Data Exchange (CDX) full data sync will be performed to bring the channel data to the new scale unit.

Planning and testing for Commerce Scale Unit initialization As a general rule, when initializing Commerce Scale Unit, you must plan for a five-hour downtime window for store operations as well as any e-

commerce channel operations that use Retail Server or Cloud Point of Sale.

1. Perform a database refresh from your production environment to a sandbox UAT environment.
2. Initialize Commerce Scale Unit in the sandbox UAT environment.
3. Note the initialization time to complete for Commerce Scale Unit. This will be comparable to the time this operation takes in your production environment, during which store operations and e-commerce operations will be unavailable.

You must perform the following additional steps before initializing Commerce Scale Unit.

- **Close all POS shifts** - After migration, POS users will be unable to close any shifts that were active during the migration process.
- **Validate that all P-jobs have been successfully completed** - It is recommended that P-jobs to synchronize pending transactions have completed before CSU is initialized.
- **Sign out of all POS device** - POS operations are not supported during migration.
- **Recall and void all suspended transactions at POS** - Suspended transactions are not preserved as part of the initialization.

> **IMPORTANT**
>
> As part of Commerce Scale Unit initialization, prior suspended transactions will be lost and cannot be recalled.

Here is what occurs during the initialization period:

- Cloud-hosted Commerce channels won't work, unless you turn on POS offline capability.
- POS devices with offline capability turned on will have reduced functionality.
- Any e-Commerce clients that depend on Retail Server will be disrupted.
- Channels that are hosted on Commerce Scale Units (self-hosted) won't be affected.
- Head office functionality is not affected.

Here is what occurs after initialization is completed:

- The device activation state of all activated POS devices is preserved, which means that the devices won't have to be reactivated.
- Stand-alone hardware station instances will continue to work.
- POS channel–side reports will be reset and won't show data from before the initialization.
- Show journal operation will also be reset and won't show data from before the initialization.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Migrate channels to a different Commerce Scale Unit

2/18/2021 • 3 minutes to read • Edit Online

This topic explains how to migrate Microsoft Dynamics 365 Commerce store channels from the Commerce Scale Unit (CSU) that they are currently working with to a different CSU. You might want to migrate channels to a different CSU for better load isolation and resource governance between channels, to reduce latency to your stores, or to manage different update/extension deployment schedules for staged roll-out and pilots. Migration to a different CSU involves downtime for the channels.

This topic describes best practices that will help you minimize business disruption and downtime while you migrate channels. It applies to the migration of channels between cloud-hosted CSUs, between self-hosted CSUs, from cloud-hosted CSUs to self-hosted CSUs, and from self-hosted CSUs to cloud-hosted CSUs.

> **NOTE**
> If you migrate channels between CSUs, temporary sales data that was used for journal records and point of sale (POS) reports before the migration will no longer be available at the POS after migration. After the migration is completed, journals and channel reports will be started afresh by using new data.

In the following procedures, the terms *origin* and *destination* are used to distinguish the CSUs and corresponding channel databases that are involved in the migration.

Planning for downtime

When you follow the procedures that are described in this topic, all long-running system processes that are involved are run before the actual migration, while the stores are still operational. These processes including synchronization of master data for products, prices, and customers. Then, during the migration, the critical period when you must take planned downtime in your environment involves data synchronization of a very small payload of channel configuration data to the new CSU. In most cases, this synchronization can be completed in under 10 minutes. However, from an operational perspective, you must plan for a longer downtime window to ensure that all prerequisite steps have enough time to be completed. These steps include closing all shifts, syncing transactions to Commerce headquarters, and posting statements. The amount of time that is required will vary by organization.

Prerequisites

First, complete all the following procedures in a sandbox user acceptance testing (UAT) environment. Then repeat them in your production environment. In this way, you can measure and estimate the downtime that you should expect in the production environment.

Migration

Configure data synchronization

The following steps can be completed while the stores are still operational. They synchronize master data to the destination CSU.

1. In Commerce headquarters, on the **Channel database** page, select the record for the channel database for the destination CSU.

2. Add the desired channels to the destination channel database.
3. Select **Full data sync**, and specify that job **9999** (**All jobs**) should be used.

> **NOTE**
> If your destination CSU is self-hosted, consider creating separate channel database groups to reduce the volume of unnecessary master data synchronization.

Prepare for migration

This procedure and the next procedure must be completed during the planned downtime for your channels.

1. On all POS devices, make sure that all shifts are closed.
2. Sign out of all POS devices.
3. Confirm that all POS offline transactions are synced to Commerce headquarters. Run P-jobs for all existing channel databases that will be migrated. If you start the migration before P-jobs are completed, and if you use Cloud POS, you might lose transactions because of duplicate transaction numbers.
4. Confirm that all statements are posted.

Migrate channels to a new CSU

1. On the **Store details** page, set the **Live channel database** field to the destination CSU database.
2. Set the **Channel profile** field to the channel profile that is associated with the destination CSU.
3. On the **Distribution schedule** page, run job **1070** (**Channel configuration job**) and job **1110** (**Global configuration job**). Select **Run now** for each job. (For asynchronous processing, select **Create batch job** instead of **Run now**.) After the jobs are completed, your channels have been migrated to the new CSU.
4. If you're using Cloud POS, you must use the URL of Cloud POS for the new CSU and reactivate the POS device. If you reactivate the POS device before all P-jobs on the origin channel database are completed, you might lose transactions because of duplicate transaction numbers.

   If you're using Modern POS, close each POS device, reopen it, and sign in.

Post-migration

You can continue to use the origin CSU to serve other channels.

> **NOTE**
> Do not delete the origin CSU. Doing so may make the store unoperable.

After you've completed all the procedures in a sandbox UAT environment, repeat them in your production environment.

Commerce data residency

2/18/2021 • 2 minutes to read • Edit Online

Choose a region

When you initialize a Commerce Scale Unit (cloud), you need to select a data center location to host. To minimize network latency and improve performance, you should choose a datacenter location that is in proximity to the channels that you plan to serve using the RCSU. To reference approximate locations of each datacenter, see Azure regions. You can also reference a web-based utility, such as Azure speed reference, and measure latency to Azure datacenters from each store location. This can help you to make the right choice of datacenter when you initialize the RCSU.

Data between regions

If you initialize RCSU in a data center that is different than where your head office is located, the data will travel between these data centers with periodic synchronization. The system is pre-configured to transfer specific types of data. You can modify this configuration to synchronize different data.

To view the data synchronization configuration, go to **Retail and Commerce** > **Headquarters setup** > **Commerce scheduler** > **Scheduler jobs** to view the data synchronization jobs and sub-jobs. To view the fields being synchronized, click through a sub-job.

Synchronize specific segments of records

You can configure Commerce Data Exchange (CDX) so that only specific segments of records are synchronized to specific RCSUs.

Prerequisites

Before you configure the record segments that will be synchronized, you need to configure Channel database groups. These database groups must be configured for each CSU where you want to synchronize segmented data. All CSUs in the same Channel database group receive the data that is needed to serve all the channels in that CSU. You will need to create a separate database group for each CSU channel database where you plan to synchronize segmented data. To do this, perform the following steps:

1. Go to **Retail and Commerce** > **Headquarters setup** > **Commerce scheduler** > **Channel database group**.
2. Create a new database group for each CSU where you want to synchronize segmented data.
3. Go to **Retail and Commerce** > **Retail and Commerce IT** > **Distribution schedule**.
4. Add the newly created Channel database group to each scheduler job.
5. Go to **Retail and Commerce** > **Headquarters setup** > **Commerce scheduler** > **Channel database**.
6. Select the Channel database for the corresponding Commerce Scale Unit (cloud).
7. Select **Edit** and choose the channel database group that you created in step 2.
8. Select **Save**.
9. Select **Full data sync for all jobs** for the selected channel database.

Available controls for synchronizing segments of data to different RCSUs

Channel configuration data will only be synchronized for channels that are served by specific RCSUs. For example, only the channel configuration data for channels that are configured to be served by the West Europe channel will be synchronized to that RCSU.

Using assortments, product and pricing related data can be segmented by specific channels. For example, if your stores in North America only sell women's shoes and your stores in West Europe only sell sporting goods, you can use Assortments to segment this data. When data is synchronized to RCSU, women's shoes data will only be synchronized to the North America RCSU and sporting goods data will only be synchronized to the West Europe RCSU.

Customer and employee records are configured using the Global Address Book, which can be configured for specific channels. For example, you can configure separate address books for customers and employees for channels served by the West Europe RCSU and those for North America RCSU.

On-premises deployment home page

2/18/2021 • 2 minutes to read • Edit Online

You can deploy Dynamics 365 Finance + Operations (on-premises). When you choose an on-premises deployment type, the system requirements, hardware sizing, and functionality differ from a cloud deployment. This topic provides links to content that contains information specific to on-premises deployments.

Get started

- On-premises deployment overview
- Plan and prepare for on-premises deployments
- System requirements for on-premises deployments
- Hardware sizing requirements for on-premises environments
- Buy Finance + Operations (on-premises)
- Comparison of cloud and on-premises features

Onboard

- Set up on-premises projects in Lifecycle Services (LCS)
- Set up and deploy on-premises environments (Platform update 12 and later)
- Install network printer devices in on-premises environments
- Configure SQL Server Reporting Services for on-premises deployments
- Develop and deploy custom models to on-premises environments

Work in your on-premises deployment

- Configure document management
- Import Electronic reporting (ER) configurations
- Document generation, publishing, and printing in on-premises deployments
- Configure proxies for on-premises environments
- Set up technical support for Finance and Operations apps
- Client internet connectivity
- Apply updates to on-premises deployments
- Redeploy on-premises environments
- Reuse the same AD FS instance for multiple environments

Commerce

- Commerce capabilities that are available in on-premises deployments
- Installation steps for Retail channel components in an on-premises environment
- Configure, install, and activate Modern POS (MPOS)
- Configure and install Commerce Scale Unit

Upgrade

- In-place upgrade process for on-premises environments

Other resources

- Troubleshoot on-premises deployments
- Scripts for resolving issues in on-premises environments
- Certificate rotation
- On-premises diagnostics
- Removed or deprecated features for Finance and Operations
- Software lifecycle policy and on-premises releases

On-premises deployment overview

2/18/2021 • 2 minutes to read • Edit Online

Microsoft Dynamics 365 Finance + Operations (on-premises) supports running business processes in customer data centers. With this deployment option, application servers and the Microsoft SQL Server database will run in the customer's data center. Customers and partners will utilize Microsoft Dynamics Lifecycle Services (LCS) to manage their on-premises deployments. LCS is an application management portal that provides tools and services for managing the application lifecycle of your implementations in the cloud and on-premises. LCS features, such as business process modeling, software deployment and patching, and monitoring and diagnostics, are used to help support on-premises deployments.

> **IMPORTANT**
>
> Dynamics 365 Finance + Operations (on-premises) is not supported on any public cloud infrastructure, including Azure.

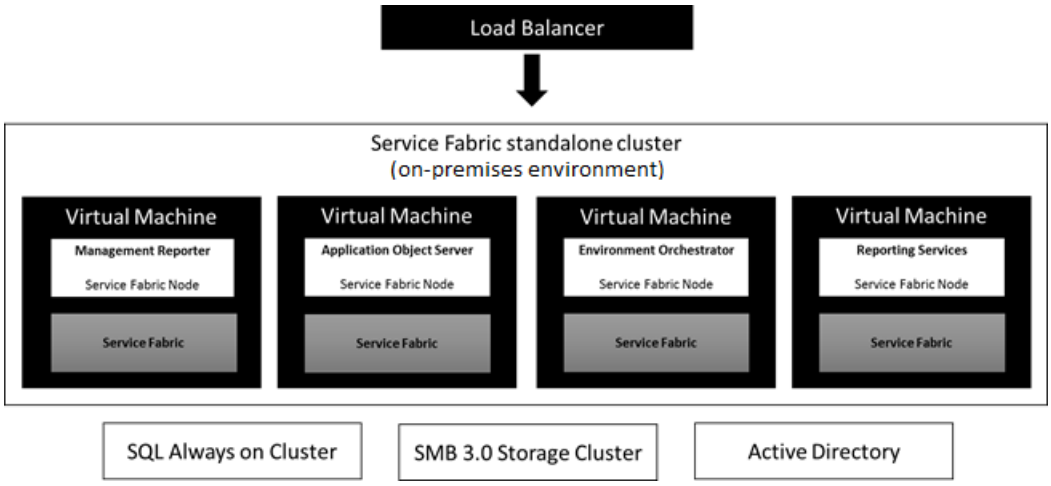
Architecture

The on-premises deployment option uses cloud components running on-premises using Microsoft Azure Server Service Fabric standalone clusters. Service Fabric is the next-generation Microsoft middleware platform for building and managing enterprise-class high-scale applications. Service Fabric standalone clusters can be deployed on any computer that is running Windows Server.

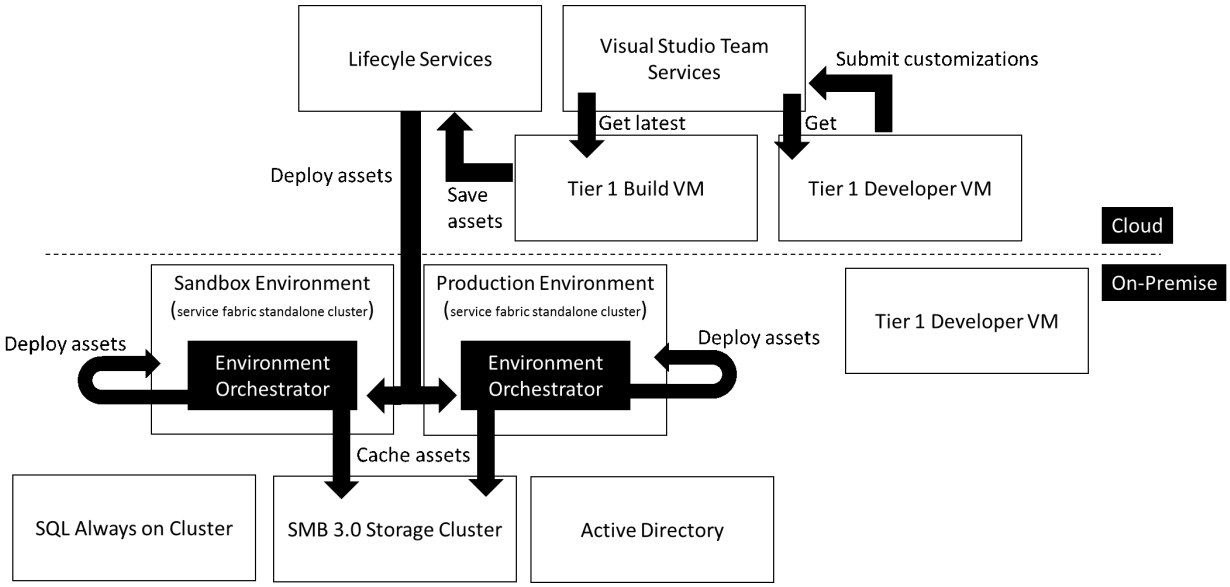
On-premises deployment defines two types of Service Fabric standalone clusters: clusters for production environments and clusters for sandbox environments. The following roles or node types are deployed into both types of clusters:

- Application Object Servers (AOS) – Provides the ability to run the application functionality in client, batch, and import/export scenarios.
- Management Reporter (MR) – Provides financial reporting functionality.
- SQL Server Reporting Services (SSRS) – Provides document reporting functionality.
- Environment Orchestrator – Enables on-premises environment management from LCS.

Figure 1 shows a logical diagram of the node types deployed in a Service Fabric standalone cluster.



Application lifecycle management for on-premises deployments is orchestrated through LCS. Customers can use the proven tools and methodologies in LCS to help manage their on-premises deployments (Figure 2). The development experience continues to be the same as in cloud deployments through 1-box VHDs.



Data storage

The on-premises deployment option stores core customer data on-premises. Core customer data is a subset of the customer data definition provided in the Microsoft Trust Center. Table 1 outlines the categories of customer data that are stored in Microsoft Azure data centers located in the United States by services such as LCS, Azure Active Directory, and Microsoft Office signup portal. All other customer data, referred to as core customer data, is stored on-premises.

Table 1: Customer data stored in Microsoft Azure data centers located in the United States by services supporting on-premises environments. These services enable initial onboarding, initiation, and tracking of support incidents, and service updates and upgrades.

SUPPORTING SERVICES	CUSTOMER DATA DEFINITION
Microsoft Dynamics Lifecycle Services	Project content and files are stored in a project. This includes application configuration data, code, metadata, and data assets that comprise the application and business process models. Also included is anonymized user activity logs and information that is collected during the onboarding process.
Microsoft Office signup portal	Customer information that is collected during the onboarding process.
Microsoft Azure Active Directory	Authentication for LCS and Azure DevOps.

Additional services or components can be configured to extend an on-premises deployment as needed; however, configuration choices may cause core customer data to be transferred outside of the customer's data center. For example, configuring data management features that are used to integrate external services with an on-premises deployment may result in the transfer of core customer data outside the on-premises deployment.

Plan and prepare for on-premises deployments

2/18/2021 • 12 minutes to read • Edit Online

Dynamics 365 Finance + Operations (on-premises) supports running business processes in customer data centers. With this deployment option, application servers and the Microsoft SQL Server database will run in the customer's data center.

This topic will help you plan and prepare for your on-premises deployment.

> **IMPORTANT**
> Dynamics 365 Finance + Operations (on-premises) is not supported on any public cloud infrastructure, including Azure.

Differences between cloud deployments and on-premises deployments

The features in cloud deployments and on-premises deployments differ. These differences will affect your planning. The differences are described in the following topics:

- Deployment options
- Comparison of cloud and on-premises features
- Removed or deprecated features for Finance and Operations

How LCS is used with on-premises deployments

Microsoft Dynamics Lifecycle Services (LCS) is an application management portal that provides tools and services for managing the application lifecycle. Customers and partners use LCS to manage both cloud and on-premises deployments. You can use LCS for the following tasks:

- Deploy cloud and on-premises environments.
- Service your environments.
- Monitor, diagnose, and analyze the health of the environments that you manage (cloud only).
- Search for product issues and regulatory features.
- Obtain support.

For more information about LCS, see Lifecycle Services resources.

Environments

There are four types of environments that you need to plan for. This section describes the four environments and how to access and deploy them.

Demo environment

You can sign up for a demo environment to learn about the system. The demo environment is applicable to both cloud and on-premises deployments. For more information, see Sign up for preview subscriptions.

Developer environment

The development experience is the same for cloud and on-premises deployments. To access a developer environment, see Deploy and access development environments.

Sandbox environment

Business users and functional team members validate application functionality by using a sandbox environment. This functionality includes customizations and data that was brought forward from Microsoft Dynamics AX 2012 environments. To deploy an on-premises sandbox environment, see Set up and deploy on-premises environments home page.

At a minimum, an on-premises sandbox environment requires:

- 3 machines running Environment Orchestrator
- 2 machines running Application Object Servers (AOS)
- 1 machine running Management Reporter (MR)
- 1 machine running SQL Server Reporting Services (SSRS)
- 1 machine running Active Directory
- 1 machine running SQL Server

Production environment

The production environment is the live deployment that your users and customers have access to. To deploy a production environment, see Set up and deploy on-premises environments home page.

At a minimum, an on-premises production environment requires:

- 3 machines running Environment Orchestrator
- 3 machines running Application Object Servers (AOS)
- 1 machine running Management Reporter (MR)
- 1 machine running SQL Server Reporting Services (SSRS)
- 2 or more machines running SQL Server
- 2 or more machines running Active Directory

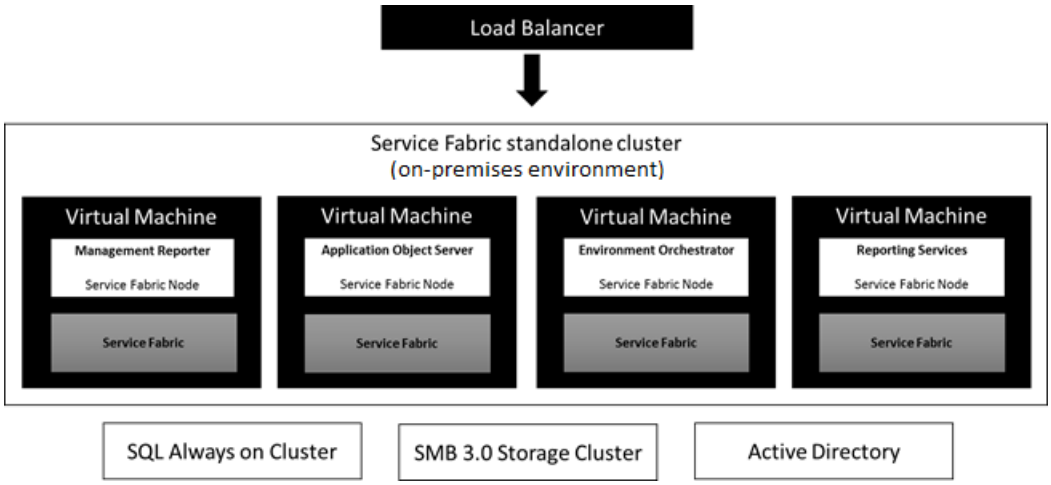
Service Fabric

An on-premises deployment uses Azure Service Fabric standalone clusters. Service Fabric is the next-generation Microsoft middleware platform for building and managing enterprise-class, high-scale applications. Service Fabric standalone clusters can be deployed on any computer that is running Windows Server.

An on-premises deployment has a standalone cluster for each sandbox environment and a standalone cluster for each production environment. The following roles or node types are deployed into both types of clusters:

- Application Object Servers (AOS) – Provides the ability to run the application functionality in client, batch, and import/export scenarios.
- Management Reporter (MR) – Provides financial reporting functionality.
- SQL Server Reporting Services (SSRS) – Provides document reporting functionality.
- Environment Orchestrator – Enables on-premises environment management from LCS.

The following diagram shows the node types deployed in a Service Fabric standalone cluster.



Service Fabric resources

To learn more about Service Fabric, see the following topics:

- Azure Service Fabric documentation - To learn more about Service Fabric.
- Service Fabric application upgrade - An Azure Service Fabric application is a collection of services that requires periodic upgrades.
- Plan and prepare your Service Fabric standalone cluster deployment - Additional information about Service Fabric clusters and antivirus exclusions.

System requirements

Review the system requirements in System requirements for on-premises deployments and be aware of the number of machines that are required for on-premises deployments.

Hardware sizing

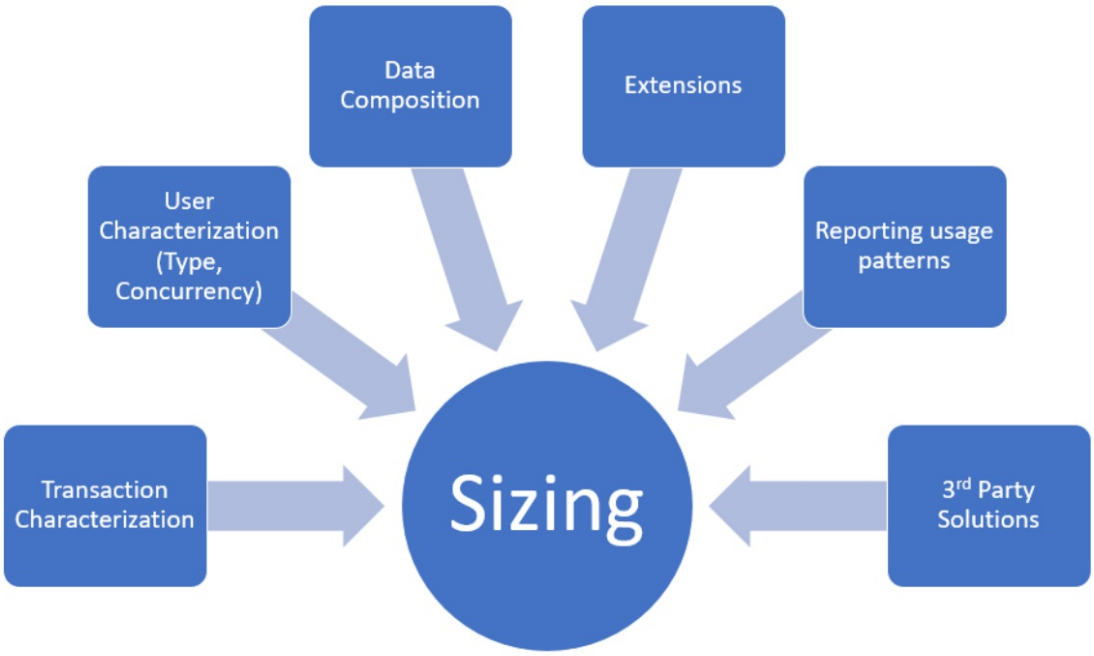
Before you begin the hardware and infrastructure sizing process for an on-premises environment, familiarize yourself with the System requirements for on-premises deployments and Set up and deploy on-premises environments home page to gain a solid understanding of the underlying infrastructure. Pay close attention to the system setup best practices for optimum performance. After you have reviewed the documentation, you can start the process of estimating your transactional and concurrent user volume and sizing your environment based on the average core throughput.

Factors that affect sizing

The core factors that affect sizing are:

- Transaction characterization
- User characterization - Type and concurrency
- Data composition
- Extensions
- Reporting usage patterns
- Third-party solutions

The more detailed data that you collect, the more precisely you can estimate sizing. Hardware sizing, without supporting data, is likely to be inaccurate. The minimum data that you need to collect is the peak transaction line load per hour. The factors that affect sizing are shown in the following diagram.



From left to right, the first and most important factor needed to accurately estimate sizing is a transaction profile or a transaction characterization. It's important to find the peak transactional volume per hour. If there are multiple peak periods, then these periods need to be accurately defined.

As you understand the load that impacts your infrastructure, you also need to understand more detail about these factors:

- **Transactions** – Transactions typically have certain peaks throughout the day or week. The peaks might depend on the transaction type. For example, time and expense entries usually show peaks once per week, while sales orders might arrive in bulk via integration or trickle in during the day.
- **Number of concurrent users** – The number of concurrent users is the second most important sizing factor. You cannot get reliable sizing estimates based only on the number of concurrent users. If concurrent users is the only data that you have available, then estimate an approximate number for transactions, and revisit this when you have more data. An accurate concurrent user definition means that:
  - Named users are not concurrent users.
  - Concurrent users are always a subset of named users.
  - Peak workload defines the maximum concurrency for sizing. For concurrent users, the user must meet all the following criteria:
    - The user is logged on.
    - There are working transactions or inquiries at the time of counting.
    - The session is not idle.
- **Data composition** – Data composition is how your system will be set up and configured. For example, this can include the number of legal entities, the number items, the number of BOM levels, and how complex the security setup will be. Each of these factors might have an impact on performance, however the impact can be offset by using smart choices when it comes to infrastructure.
- **Extensions** – Customizations can be simple or complex. The number of customizations and the nature of complexity and usage has a varied impact on the size of the infrastructure needed. For complex customizations, you should conduct performance evaluations to ensure that they are not only tested for efficiency but also help understand the infrastructure needs. This is even more critical when the extensions are not coded according to best practices for performance and scalability.
- **Reporting and analytics** – Reporting and analytics typically include running heavy queries against the database systems. Reducing the frequency of when data intensive reports run will help reduce their impact. It's also important to understand how the design of your queries impacts their performance.

- **Third-party solutions** – These solutions, like ISVs, have the same implications and recommendations as extensions.

Sizing your environment

To determine your sizing requirements, you must know the peak volume of transactions that you need to process. Most auxiliary systems, like Management Reporter or SSRS, are less mission critical. As a result, this topic focuses primarily on AOS and SQL Server.

In general, the compute tiers scale out and should be set up in an N+1 fashion, meaning if you estimate three AOS, add a fourth AOS. The database tier should be set up in an Always On highly-available setup.

SQL Server (OLTP)

Sizing

- 3K to 15K transaction lines per hour per core on DB server.
- Typical AOS-to-SQL core ratio 3:1 for the primary SQL Server. Additional cores are required based on the high-availability configuration.
  - Processing database-heavy operations may regress this to 2:1.
- The following factors influence variations:
  - Parameter settings in use.
  - Levels of extensions.
  - Additional functionality usage, such as database logs and alerts. Extreme database logging will further reduce throughput per hour per core below 3K lines.
  - Complexity of data composition. For example, a simple chart of accounts versus a detailed fine-grained chart of accounts has implications on throughput.
  - Transaction characterization.
  - 2 GB to 4 GB memory for each core.
  - Auxiliary databases on DB server such as Management reporter and SSRS databases.
  - Temp DB = 15% of DB size, with as many files as physical processors.
  - SAN size and throughput based on total concurrent transaction volume/usage.

High availability

You should always utilize SQL Server in either a cluster or mirroring setup. The second SQL node should have the same number of cores as the primary node.

Active Directory Federation Services (AD FS)

For AD FS sizing, see the AD FS Server Capacity documentation. A sizing spreadsheet is available for planning the number of instances in your deployment.

AOS (Online and batch)

Sizing

- Sizing by transaction volume/usage
  - 2K to 6K lines per core
  - 16 GB per instance
  - Standard box – 4 to 24 cores
  - 10 to 15 Enterprise users per core
  - 15 to 25 Activity users per core
  - 25 to 50 Team members per core
- Batch
  - 1 to 4 batch threads per core
  - Size based on batch window characterization
- Note that AOS, Data Management, and Batch are the same role in the Service Fabric. You need to size for

these three workloads combined, and not separately as you did with Microsoft Dynamics AX 2012.

- The same variability factors for SQL Server apply here.

High availability

- Ensure that you have at least 1 to 2 more AOS available than you estimate.
- Ensure that you have at least 3 to 4 virtual hosts available.

Management reporter

In most cases, unless used extensively, the recommended minimum requirements using two nodes should work well. Only in cases where there is heavy use will you need more than two nodes, after which you can scale as needed.

SQL Server Reporting Services

For the current release of Finance + Operations, only one SSRS node can be deployed. Monitor your SSRS node while testing and increase the number of cores available for SSRS as needed. Make sure that you have a preconfigured secondary node available on a virtual host that is different than the SSRS VM. This is important if there is an issue with the virtual machine that hosts SSRS or the virtual host. If this the case, the node would need to be replaced.

Environment Orchestrator

The Orchestrator service is the service that manages your deployment and the related communication with LCS. This service is deployed as the primary Service Fabric service and requires at least three VMs. This service is co-located with the Service Fabric orchestration services. This should be sized to the peak load of the cluster. For more information, see Plan and prepare your Service Fabric Standalone cluster deployment.

Virtualization and oversubscription

Mission critical services like the AOS should be hosted on Virtual hosts that have dedicated resources – core, memory, and disk.

Authentication methods

The following authentication methods are used with on-premises deployments:

- **Azure Active Directory (Azure AD)** - Azure AD is the authentication method used to log in to LCS. Azure AD is used configure the LCS Local Agent. For more information, see What is Azure Active Directory?
- **Active Directory Domain Services (AD DS)** - The machines that host Finance + Operations components must belong to an Active Directory domain. You must configure Active Directory Domain Services (AD DS) in native mode. For more information, see Active Directory Domain Services.
- **Active Directory Federation Services (AD FS)** - AD FS is the authentication method used in an on-premises deployment. AD FS provides access control and single sign on across a wide variety of applications including Microsoft 365, cloud-based SaaS applications, and applications on the corporate network.
  - For the IT organization, it enables you to provide sign on and access control to both modern and legacy applications, on-premises and in the cloud, based on the same set of credentials and policies.
  - For the user, it provides seamless sign on using the same, familiar account credentials.
  - For the developer, it provides an easy way to authenticate users whose identities live in the organizational directory. This means you can focus your efforts on your application, not authentication or identity.

    For more information, see Active Directory Federation Services.

Data stored in Azure data centers

The on-premises deployment option for Finance + Operations stores core customer data on-premises. Core customer data is a subset of the customer data definition provided in the Microsoft Trust Center.

The following table outlines the services that are used to store customer data in Azure data centers located in the United States. Services include Lifecycle Services (LCS), Microsoft Office signup portal, and Azure Active Directory. These services enable initial onboarding, initiation, tracking of support incidents, and service updates and upgrades. All other customer data, referred to as core customer data, is stored on-premises.

SUPPORTING SERVICES	CUSTOMER DATA DEFINITION
Microsoft Dynamics Lifecycle Services	Project content and files are stored in a project. This includes application configuration data, code, metadata, and data assets that include the application and business process models.
Microsoft Office signup portal	Customer information that is collected during the onboarding process.
Microsoft Azure Active Directory	Authentication for LCS and Azure DevOps.

Additional services or components can be configured to extend an on-premises deployment as needed; however, configuration choices may cause core customer data to be transferred outside of the customer's data center. For example, configuring data management features that are used to integrate external services with an on-premises deployment may result in the transfer of core customer data outside the on-premises deployment.

Next steps

After you've completed the planning activities mentioned in this topic, you can begin the procedures listed in the Onboard section of the On-premises deployment home page.

Be sure to refer to the On-premises deployment home page throughout your implementation for more information about planning, deployment, maintenance, and troubleshooting.

Hardware sizing requirements for on-premises environments

2/18/2021 • 6 minutes to read • Edit Online

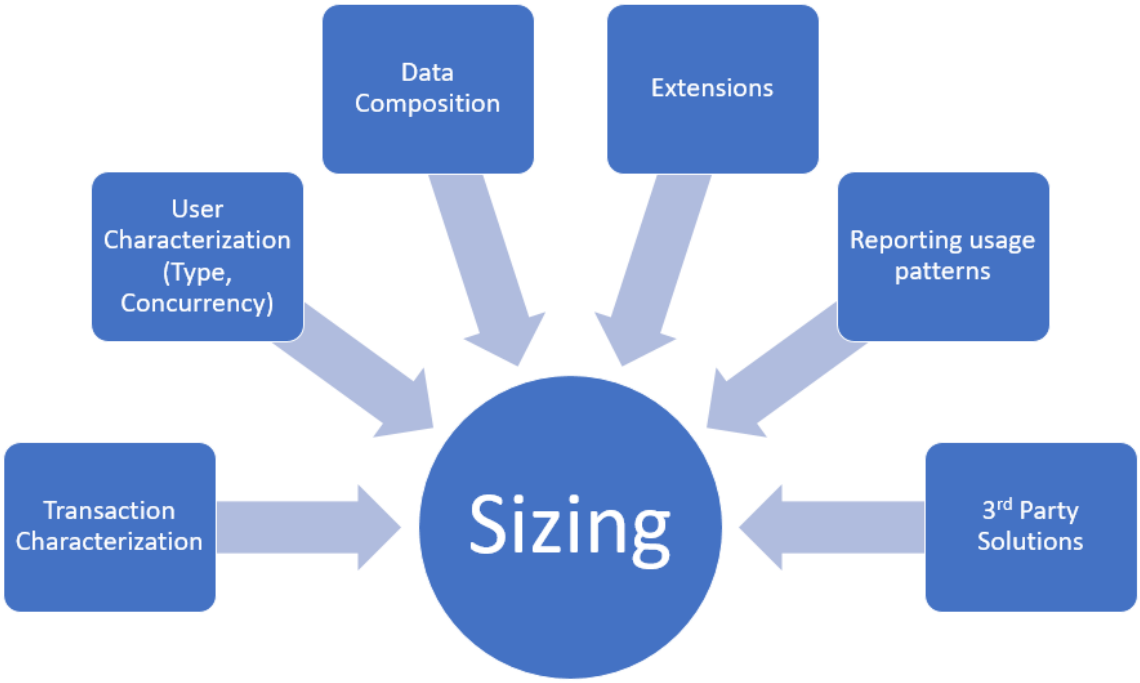
Before you begin the hardware and infrastructure sizing process for an on-premises environment, familiarize yourself with the System requirements for cloud deployments and Setup and deployment instructions to gain a solid understanding off the underlying infrastructure.

> **NOTE**
> Pay close attention to the system setup best practices for optimum performance.

After you have reviewed the documentation, you can start the process of estimating your transactional and concurrent user volume and sizing your environment based on the average core throughput.

Factors that affect sizing

All the factors shown in the following illustration contribute to sizing. The more detailed information that is collected, the more precisely you can determine sizing. Hardware sizing, without supporting data, is likely to be inaccurate. The absolute minimum requirement for necessary data is the peak transaction line load per hour.



Viewed from left to right, the first and most important factor needed to accurately estimate sizing is a transaction profile or a transaction characterization. It's important to always find the peak transactional volume per hour. If there are multiple peak periods, then these periods need to be accurately defined.

As you understand the load that impacts your infrastructure, you also need to understand more detail about these factors:

- **Transactions** – Typically transactions have certain peaks throughout the day/week. This mostly depends

on the transaction type. Time and expense entries usually show peaks once per week, whereas Sales order entries often come in bulk via integration or trickle in during the day.

- **Number of concurrent users** – The number of concurrent users is the second most important sizing factor. You cannot reliably get sizing estimates based on the number of concurrent users, so if this is the only data you have available, estimate an approximate number, and then revisit this when you have more data. An accurate concurrent user definition means that:
  - Named users are not concurrent users.
  - Concurrent users are always a subset of named users.
  - Peak workload defines the maximum concurrency for sizing.

  Criteria for concurrent users is that the user meets all the following criteria:
  - Logged on.
  - Working transactions/inquiries at the time of counting.
  - Not an idle session.
- **Data composition** – This is mostly about how your system will be set up and configured. For example, how many legal entities you will have, how many items, how many BOM levels, and how complex your security setup will be. Each of those factors may have a small impact on performance, so these factors can be offset by using smart choices when it comes to infrastructure.
- **Extensions** – Customizations can be simple or complex. The number of customizations and the nature of complexity and usage has a varied impact on the size of the infrastructure needed. For complex customizations, it's advised to conduct performance evaluations to ensure that they are not only tested for efficiency but also help understand the infrastructure needs. This is even more critical when the extensions are not coded according to best practices for performance and scalability.
- **Reporting and analytics** – These factors typically include running heavy queries against the various databases in the system. Understanding and reducing the frequency when expensive reports run will help you understand the impact of them.
- **Third-party solutions** – These solutions, like ISVs, have the same implications and recommendations as extensions.

Sizing your environment

To understand your sizing requirements, you need to know the peak volume of transactions that you need to process. Most auxiliary systems, like Management Reporter or SSRS, are less mission critical. As a result, this document focuses mostly on AOS and SQL Server.

> **NOTE**
>
> In general, the compute tiers scale out and should be set up in an N+1 fashion, meaning if you estimate three AOS, add a fourth AOS. The database tier should be set up in an Always On highly-available setup.

SQL Server (OLTP)

Sizing

- 3K to 15K transaction lines per hour per core on DB server.
- Typical AOS-to-SQL core ratio 3:1 for the primary SQL Server. Additional cores are required based on the chosen high availability configuration.
  - Processing database-heavy operations may regress this to 2:1.

- The following factors influence variations:
    - Parameter settings in use.
    - Levels of extensions.
    - Usage of additional functionality, such as database log and alerts. Extreme database logging will further reduce throughput per hour per core below 3K lines.
    - Complexity of data composition – A simple chart of accounts versus a detailed fine-grained chart of accounts has implications on throughput (as an example).
    - Transaction characterization.
    - 2 GB to 16 GB memory for each core.
    - Auxiliary databases on DB server such as Management reporter and SSRS databases.
    - Temp DB = 15% of DB size, with as many files as physical processors.
    - SAN size and throughput based on total concurrent transaction volume/usage.

High availability

We recommend always utilizing SQL Server in either a cluster or mirroring setup. The second SQL node should have the same number of cores as the primary node.

Active Directory Federation Services (AD FS)

For AD FS sizing, see the AD FS Server Capacity documentation.

A sizing spreadsheet is available for planning the number of instances in your deployment.

AOS (Online and batch)

Sizing

- Sizing by transaction volume/usage
    - 2K to 6K lines per core
    - 16 GB per instance
    - Standard box – 4 to 24 cores
    - 10 to 15 Enterprise users per core
    - 15 to 25 Activity users per core
    - 25 to 50 Team members per core
- Batch
    - 1 to 4 batch threads per core
    - Size based on batch window characterization
- Note that the AOS, Data Management, and Batch are on the same role in the Service Fabric. You need to size for these three workloads combined, and not separate these like in Microsoft Dynamics AX 2012.
- The same variability factors for SQL Server apply here.

High availability

- Ensure that you have at least 1 to 2 more AOS available than you estimate.
- Ensure that you have at least 3 to 4 virtual hosts available.

Management reporter

In most cases, unless used extensively, the recommended minimum requirements using two nodes should work well. Only in cases where there is heavy use will you need more than two nodes. Please scale as needed.

SQL Server Reporting Services

For the general availability release, only one SSRS node can be deployed. Monitor your SSRS node while testing and increase the number of cores available for SSRS on a need basis. Make sure that you have a preconfigured secondary node available on a virtual host that is different than the SSRS VM. This is important if there is an issue with the virtual machine that hosts SSRS or the virtual host. If this the case, they would need to be replaced.

Environment Orchestrator

The Orchestrator service is the service that manages your deployment and the related communication with LCS. This service is deployed as the primary Service Fabric service and requires at least three VMs. This service is co-located with the Service Fabric orchestration services. This and should be sized to the peak load of the cluster. For more information, see Plan and prepare your Service Fabric Standalone cluster deployment.

Virtualization and oversubscription

Mission critical services like the AOS should be hosted on Virtual hosts that have dedicated resources – cores, memory, and disk.

Authentication in Dynamics 365 Finance + Operations (on-premises)

2/18/2021 • 4 minutes to read • Edit Online

This topic explains authentication in Dynamics 365 Finance + Operations (on-premises). This topic also provides background information about how the process works so that if you encounter issues with authentication you can work to resolve them.

The URL for Active Directory Federation Services (AD FS)

The first part of the authentication process is to provide the URL for Active Directory Federation Services (AD FS). This URL will be similar to: `https://adfs.contoso.com/adfs/.well-known/openid-configuration`

You'll find this URL in the deployment instructions found in Configure AD FS. During deployment, the URL is used to set various options in the AOS startup variables of each AOS instance. These startup variables reside in an .XML config file located in a Service Fabric directory. This directory will vary from machine to machine, but the path should look similar to:

C:\ProgramData\SF\AOS_10\Fabric\work\Applications\AXSFType_App218\AXSF.Package.1.0.xml

XML configuration file

There is a file called AXSF.Package.Current.xml. This file will be a copy of the AXSF.Package.1.0.xml in Finance and Operations deployments. The AXSF.Package.Current.xml file represents the variable that have been used to initialize the currently running AOS instance (AxService.exe).

Within this configuration file (which is on each AOS machine), you'll find some sections that are set from the Lifecycle Services (LCS) deployment setting for AD FS.

```
<Section Name="Aad">
    <Parameter Name="AADIssuerNameFormat" Value="http://ADFS.contoso.com/{0}/services/trust" />
    <Parameter Name="AADLoginWsfedEndpointFormat" Value="https://ADFS.contoso.com/{0}/wsfed" />
    <Parameter Name="AADMetadataLocationFormat" Value="https://ADFS.contoso.com/FederationMetadata/2007-
06/FederationMetadata.xml" />
    <Parameter Name="AADTenantId" Value="adfs" />
    <Parameter Name="AADValidAudience" Value="https://ax.contoso.com/" />
    <Parameter Name="ACSServiceEndpoint" Value="https://accounts.accesscontrol.windows-
ppe.net/tokens/OAuth/2" />
    <Parameter Name="ACSServicePrincipal" Value="00000001-0001-0000-c000-000000000000" />
    <Parameter Name="ADFSEndpoint" Value="https://ADFS.contoso.com/adfs" />
    <Parameter Name="ADFSIdentifier" Value="http://ADFS.contoso.com/adfs/services/trust" />
    <Parameter Name="FederationMetadataLocation" Value="https://ADFS.contoso.com/FederationMetadata/2007-
06/FederationMetadata.xml" />
    <Parameter Name="Realm" Value="spn:00000015-0000-0000-c000-000000000000" />
    <Parameter Name="TenantDomainGUID" Value="adfs" />
    <Parameter Name="TrustedServiceAppIds" Value="913c6de4-2a4a-4a61-a9ce-945d2b2ce2e0" />
</Section>
```

You will also find the following sections.

```
<Section Name="OfficeApps">
    <Parameter Name="AppInsightsKey" Value="" />
    <Parameter Name="AuthClientId" Value="d8230a86-015d-4c14-bcd6-c7fb65176b16" />
</Section>
<Section Name="OpenIDConnect">
    <Parameter Name="ClientID" Value="d8230a86-015d-4c14-bcd6-c7fb65176b16" />
    <Parameter Name="Metadata" Value="https://ADFS.contoso.com/adfs/.well-known/openid-configuration" />
</Section>
<Section Name="Provisioning">
    <Parameter Name="AdminIdentityProvider" Value="http://ADFS.contoso.com/adfs/services/trust" />
    <Parameter Name="AdminPrincipalName" Value="axserviceuser@contoso.com" />
</Section>
```

NOTE

The settings shown above represent a deployment configured with Microsoft 365 compatibility. For more information, see AD FS Microsoft 365 compatibility.

Configuration values used by the AOS

The AOS uses the configuration values above to determine where to redirect an unauthenticated request when a user sends a request to the application URL.

1. Request is sent by the browser to the application URL (`https://ax.contoso.com/namespaces/AXSF/`).
2. The request is processed by the Gateway and gets forwarded to an AOS node that accepts interactive sessions.
3. The request reaches the AOS server and checks for the authentication cookies.
4. No authentication is present so the AOS server returns a redirect request for the user to authenticate with AD FS. At this point, the AOS also sets an affinity cookie to bind the user session to that AOS.
5. The Gateway receives the response and forwards it back to the browser.
6. The browser receives the redirect request and displays the AD FS authentication page so the user signs in.
7. When successfully authenticated against AD FS, the AD FS then redirects the user back to the application URL and provides the authentication cookies.
8. The Gateway receives this response and forwards the affinitized request to the appropriate AOS node.
9. The AOS checks the authentication information provided and checks against the **UserInfo** table to determine whether the user is allowed to access the application and which permissions are available.

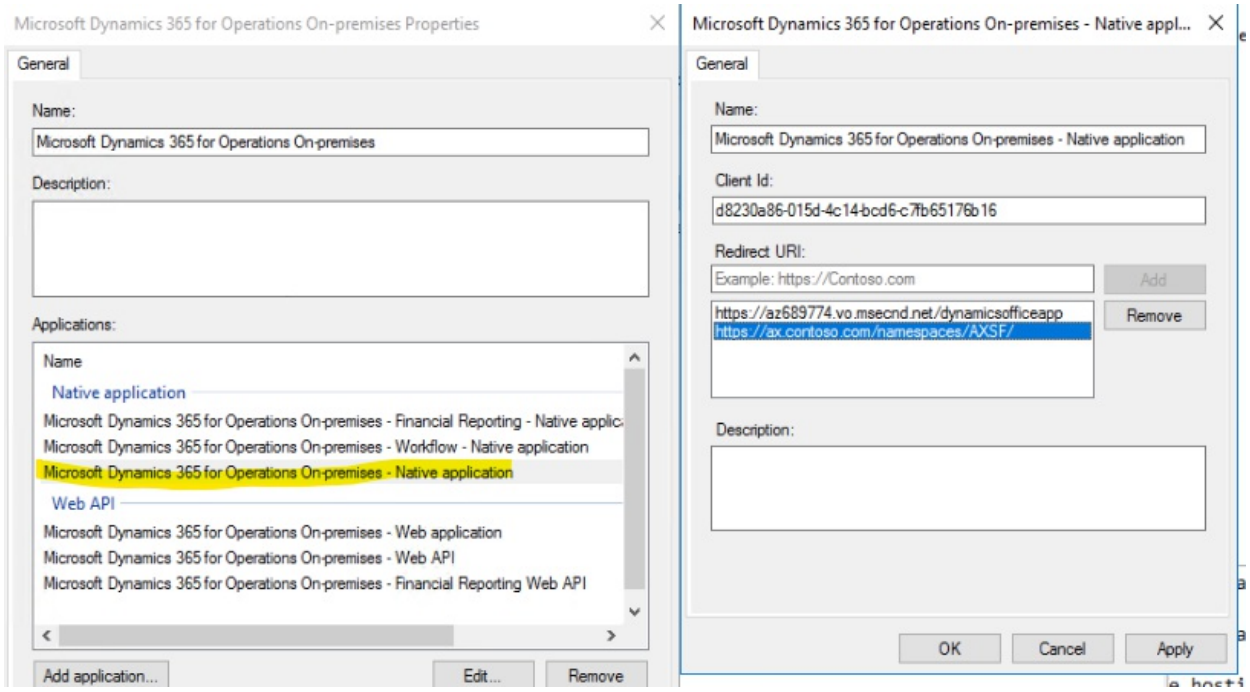
If values in the AOS config file are incorrect, then that typically means the value provided for the AD FS endpoint when deploying the environment was incorrect. The easiest thing is to delete and redeploy the environment from LCS with the correct value. It is possible to manually edit the configuration files, but to be safe, do a redeploy. Otherwise you will need to manually change the values after each servicing operation on each AOS node. If you do edit the config files, then you need to restart the AOS service (AxService.exe) for it to take effect. You can do that from the Service Fabric explorer (right-click the AOS node under **Nodes**, choose **Restart**, and then wait at least a minute for the status to change to green). You can also reboot the machine.

Receiving a 500 error when accessing the application URL is an indication that there may be an invalid URL for AD FS. This is because on startup the AOS will use that URL to obtain information from the AD FS server. If the URL is incorrect or inaccessible, the AOS will be unable to start.

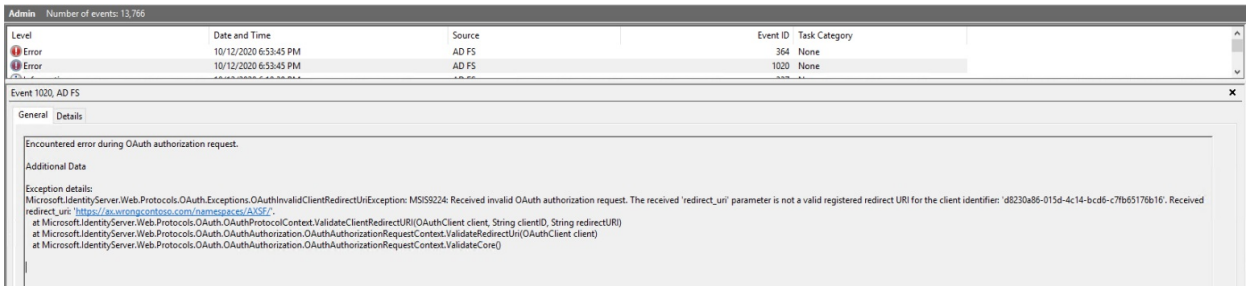
AD FS

The second part of the authentication process is AD FS itself. On the AD FS server if you open AD FS Management ((from **Control Panel > System and Security > Administrative Tools**), and go to **Application groups**, you'll find a group called **Microsoft Dynamics 365 for Operations On-premises**.

Within this group, the settings for AD FS for your Dynamics 365 application are stored.



AD FS uses the client ID and the URLs to determine whether the request for access should be honored. You will notice that the client ID from the screenshot above matches the IDs specified in the OfficeApps and OpenIDConnect sections from earlier. If both the client ID and the redirect URL don't match what the AOS is requesting, then AD FS will deny the request to authenticate. If that happens, you'll find an error in the event log on the AD FS server. There's a special event log for AD FS under **Application and Services logs > AD FS > Admin**.



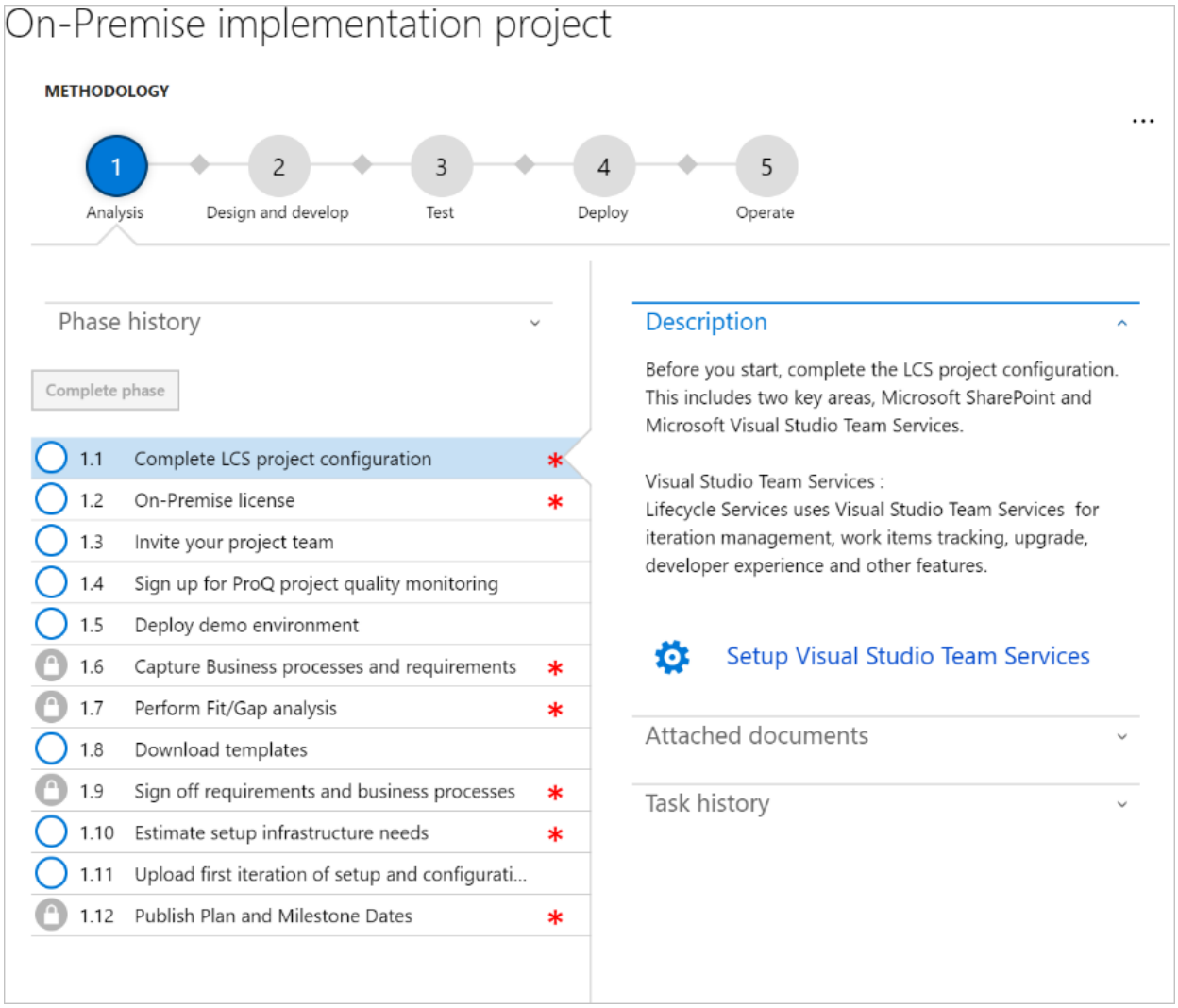
If any of the AD FS application group setup is incorrect, you're likely see an error in the event log that explains the value it was looking for, so you can determine what is set incorrectly.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Set up on-premises projects in Lifecycle Services (LCS)

2/18/2021 • 2 minutes to read • Edit Online

You must use Microsoft Dynamics Lifecycle Services (LCS) to deploy and update an instance of Dynamics 365 Finance + Operations (on-premises). After you purchase a server and user license through the Volume Licensing flow or the Dynamics Price List flow, see the topic, Buy Finance + Operations (on-premises), to create an Azure AD account or use an existing Azure AD account, and then complete all the sign-up steps. You will be redirected to LCS, where an on-premises implementation project will be provisioned for you.



The on-premises project has all the tools that you require in order to implement, maintain, and operate an on-premises solution. Here are some of the tools that are available in the on-premises project:

- **Methodology** – The on-premises methodology provides best practices that will help customers implement and manage on-premises projects.
- **Business process modeler** – Business process modeler (BPM) is used to capture requirements and do fit gap analysis.
- **Cloud-hosted environments** – Cloud-hosted environments are used to deploy developer and build topologies, and to complete Dev Application Lifecycle Management (ALM) for on-premises solutions.
- **Code upgrade** – These tools will help you upgrade code to a newer release.
- **Issue search** – Search for published KBs that are related to application and platform issues.

- **Localization and translation** – Localize and translate assets.
- **Support** – File and track support incidents.
- **Project users** – Assign users to a project.
- **Project settings** – Edit project-level settings, such as connectors, the project name, organization users, and the license number.
- **Asset library** – The Asset library is a library for various assets, such as packages.
- **SharePoint online library** – Connect to an online Microsoft SharePoint library.

To start your on-premises implementation, you must follow the steps in the methodology to correctly set up the project, deploy the developer and build environments, and then deploy sandbox and production environments. To help you with deployments, two environment slots are pre-allocated to the on-premises project. One slot is for a sandbox environment, and the other slot is for a production environment. These slots will be used during the Servicing flow to help guarantee that packages are tested in the sandbox environment before they are applied in the production environment.

Set up and deploy on-premises environments (Platform update 12 and later)

2/18/2021 • 42 minutes to read • Edit Online

This topic provides information about how to plan, set up, and deploy Dynamics 365 Finance + Operations (on-premises) with Platform update 12 and later.

The Local Business Data Yammer group is available. You can post questions or feedback you may have about the on-premises deployment there.

If you have questions or feedback about the content in this topic, please post them in the **Comments** section at the bottom of this page.

Finance + Operations components

The Finance + Operations application consists of three main components:

- Application Object Server (AOS)
- Business Intelligence (BI)
- Financial Reporting/Management Reporter

These components depend on the following system software:

- Microsoft Windows Server 2016 (only English OS installations are supported)
- Microsoft SQL Server 2016 SP1 and SP2 (from Platform update 33), which has the following features:
  - Full-text index search is enabled.
  - SQL Server Reporting Services (SSRS) - This is deployed on BI virtual machines.
  - SQL Server Integration Services (SSIS) - This is deployed on AOS virtual machines.

    WARNING

    Full Text Search must be enabled.
- SQL Server Management Studio
- Standalone Microsoft Azure Service Fabric
- Microsoft Windows PowerShell 5.0 or later
- Active Directory Federation Services (AD FS) on Windows Server 2016
- Domain controller

WARNING

The domain controller must be Microsoft Windows Server 2012 R2 or later and must have a domain functional level of 2012 R2 or more. For more information about domain functional levels, see the following topics:

- What Are Active Directory Functional Levels
- Understanding Active Directory Domain Services Functional Levels
- Full 2-way trust

Lifecycle Services

Finance + Operations bits are distributed through Microsoft Dynamics Lifecycle Services (LCS). Before you can deploy, you must purchase license keys through the Enterprise Agreements channel and set up an on-premises project in LCS. Deployments can be initiated only through LCS. For more information about how to set up on-premises projects in LCS, see Set up on-premises projects in Lifecycle Services (LCS).

Authentication

The on-premises application works with AD FS. To interact with LCS, you must also configure Azure Active Directory (AAD). To complete the deployment and configure the LCS Local agent, you will need AAD. If you do not already have an AAD tenant, you can get one for free by using one of the options provided by AAD. For more information, see How to get an Azure Active Directory tenant.

Standalone Service Fabric

Finance + Operations uses standalone Service Fabric. For more information, see the Service Fabric documentation.

Setup of Finance + Operations will deploy a set of applications inside Service Fabric (SF). During deployment, each node in the cluster will be defined via configuration to have one of the following node types:

- **AOSNodeType** - Hosts the application object server (business logic).
- **OrchestratorType** - Functions as Service Fabric primary nodes, and hosts deployment- and servicing logic.
- **ReportServerType** - Hosts SSRS and reporting logic.
- **MRType** - Hosts management reporting logic.

Infrastructure

Finance + Operations falls under the standard Microsoft support policy about operation on non-Microsoft virtualization platforms, specifically VMware. For more information, see Support policy for Microsoft software. In short, we support our products in this environment. However, if we are asked to investigate an issue, we might first ask the customer to reproduce the issue without the virtualization platform or on the Microsoft virtualization platform.

If you are using VMWare, you must implement the fixes that are documented on the following web pages:

- After upgrading a virtual machine to hardware version 11, network dependent workloads experience performance degradation (2129176)
- Several issues with vmxnet3 virtual adapter

WARNING

Dynamics 365 Finance + Operations (on-premises) is not supported on any public cloud infrastructure, including Azure.

The hardware configuration includes the following components:

- Standalone Service Fabric cluster that is based on Windows Server 2016 virtual machines (VMs)
- Microsoft SQL Server (both Clustered SQL and Always-On are supported)
- AD FS for authentication
- Server Message Block (SMB) version 3 file share for storage
- Optional: Microsoft Office Server 2017

For more information, see System requirements for on-premises deployments.

Hardware layout

Plan your infrastructure and Service Fabric cluster based on the recommended sizing in Hardware sizing requirements for on-premises environments. For more information about how to plan the Service Fabric cluster, see Plan and prepare your Service Fabric standalone cluster deployment.

The following table shows an example of a hardware layout. This example is used throughout this topic to illustrate the setup. You will need to replace the machine names and IP addresses given in the following instructions with the names and IP addresses for the machines in your environment.

> **NOTE**
> The Primary node of the Service Fabric cluster must have at least three nodes. In this example, **OrchestratorType** is designated as the Primary node type.

MACHINE PURPOSE	SF NODE TYPE	MACHINE NAME	IP ADDRESS
Domain controller		DAX7SQLAODC1	10.179.108.2
AD FS		DAX7SQLAOADFS1	10.179.108.3
File server		DAX7SQLAOFILE1	10.179.108.4
SQL Always-On cluster		DAX7SQLAOSQLA01	10.179.108.5
		DAX7SQLAOSQLA02	10.179.108.6
		DAX7SQLAOSQLA	10.179.108.9
Client		SQLAOCLIENT1	10.179.108.11
AOS 1	AOSNodeType	SQLAOSF1AOS1	10.179.108.12
AOS 2	AOSNodeType	SQLAOSF1AOS2	10.179.108.13
AOS 3	AOSNodeType	SQLAOSF1AOS3	10.179.108.14
Orchestrator 1	OrchestratorType	SQLAOSF1ORCH1	10.179.108.15
Orchestrator 2	OrchestratorType	SQLAOSF1ORCH2	10.179.108.16
Orchestrator 3	OrchestratorType	SQLAOSF1ORCH3	10.179.108.17

MACHINE PURPOSE	SF NODE TYPE	MACHINE NAME	IP ADDRESS
Management Reporter node	MRType	SQLAOSMR1	10.179.108.18
SSRS node	ReportServerType	SQLAOSFBIN1	10.179.108.10

Setup

Prerequisites

Before you start the setup, the following prerequisites must be in place. The setup of these prerequisites is out of scope for this document.

- Active Directory Domain Services (AD DS) must be installed and configured in your network.
- AD FS must be deployed.
- SQL Server 2016 SP2 must be installed on the SSRS machines.
- SQL Server Reporting Services 2016 must be installed in **Native** mode on the SSRS machines.

The following prerequisite software is installed on the VMs by the infrastructure setup scripts downloaded from LCS.

NODE TYPE	COMPONENT	DETAILS
AOS	SNAC – ODBC driver 13	/sql/connect/odbc/windows/release-notes-odbc-sql-server-windows#131
AOS	SNAC – ODBC driver 17	This driver is needed for upgrading to PU15 or higher: https://aka.ms/downloadmsodbcsql
AOS	The Microsoft .NET Framework version 2.0–3.5 (CLR 2.0)	**Windows features:** NET-Framework-Features, NET-Framework-Core, NET-HTTP-Activation, NET-Non-HTTP-Activ
AOS	The Microsoft .NET Framework version 4.0–4.6 (CLR 4.0)	**Windows features:** NET-Framework-45-Features, NET-Framework-45-Core, NET-Framework-45-ASPNET, NET-WCF-Services45, NET-WCF-TCP-PortSharing45
AOS	The Microsoft .NET Framework version 4.7.2 (CLR 4.0)	https://dotnet.microsoft.com/download/thank-you/net472-offline
AOS	Internet Information Services (IIS)	**Windows features:** WAS, WAS-Process-Model, WAS-NET-Environment, WAS-Config-APIs, Web-Server, Web-WebServer, Web-Security, Web-Filtering, Web-App-Dev, Web-Net-Ext, Web-Mgmt-Tools, Web-Mgmt-Console
AOS	SQL Server Management Studio 17.2	https://go.microsoft.com/fwlink/?linkid=854085

NODE TYPE	COMPONENT	DETAILS
AOS	Microsoft Visual C++ Redistributable Packages for Microsoft Visual Studio 2013	https://support.microsoft.com/help/3179560
AOS	Microsoft Visual C++ Redistributable Packages for Microsoft Visual Studio 2017	https://lcs.dynamics.com/V2/SharedAssetLibrary > Models > "VC++ 17 Redistributables"
AOS	Microsoft Access Database Engine 2010 Redistributable	https://www.microsoft.com/download/details.aspx?id=13255
BI	.NET Framework version 2.0–3.5 (CLR 2.0)	**Windows features:** NET-Framework-Features, NET-Framework-Core, NET-HTTP-Activation, NET-Non-HTTP-Activ
BI	.NET Framework version 4.0–4.6 (CLR 4.0)	**Windows features:** NET-Framework-45-Features, NET-Framework-45-Core, NET-Framework-45-ASPNET, NET-WCF-Services45, NET-WCF-TCP-PortSharing45
BI	The Microsoft .NET Framework version 4.7.2 (CLR 4.0)	https://dotnet.microsoft.com/download/thank-you/net472-offline
BI	SQL Server Management Studio 17.2	https://go.microsoft.com/fwlink/?linkid=854085
MR	.NET Framework version 2.0–3.5 (CLR 2.0)	**Windows features:** NET-Framework-Features, NET-Framework-Core, NET-HTTP-Activation, NET-Non-HTTP-Activ
MR	.NET Framework version 4.0–4.6 (CLR 4.0)	**Windows features:** NET-Framework-45-Features, NET-Framework-45-Core, NET-Framework-45-ASPNET, NET-WCF-Services45, NET-WCF-TCP-PortSharing45
MR	The Microsoft .NET Framework version 4.7.2 (CLR 4.0)	https://dotnet.microsoft.com/download/thank-you/net472-offline
MR	Visual C++ Redistributable Packages for Visual Studio 2013	https://support.microsoft.com/help/3179560
ORCH	The Microsoft .NET Framework version 4.0–4.8 (CLR 4.0)	https://dotnet.microsoft.com/download/thank-you/net48-offline

Overview

The following steps must be completed to set up the infrastructure for Finance + Operations. Reading all the steps before you begin will make it easier to plan your setup.

1. Plan your domain name and DNS zones
2. Plan and acquire your certificates
3. Plan your users and service accounts
4. Create DNS zones, and add A records

5. Join VMs to the domain
6. Download setup scripts from LCS
7. Describe your configuration
8. Configure certificates
9. Setup VMs
10. Set up a standalone Service Fabric cluster
11. Configure LCS connectivity for the tenant
12. Set up file storage
13. Set up SQL Server
14. Configure the databases
15. Encrypt credentials
16. Set up SSIS
17. Set up SSRS
18. Configure AD FS
19. Configure a connector and install an on-premises local agent
20. Tear down CredSSP, if remoting was used
21. Deploy your Finance + Operations environment from LCS
22. Connect to your Finance + Operations environment

1. Plan your domain name and DNS zones

We recommend that you use a publicly registered domain name for your production installation of AOS. In that way, the installation can be accessed outside the network, if outside access is required.

For example, if your company's domain is contoso.com, your zone for Finance + Operations might be d365ffo.onprem.contoso.com, and the host names might be as follows:

- ax.d365ffo.onprem.contoso.com for AOS machines
- sf.d365ffo.onprem.contoso.com for the Service Fabric cluster

2. Plan and acquire your certificates

Secure Sockets Layer (SSL) certificates are required in order to secure a Service Fabric cluster and all the applications that are deployed. For your production and sandbox workloads, we recommend that you acquire certificates from a certificate authority (CA) such as DigiCert, Comodo, Symantec, GoDaddy, or GlobalSign. If your domain is set up with Active Directory Certificate Services (AD CS), you can create the certificates through AD CS. Each certificate must contain a private key that was created for key exchange, and it must be exportable to a Personal Information Exchange (.pfx) file.

Self-signed certificates can be used only for testing purposes. For convenience, the setup scripts provided in LCS include scripts that generate and export self-signed certificates. If you are using self-signed scripts, you will be instructed to run the creation scripts in later steps. As we've mentioned, these certificates can be used for testing purposes only.

Recommended settings for certificates are:

- Signature algorithm: sha256RSA
- Signature hash algorithm: sha256
- Public key: RSA (2048 bits)
- Thumbprint algorithm: sha1

PURPOSE	EXPLANATION	ADDITIONAL REQUIREMENTS

PURPOSE	EXPLANATION	ADDITIONAL REQUIREMENTS
SQL Server SSL certificate	This certificate is used to encrypt data that is transmitted across a network between an instance of SQL Server and a client application.	The domain name of the certificate should match the fully qualified domain name (FQDN) of the SQL Server instance or listener. For example, if the SQL listener is hosted on the machine DAX7SQLAOSQLA, the certificate's DNS name is DAX7SQLAOSQLA.contoso.com. CN: DAX7SQLAOSQLA.contoso.com DNS Name: DAX7SQLAOSQLA.contoso.com
Service Fabric Server certificate	This certificate is used to help secure the node-to-node communication between the Service Fabric nodes. This certificate is also used as the Server certificate that is presented to the client that connects to the cluster.	For this certificate you can also use SSL wild card certificate of your domain. For example, *.contoso.com. This is explained in more details below the table. Otherwise, use the following values: CN: sf.d365ffo.onprem.contoso.com DNS Name: sf.d365ffo.onprem.contoso.com
Service Fabric Client certificate	This certificate is used by clients to view and manage the Service Fabric cluster.	CN: client.d365ffo.onprem.contoso.com DNS Name: client.d365ffo.onprem.contoso.com
Encipherment Certificate	This certificate is used to encrypt sensitive information such as the SQL Server password and user account passwords.	The certificate must be created by using the provider **Microsoft Enhanced Cryptographic Provider v1.0**. The certificate key usage must include Data Encipherment (10) and should not include Server authentication or Client authentication. For more information, see Managing secrets in Service Fabric applications. CN: axdataenciphermentcert DNS Name: axdataenciphermentcert

PURPOSE	EXPLANATION	ADDITIONAL REQUIREMENTS
AOS SSL Certificate	This certificate is used as the Server certificate that is presented to the client for the AOS website. It's also used to enable Windows Communication Foundation (WCF)/Simple Object Access Protocol (SOAP) certificates.	You can use the same wild card certificate that you used as the Service Fabric Server certificate. Otherwise, use the following values: CN: ax.d365ffo.onprem.contoso.com DNS Name: ax.d365ffo.onprem.contoso.com
Session Authentication certificate	This certificate is used by AOS to help secure a user's session information.	This certificate is also the File Share certificate that will be used at the time of deployment from LCS. CN: SessionAuthentication DNS Name: SessionAuthentication
Data Encryption certificate	This certificate is used by the AOS to encrypt sensitive information.	This must be created using the provider **Microsoft Enhanced RSA and AES Cryptographic Provider**. CN: DataEncryption DNS Name: DataEncryption
Data Signing certificate	This certificate is used by the AOS to encrypt sensitive information.	This is separate from the Data Encryption certificate and must be created using the provider **Microsoft Enhanced RSA and AES Cryptographic Provider**. CN: DataSigning DNS Name: DataSigning
Financial Reporting client certificate	This certificate is used to help secure the communication between the Financial Reporting services and the AOS.	CN: FinancialReporting DNS Name: FinancialReporting
Reporting certificate	This certificate is used to help secure the communication between SSRS and the AOS.	**Do not reuse the Financial Reporting Client certificate.** CN: ReportingService DNS Name: ReportingService

PURPOSE	EXPLANATION	ADDITIONAL REQUIREMENTS
On-Premises local agent certificate	This certificate is used to help secure the communication between a local agent that is hosted on-premises and on LCS. This certificate enables the local agent to act on behalf of your Azure AD tenant, and to communicate with LCS to orchestrate and monitor deployments. **Note:** Only 1 on-premises local agent certificate is needed for a tenant.	CN: OnPremLocalAgent DNS Name: OnPremLocalAgent

SSL wild card certificate of your domain can be used to combine Service Fabric Server certificate and AOS SSL certificate.

The following is an example of a Service Fabric Server certificate combined with an AOS SSL certificate.

Subject name

```
CN = *.d365ffo.onprem.contoso.com
```

Subject alternative names

```
DNS Name=ax.d365ffo.onprem.contoso.com
DNS Name=sf.d365ffo.onprem.contoso.com
DNS Name=*.d365ffo.onprem.contoso.com
```

> **NOTE**
>
> The wild card certificate allows you to secure only the first-level subdomain of the domain to which it is issued.

3. Plan your users and service accounts

You must create several user or service accounts for Finance + Operations to work. You must create a combination of group managed service accounts (gMSAs), domain accounts, and SQL accounts. The following table shows the user accounts, their purpose, and example names that will be used in this topic.

USER ACCOUNT	TYPE	PURPOSE	USER NAME
Financial Reporting Application Service Account	gMSA		Contoso\svc-FRAS$
Financial Reporting Process Service Account	gMSA		Contoso\svc-FRPS$
Financial Reporting Click Once Designer Service Account	gMSA		Contoso\svc-FRCO$

USER ACCOUNT	TYPE	PURPOSE	USER NAME
AOS Service Account	gMSA	This user should be created for future proofing. We plan to enable AOS to work with the gMSA in upcoming releases. By creating this user at the time of setup, you will help to ensure a seamless transition to the gMSA.	Contoso\svc-AXSF$
AOS Service Account	Domain account	AOS uses this user in the general availability (GA) release.	Contoso\AXServiceUser
AOS SQL DB Admin user	SQL user	Finance + Operations uses this user to authenticate with SQL*. This user will also be replaced by the gMSA user in upcoming releases**.	AXDBAdmin
Local Deployment Agent Service Account	gMSA	This account is used by the local agent to orchestrate the deployment on various nodes.	Contoso\Svc-LocalAgent$

* If the password of the SQL user contains special characters, this could cause problems during deployment.

** The SQL user name and password for SQL authentication are secured because they are encrypted and stored in the file share.

4. Create DNS zones and add A records

DNS is integrated with AD DS, and lets you organize, manage, and find resources in a network. The following instructions provide steps to create a DNS forward lookup zone and A records for the AOS host name and Service Fabric cluster. In this example setup, the DNS zone name is d365ffo.onprem.contoso.com, and the A records/host names are as follows:

- **ax**.d365ffo.onprem.contoso.com for AOS machines
- **sf**.d365ffo.onprem.contoso.com for the Service Fabric cluster

Add a DNS zone

Use the following procedure to add a DNS zone.

1. Sign in to the domain controller machine, select **Start**, and start DNS Manager by typing **dnsmgmt.msc** and selecting the **dnsmgmt (DNS)** application.
2. Right-click the domain controller name in the console tree, and then select **New Zone** > **Next**.
3. Select **Primary Zone**.
4. Leave the **Store the zone in Active Directory (available only if the DNS Server is a writeable domain controller** check box selected, and then select **Next**.
5. Select **To all DNS Servers running on Domain Controllers in this domain: Contoso.com**, and then select **Next**.
6. Select **Forward Lookup Zone**, and then select **Next**.
7. Enter the zone name for your setup, and then select **Next**. For example, enter **d365ffo.onprem.contoso.com**.
8. Select **Do not allow dynamic updates**, and then select **Next**.

9. Select **Finish**.

Set up an A record for AOS

In the new DNS zone, create one A record that is named **ax.d365ffo.onprem.contoso.com** for **each** Service Fabric cluster node of the **AOSNodeType** type. Don't create A records for the other node types.

1. Find the newly created zone under the **Forward Lookup Zones** folder in DNS Manager.
2. Right-click the new zone, and then select **New Host**.
3. Enter the name and IP address of the Service Fabric node. (For example, enter **ax** as the name and enter **10.179.108.12** as the IP address.) Select **Add Host**.
4. Do not select either check box.
5. Repeat steps 1-4 for each AOS node.

Set up an A record for the orchestrator

In the new DNS zone, create an A record that is named **sf.d365ffo.onprem.contoso.com** for **each** Service Fabric cluster node of the **OrchestratorType** type. Don't create A records for the other node types.

1. Right-click the new zone, and then select **New Host**.
2. Enter the name and IP address of the Service Fabric node. (For example, enter **sf** as the name and enter **10.179.108.15** as the IP address.) Select **Add Host**.
3. Do not select either check box.
4. Repeat for each Orchestrator node.

5. Join VMs to the domain

Join each VM to the domain by completing the steps in the Join a Computer to a Domain document. Alternatively, use the following Windows PowerShell script.

```
$domainName = Read-Host -Prompt 'Specify domain name (ex: contoso.com)'
Add-Computer -DomainName $domainName -Credential (Get-Credential -Message 'Enter domain credential')
```

> **IMPORTANT**
>
> You must restart the VMs after you join them to the domain.

6. Download setup scripts from LCS

We have provided several scripts to help improve the setup experience. Follow these steps to download the setup scripts from LCS.

> **IMPORTANT**
>
> The scripts must be executed from a computer in the same domain that the on-premises infrastructure is in.

1. Sign in to LCS.
2. On the dashboard, select the **Shared asset library** tile.
3. On the **Model** tab, in the grid, select the **Dynamics 365 for Operations on-premises - Deployment scripts** row.
4. Select **Versions**, and then download the latest version of the zip file for the scripts.

> **NOTE**
>
> If you need the older version for Platform update 8 or Platform update 11, download version 1.

5. Right-click the zip file, and then select **Properties**. In the dialog box, select the **Unblock** check box.
6. Copy the zip file to the machine that will be used to execute the scripts.
7. Unzip the files into a folder that is named **infrastructure**.

> **IMPORTANT**
> Ensure all edits are made to the ConfigTemplate.xml file in this folder.

7. Describe your configuration

The infrastructure setup scripts use the following configuration files to drive the setup.

- infrastructure\ConfigTemplate.xml
- infrastructure\D365FO-OP\NodeTopologyDefinition.xml
- infrastructure\D365FO-OP\DatabaseTopologyDefinition.xml

> **NOTE**
> Configuration files must be updated based on your environment for the setup scripts to work correctly. Be sure to update these files with the proper computer names, IP addresses, service accounts, and domain based on your setup.

infrastructure\ConfigTemplate.xml describes:

- Service Accounts that are needed for the application to operate
- Certificates necessary for securing communications
- Database configuration
- Service Fabric cluster configuration

> **IMPORTANT**
> Make sure that there are three fault domains for OrchestratorType when you configure Service Fabric cluster.
> Make sure that no more than one type of node is deployed in a single machine when you configure Service Fabric cluster.

For each Service Fabric node type, **infrastructure\D365FO-OP\NodeTopologyDefinition.xml** describes:

- The mapping between each node type and the application, domain and service accounts, and certificates.
- Whether to enable the UAC.
- Prerequisites for Windows features and system software.
- Whether strong name validation should be enabled.
- List of firewall ports to be opened.

For each database, **infrastructure\D365FO-OP\DatabaseTopologyDefinition.xml** describes:

- The database settings.
- The mappings between users and roles.

Create gMSA and domain user accounts

1. Navigate to the machine that has the unzipped infrastructure scripts in the **infrastructure** folder.
2. Copy the **infrastructure** folder to the domain controller machine.
3. Start Windows PowerShell in elevated mode, change the directory to the **infrastructure** folder, and run

the following commands.

> **IMPORTANT**
> The following script doesn't create a domain user AxServiceUser for you. You must create it yourself.

```
Import-Module .\D365FO-OP\D365FO-OP.psd1
New-D365FOGMSAAccounts -ConfigurationFilePath .\ConfigTemplate.xml
```

4. Add the AOS Service Accounts, **Contoso\svc-AXSF$** and **Contoso\AXServiceUser** to the local administrators group for all AOS machines. For more information, see Add a member to local group.

5. If you must make changes to accounts or machines, update the ConfigTemplate.xml file in the original **infrastructure** folder, copy it to this machine and then run the following script.

```
Update-D365FOGMSAAccounts -ConfigurationFilePath .\ConfigTemplate.xml
```

8. Configure certificates

1. Navigate to the machine that has the **infrastructure** folder.

2. Generate certificates:

   a. If you must generate self-signed certificates:

      a. Set the **generateSelfSignedCert** attribute to **true**. Only set this for the certificates that you need to generate.
      b. Run the following command. The script will create the certificates. Put the certificates in the CurrentUser\My certificate store on the machine, and update the thumbprints in the XML file.

```
# Create self-signed certs
.\New-SelfSignedCertificates.ps1 -ConfigurationFilePath .\ConfigTemplate.xml
```

   b. If you want to generate Active Directory Certificate Services (AD CS) certificates:

      a. Set the **generateADCSCert** attribute to **false** for the certificates that you don't want generated.
      b. Run the following commands. The script will create the certificate templates in AD CS. Generate the certificates from the templates, place the certificates in the CurrentUser\My certificate store on the machine, and update the thumbprints in the XML file.

```
.\New-ADCSCertificates.ps1 -ConfigurationFilePath .\ConfigTemplate.xml -CreateTemplates
.\New-ADCSCertificates.ps1 -ConfigurationFilePath .\ConfigTemplate.xml
```

> **NOTE**
> The AD CS scripts need to run on a Domain Controller, or a Windows Server with Remote Server Admin Tools installed.

3. If you're using SSL certificates that were already generated, skip the certificate generation and update the thumbprints in the configTemplate.xml file. The certificates need to be installed in the CurrentUser\My store and their private keys must be exportable.

WARNING

Because of a leading not-printable special character, which is difficult to determine when present, the cert manager should not be used to copy thumbprints. If the not-printable special character is present, you will get the error, **X509 certificate not valid**. To retrieve the thumbprints, see results from PowerShell commands or run the following commands in PowerShell.

```
dir cert:\CurrentUser\My
dir cert:\LocalMachine\My
dir cert:\LocalMachine\Root
```

4. Specify a semi-colon separated list of users or groups in the **ProtectTo** tag for each certificate. Only Active directory users and groups specified in the **ProtectTo** tag will have permissions to import the certificates that are exported using the scripts. Passwords are not supported by the script to protect the exported certificates

5. Export the certificates into .pfx files. As part of the export, this script will check that your certificates have the correct cryptographic provider set.

```
# Exports Pfx files into a directory VMs\<VMName>, all the certs will be written to
infrastructure\Certs folder.
.\Export-PfxFiles.ps1 -ConfigurationFilePath .\ConfigTemplate.xml
```

9. Setup VMs

1. Export the scripts that must be run on each VM.

```
# Exports the script files to be execute on each VM into a directory VMs\<VMName>.
.\Export-Scripts.ps1 -ConfigurationFilePath .\ConfigTemplate.xml
```

2. Download the following Microsoft Windows Installers (MSIs) into a file share that is accessible by all VMs.

COMPONENT	DOWNLOAD LINK	EXPECTED FILE NAME
SNAC – ODBC driver 13	/sql/connect/odbc/windows/release-notes-odbc-sql-server-windows#131	msodbcsql.msi
SNAC – ODBC driver 17	https://aka.ms/downloadmsodbcsql	msodbcsql_17.msi
Microsoft SQL Server Management Studio 17.5	/sql/ssms/download-sql-server-management-studio-ssms	SSMS-Setup-*.exe
Microsoft Visual C++ Redistributable Packages for Microsoft Visual Studio 2013	https://support.microsoft.com/help/3179560	vcredist_x64.exe
Microsoft Visual C++ Redistributable Packages for Microsoft Visual Studio 2017	Go to https://lcs.dynamics.com/V2/SharedAssetLibrary, select **Model** as the asset type, and then select **VC++ 17 Redistributables**.	vc_redist.x64_14_16_27024.exe
Microsoft Access Database Engine 2010 Redistributable	https://www.microsoft.com/download/details.aspx?id=13255	AccessDatabaseEngine_x64.exe

COMPONENT	DOWNLOAD LINK	EXPECTED FILE NAME
The Microsoft .NET Framework version 4.8 (CLR 4.0)	https://dotnet.microsoft.com/download/thank-you/net48-offline	ndp48-x86-x64-allos-enu.exe
The Microsoft .NET Framework version 4.7.2 (CLR 4.0)	https://dotnet.microsoft.com/download/thank-you/net472-offline	ndp472-x86-x64-allos-enu.exe

IMPORTANT

- Make sure the Microsoft SQL Server Management Studio setup is in the same language as the operating system of the target machine.
- Make sure that the installer files have the names that are specified in the **"Expected file name"** column of the preceding table.
- You may need to rename some of the downloads if the **"Expected file name"** is different. Failure to do so will result in errors when running the "Configure-PreReqs.ps1" script.
- When you download **VC++ 17 Redistributables**, the executable file is inside the zip file.

Follow these steps for each VM, or use remoting from a single machine

NOTE

The following section requires execution on multiple VMs. This process can be eased by using the supplied remoting scripts, which provide the option of running the necessary scripts from a single machine, such as the same machine used to execute `.\Export-Scripts.ps1`. The remoting scripts, when available, are declared after a "`# If Remoting`" comment in the PowerShell sections. When the remoting scripts are used, you may not need to execute the remaining scripts in a section, please see the section text for cases such as that. Remoting uses WinRM and requires CredSSP to be enabled in certain cases. The enabling and disabling of CredSSP is handled by the remoting module on a per-execution basis. Keeping CredSSP enabled when it is not in use is not advised, as it introduces security risks in the shape of credential theft. See the Tear down CredSSP section when you are finished setting up.

1. Copy the contents of each infrastructure\VMs<VMName> folder into the corresponding VM (if remoting scripts are used, they will automatically copy the content to the target VMs), and then run the following scripts as an Administrator.

```
# Install pre-req software on the VMs.

# If Remoting, execute
# .\Configure-PreReqs-AllVMs.ps1 -MSIFilePath <share folder path of the MSIs> -ConfigurationFilePath
.\ConfigTemplate.xml

.\Configure-PreReqs.ps1 -MSIFilePath <path of the MSIs>
```

IMPORTANT

1. Each time you are prompted, restart the machine. Make sure that you rerun the `.\Configure-PreReqs.ps1` script after each restart until all of the prerequisites are installed. In the case of remoting, rerun the AllVMs script when all of the machines are back online.
2. When you use the remoting script, ensure that the current user has access to the share folder of MSIs.
3. When you use the remoting script, ensure no user is accessing the AOSNoteType, MRType, and ReportServerType type machines. Otherwise, the remoting script will fail to restart the computer because of the users being logged on to the computer.

2. Run the following scripts, if they exist, to complete the VM setup.

```
# If Remoting, only execute
# .\Complete-PreReqs-AllVMs.ps1 -ConfigurationFilePath .\ConfigTemplate.xml

# Note: Script "Add-GMSAOnVM.ps1" is not present on BI node
.\Add-GMSAOnVM.ps1
.\Import-PfxFiles.ps1
.\Set-CertificateAcls.ps1
```

3. Run the following script to validate the VM setup.

```
# If Remoting, execute
# .\Test-D365FOConfiguration-AllVMs.ps1 -ConfigurationFilePath .\ConfigTemplate.xml

.\Test-D365FOConfiguration.ps1
```

> **IMPORTANT**
> If remoting was used, be sure to execute the clean up steps when the setup is complete. See the 20. Tear down CredSSP section.

10. Set up a standalone Service Fabric cluster

1. Download the Service Fabric standalone installation package onto one of your Service Fabric nodes. After the zip file is downloaded, unblock it by right-clicking the zip file and then selecting **Properties**. In the dialog box, select the **Unblock** check box in the lower right.
2. Copy the zip file to one of the nodes in the Service Fabric cluster, and unzip it. Ensure the **infrastructure** folder has access to this folder.
3. Navigate to the **infrastructure** folder and execute the following command to generate the Service Fabric ClusterConfig.json file.

```
.\New-SFClusterConfig.ps1 -ConfigurationFilePath .\ConfigTemplate.xml -TemplateConfig
<ServiceFabricStandaloneInstallerPath>\ClusterConfig.X509.MultiMachine.json
```

4. Additional modifications to your cluster configuration may be necessary based on your environment. For more information, see, Step 1B: Create a multi-machine cluster, Secure a standalone cluster on Windows using X.509 certificates, and Create a standalone cluster running on Windows Server.
5. Copy the generated ClusterConfig.json file to the <ServiceFabricStandaloneInstallerPath>.
6. Navigate to the <ServiceFabricStandaloneInstallerPath> in Windows PowerShell by using elevated privileges. Run the following command to test ClusterConfig.

```
.\TestConfiguration.ps1 -ClusterConfigFilePath .\clusterConfig.json
```

7. If the test is successful, run the following command to deploy the cluster.

```
# If using offline (internet-disconnected) install
# .\CreateServiceFabricCluster.ps1 -ClusterConfigFilePath .\ClusterConfig.json -
FabricRuntimePackagePath <Path to MicrosoftAzureServiceFabric.cab download>

.\CreateServiceFabricCluster.ps1 -ClusterConfigFilePath .\ClusterConfig.json
```

8. After the cluster is created, open the Service Fabric explorer on any client machine to validate the

installation.

a. Install the Service Fabric client certificate in CurrentUser\My if it isn't already installed.
b. Go to **IE settings** > **Compatibility Mode**, and clear the **Display Intranet sites in compatibility mode** check box.
c. Go to `https://sf.d365ffo.onprem.contoso.com:19080`, where sf.d365ffo.onprem.contoso.com is the host name of the Service Fabric cluster that is specified in the zone. If DNS name resolution isn't configured, use the IP address of the machine.
d. Select the client certificate. The **Service Fabric explorer** page appears.
e. Verify that all nodes are appear as green.

> **IMPORTANT**
> If your client machine is a server machine like Windows Server 2016, you must turn off the IE Enhanced Security Configuration when you access the **Service Fabric explorer** page. If any antivirus software is installed, ensure you set exclusion following the guidance in the Service Fabric documentation.

11. Configure LCS connectivity for the tenant

Deployment and servicing of Finance + Operations is orchestrated through LCS by using an on-premises local agent. To establish connectivity from LCS to the Finance + Operations tenant, you must configure a certificate that enables the local agent to act on behalf on your Azure AD tenant (for example, Contoso.onmicrosoft.com).

Use the on-premises agent certificate that you acquired from a certificate authority or the self-signed certificate that you generated by using scripts.

The on-premises agent certificate can be reused across multiple sandbox and production environments per tenant.

Only user accounts that have the Global Administrator directory role can add certificates to authorize LCS. By default, the person who signs up for Microsoft 365 for your organization is the global administrator for the directory.

> **IMPORTANT**
> - You must configure the certificate exactly **one** time per tenant. All on-premises environments under the same tenant must use the same certificate to connect with LCS.
> - If you run this in a server machine like Windows Server 2016, you must turn off the IE Enhanced Security Configuration temporarily. If you don't, the Azure login window content will be blocked.

1. Sign in to the customer's Azure portal to verify that you have the Global Administrator directory role.
2. Determine whether the certificate is already registered by running the following script from the **Infrastructure** folder.

```
# If you have issues downloading the Azure PowerShell Az module, run the following:
# [Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12

Install-Module Az
Import-Module Az
.\Add-CertToServicePrincipal.ps1 -CertificateThumbprint 'OnPremLocalAgent Certificate Thumbprint' -
Test
```

> **IMPORTANT**
> If you previously installed AzureRM, please remove it as it may not be compatible with any existing AzureRM installs in PowerShell 5.1 for Windows. For more information, Migrate Azure PowerShell from AzureRM to Az.

3. If the script indicates that the certificate isn't registered, run the following command.

```
.\Add-CertToServicePrincipal.ps1 -CertificateThumbprint 'OnPremLocalAgent Certificate Thumbprint'
```

> **NOTE**
> If you have multiple tenants associated with the login account, you can pass the tenant ID as a parameter to ensure that the context is set to the correct tenant.

```
.\Add-CertToServicePrincipal.ps1 -CertificateThumbprint 'OnPremLocalAgent Certificate Thumbprint' -
TenantId 'xxxx-xxxx-xxxx-xxxx'
```

12. Set up file storage

You must set up the following SMB 3.0 file shares:

- A file share that stores user documents that are uploaded to AOS (for example, \\DAX7SQLAOFILE1\aos-storage).
- A file share that stores the latest build and configuration files to orchestrate the deployment (for example, \\DAX7SQLAOFILE1\agent).

> **WARNING**
> Keep this file share path as short as possible to avoid exceeding the maximum path length on the files that will be put in the share.

For information about how to enable SMB 3.0, see SMB Security Enhancements.

> **IMPORTANT**
> - Secure dialect negotiation can't detect or prevent downgrades from SMB 2.0 or 3.0 to SMB 1.0. Therefore, we strongly recommend that you disable the SMB 1.0 server. By disabling the SMB 1.0 server, you can take advantage of the full capabilities of SMB encryption.
> - To help ensure that your data is protected while it's at rest in your environment, BitLocker Drive Encryption must be enabled on every machine. For information about how to enable BitLocker, see BitLocker: How to deploy on Windows Server 2012 and later.

1. On the file share machine, run the following command.

```
Install-WindowsFeature -Name FS-FileServer -IncludeAllSubFeature -IncludeManagementTools
```

2. Follow these steps to set up the \\DAX7SQLAOFILE1\aos-storage file share:

   a. In Server Manager, select **File and Storage Services** > **Shares**.

b. Select **Tasks** > **New Share** to create a new share. Name the share **aos-storage**.

c. Leave **Allow caching of share** selected.

d. Check **Encrypt data access**.

e. Grant **Modify** permissions for every machine in the Service Fabric cluster except OrchestratorType.

f. Grant **Modify** permissions for the user AOS domain user (contoso\AXServiceUser) and the gMSA user (contoso\svc-AXSF$).

> **NOTE**
> You may need to enable **Computers** under **Object Types** to add machines or enable **Service Accounts** under **Object Types** to add service accounts.

3. Follow these steps to set up the \\DAX7SQLAOFILE1\agent file share:

a. In Server Manager, select **File and Storage Services** > **Shares**.

b. Select **Tasks** > **New Share** to create a new share. Name the share **agent**.

c. Grant **Full-Control** permissions to the gMSA user for the local deployment agent (contoso\svc-LocalAgent$).

```
# Specify user names
$AOSDomainUser = 'Contoso\AXServiceUser';
$LocalDeploymentAgent = 'contoso\svc-LocalAgent$';

# Specify the path
$AosStorageFolderPath = 'D:\aos-storage';
$AgentFolderPath = 'D:\agent';

# Create new directory
$AosStorageFolder = New-Item -type directory -path $AosStorageFolderPath;
$AgentFolder = New-Item -type directory -path $AgentFolderPath;

# Create new SMB share
New-SmbShare –Name aos-storage -Path $AosStorageFolderPath -EncryptData $True
New-SmbShare –Name agent -Path $AgentFolderPath

# Set ACL for AOS storage folder
$Acl = Get-Acl $AosStorageFolder.FullName;
$Ar = New-Object system.security.accesscontrol.filesystemaccessrule($AOSDomainUser,'Modify','Allow');
$Acl.SetAccessRule($Ar);
Set-Acl $AosStorageFolder.FullName $Acl;

# Set ACL for AgentFolder
$Acl = Get-Acl $AgentFolder.FullName;
$Ar = New-Object
system.security.accesscontrol.filesystemaccessrule($LocalDeploymentAgent,'FullControl','Allow');
$Acl.SetAccessRule($Ar);
Set-Acl $AgentFolder.FullName $Acl;
```

13. Set up SQL Server

1. Install SQL Server 2016 SP2 with high availability. (Unless you're deploying in a sandbox environment, where one instance of SQL Server is sufficient. You may want to install SQL Server with high availability in sandbox environments to test high-availability scenarios.)

> **IMPORTANT**
>
> You must enable the SQL Server and Windows Authentication mode.

You can install SQL Server with high availability either as SQL clusters that include a Storage Area Network (SAN) or in an Always-On configuration. Verify that the Database Engine, SSRS, Full-Text Search, and Management Tools are already installed.

> **NOTE**
>
> Make sure that Always-On is set up as described in Select Initial Data Synchronization Page (Always On Availability Group Wizards), and follow the instructions in To Prepare Secondary Databases Manually.

2. Run the SQL service as a domain user or a group-managed service account.
3. Get an SSL certificate from a certificate authority to configure SQL Server for Finance + Operations. For testing purposes, you can create and use a self-signed certificate or an AD CS certificate. You will need to replace the computer name and domain name in the following examples.

   #### Self-signed certificate for an Always-On SQL instance

   If you are setting up testing certificates for Always-On, use the following **remoting** script. This will perform the same as the following **manual** script and steps **a-e**.

   a. Self-signed certificate

   ```
   .\New-SelfSigned-SQLCert-AllVMs.ps1 -SqlMachineNames SQL1,SQL2 -SqlListenerName SQL-LS -
   ProtectTo CONTOSO\dynuser
   ```

   b. AD CS certificate

   ```
   .\New-ADCS-SQLCert-AllVMs.ps1 -SqlMachineNames SQL1,SQL2 -SqlListenerName SQL-LS -ProtectTo
   CONTOSO\dynuser
   ```

   #### Manual self-signed steps for an Always-On SQL instance or Windows Server Failover Clustering with SQL Server

   For each node of the SQL cluster, follow these steps.

   a. Run the following PowerShell script on each of the SQL Server Always-On replicas.

   ```
   # Manually create certificate for each SQL Node (i.e. 2 nodes = 2 certificates)
   # Run script on each node
   $computerName = $env:COMPUTERNAME.ToLower()
   $domain = $env:USERDNSDOMAIN.ToLower()
   $listenerName = 'dax7sqlaosqla'
   $cert = New-SelfSignedCertificate -Subject "$computerName.$domain" -DnsName "$listenerName.$domain",
   $listenerName, $computerName -Provider 'Microsoft Enhanced RSA and AES Cryptographic Provider' -
   CertStoreLocation "cert:\LocalMachine\My" -KeyAlgorithm "RSA" -HashAlgorithm "sha256" -KeyLength 2048
   ```

   b. Grant certificate permissions to the account that is used to run the SQL service.
      a. Open Manage Computer Certificates (**certlm.msc**).
      b. Right-click the certificate created, and then select **Tasks** > **Manage Private Keys**.
      c. Add in the SQL Server service account and grant Read access.

c. Enable **ForceEncryption** and the new **Certificate** in Microsoft SQL Server Configuration Manager.
   a. In **SQL Server Configuration Manager**, expand **SQL Server Network Configuration**, right-click **Protocols for [server instance]**, and then select **Properties**.
   b. In the **Properties** dialog box, on the **Certificate** tab, select the desired certificate from the drop-down menu for the **Certificate** box.
   c. In the **Properties** dialog box, on the **Flags** tab, in the **ForceEncryption** box, select **Yes**.
   d. Select **OK** to save.
d. Export the certificate (.cer file) from each SQL cluster node, and install it in the trusted root of each Service Fabric node. You will have a minimum of 2 certificates for the Always-On cluster, but there may be more if you have additional replicas.
e. Restart the SQL Server service.

> **NOTE**
> For more information, see How to enable SSL encryption for an instance of SQL Server by using Microsoft Management Console.

> **IMPORTANT**
> If remoting was used, be sure to execute the clean up steps when the setup is complete. See the 20. Tear down CredSSP section for more information.

14. Configure the databases

1. Sign in to LCS.
2. On the dashboard, select the **Shared asset library** tile.
3. On the **Model** tab, select the demo data for the release that you want and download the zip file.

RELEASE	DEMO DATA
On-premises General Availability (GA) release	Dynamics 365 for Operations on-premises - Demo data
On-premises Platform Update 11 Nov 2017 release	Dynamics 365 for Operations on-premises, Enterprise edition - Update 11 Demo data
On-premises Platform Update 12 Mar 2018 release	Dynamics 365 for Operations on-premises, Enterprise edition - Update 12 Demo data

4. The zip file contains empty and demo data .bak files. Select the .bak file, based on your requirements. For example, if you require demo data, download the AxBootstrapDB_Demodata.bak file.
5. Ensure the database section in the infrastructure\ConfigTempate.xml is configured correctly with the following:
   a. The database name.
   b. The db file and log settings. The db settings should not be lower than the defaults specified.
   c. The path to the backup file downloaded from LCS Shared Asset library. The default name for the Finance + Operations database is AXDB.

> **WARNING**
>
> - The user running the SQL service and the user running the scripts should have READ access on the folder or share where the backup file is located.
> - If a database with the same name exists, the database will be reused.

6. Copy the **infrastructure** folder to the SQL Server machine and navigate to it in a PowerShell window with elevate privileges.

Configure the OrchestratorData database

1. Execute the following script.

```
.\Initialize-Database.ps1 -ConfigurationFilePath .\ConfigTemplate.xml -ComponentName Orchestrator
```

The script will do the following:

- Create an empty database named **OrchestratorData**. This database is used by the on-premises local agent to orchestrate deployments.
- Grant the local agent gMSA (svc-LocalAgent$) **db_owner** permissions on the database.

Configure the Finance + Operations database

1. Execute the following scripts.

```
.\Initialize-Database.ps1 -ConfigurationFilePath .\ConfigTemplate.xml -ComponentName AOS
.\Configure-Database.ps1 -ConfigurationFilePath .\ConfigTemplate.xml -ComponentName AOS
```

The **Initialize-Database.ps1** script will do the following:

a. Restore the database from the specified backup file.

b. Create a new user that has SQL authentication enabled (axdbadmin).

c. Map users to database roles based on the following table for AXDB.

USER	TYPE	DATABASE ROLE
svc-AXSF$	gMSA	db_owner
svc-LocalAgent$	gMSA	db_owner
svc-FRPS$	gMSA	db_owner
svc-FRAS$	gMSA	db_owner
axdbadmin	SqlUser	db_owner

d. Map users to database roles based on the following table for TempDB.

USER	TYPE	DATABASE ROLE
svc-AXSF$	gMSA	db_datareader, db_datawriter, db_ddladmin

USER	TYPE	DATABASE ROLE
axdbadmin	SqlUser	db_datareader, db_datawriter, db_ddladmin

The **Configure-Database.ps1** script will do the following:

a. Set READ_COMMITTED_SNAPSHOT ON
b. Set ALLOW_SNAPSHOT_ISOLATION ON
c. Set the specified database file and log settings
d. GRANT VIEW SERVER STATE TO axdbadmin
e. GRANT ALTER ANY EVENT SESSION TO axdbadmin
f. GRANT VIEW SERVER STATE TO [contoso\svc-AXSF$]
g. GRANT ALTER ANY EVENT SESSION TO [contoso\svc-AXSF$]

2. Run the following command to reset the database users.

```
.\Reset-DatabaseUsers.ps1 -DatabaseServer '<FQDN of the SQL server>' -DatabaseName '<AX database name>'
```

Configure the Financial Reporting database

1. Execute the following script.

```
.\Initialize-Database.ps1 -ConfigurationFilePath .\ConfigTemplate.xml -ComponentName MR
```

The script will do the following:

a. Create an empty database named **FinancialReporting**.

b. Map the users to database roles based on the following table.

USER	TYPE	DATABASE ROLE
svc-LocalAgent$	gMSA	db_owner
svc-FRPS$	gMSA	db_owner
svc-FRAS$	gMSA	db_owner

15. Encrypt credentials

1. On any client machine, install the encipherment certificate in the LocalMachine\My certificate store.
2. Grant the current user read access to the private key of this certificate.
3. Create the Credentials.json file, as shown here.

```
{
    "AosPrincipal": {
        "AccountPassword": "<encryptedDomainUserPassword>"
    },
    "AosSqlAuth": {
        "SqlUser": "<encryptedSqlUser>",
        "SqlPwd": "<encryptedSqlPassword>"
    }
}
```

- **AccountPassword** is the encrypted domain user password for the AOS domain user (contoso\axserviceuser).
- **SqlUser** is the encrypted SQL user (axdbadmin) that has access to the Finance + Operations database (AXDB), and **SqlPassword** is the encrypted SQL password.

4. Copy the .json file to the SMB file share, \\AX7SQLAOFILE1\agent\Credentials\Credentials.json.
5. Update the Credentials.json file with encrypted values.

```
# Service fabric API to encrypt text and copy it to the clipboard.
Invoke-ServiceFabricEncryptText -Text '<textToEncrypt>' -CertThumbprint '<DataEncipherment
Thumbprint>' -CertStore -StoreLocation LocalMachine -StoreName My | Set-Clipboard
```

> **IMPORTANT**
> Before you can invoke *Invoke-ServiceFabricEncryptText*, you need to install Microsoft Azure Service Fabric SDK. If you encounter the following error, "Invoke-ServiceFabricEncryptText is not recognized command" after you install the Azure Service Fabric SDK, restart the computer and retry.

> **WARNING**
> After you've finished invoking all **Invoke-ServiceFabricEncryptText** commands, remember to delete the Windows PowerShell history. Otherwise, your non-encrypted credentials will be visible.

16. Set up SSIS

To enable Data management and Integration workloads, SSIS must be installed on each of the AOS virtual machines. Complete the following steps on each AOS virtual machine.

1. Verify that the machine has access to the SSIS installation and open the SSIS Setup Wizard.
2. In the **Feature Selection** window, in the **Features** pane, select the **Integration Services** and **SQL Client Connectivity SDK** check boxes.
3. Complete the setup and verify that the installation was successful.

For more information, see Install integration services.

17. Set up SSRS

1. Before you begin, make sure that the prerequisites that are listed at the beginning of this topic are installed.
2. Follow the steps in Configure SQL Server Reporting Services for on-premises deployments.

> **IMPORTANT**
> You must install then database engine when you install SSRS.

18. Configure AD FS

Before you can complete this procedure, AD FS must be deployed on Windows Server 2016. For information about how to deploy AD FS, see Deployment Guide Windows Server 2016 and 2012 R2 AD FS Deployment Guide.

Finance + Operations requires additional configuration beyond the default out-of-box configuration of AD FS. The following Windows PowerShell commands must be run on the machine where the AD FS role service is installed. The user account must have enough permissions to administer AD FS. For example, the user must have a domain administrator account. For complex AD FS scenarios, consult your domain administrator.

1. Configure the AD FS identifier so that it matches the AD FS token issuer.

This command is related to adding new users using the **Import users** option on the **Users** page (**System administration** > **Users** > **Users**) in the Finance + Operations client.

```
$adfsProperties = Get-AdfsProperties
Set-AdfsProperties -Identifier $adfsProperties.IdTokenIssuer
```

2. You should disable Windows Integrated Authentication (WIA) for intranet authentication connections, unless you've configured AD FS for mixed environments. For more information about how to configure WIA so that it can be used with AD FS, see Configure browsers to use Windows Integrated Authentication (WIA) with AD FS.

   This command is related to using forms authentication upon signing into the Finance + Operations client. Other options, such as single sign-on, may be available which require additional setup.

   ```
   Set-AdfsGlobalAuthenticationPolicy -PrimaryIntranetAuthenticationProvider FormsAuthentication,
   MicrosoftPassportAuthentication
   ```

3. For sign-in, the user's email address must be an acceptable authentication input.

   This command is related to setting up email claims. Other options, such as transformation rules, may be available which require additional setup.

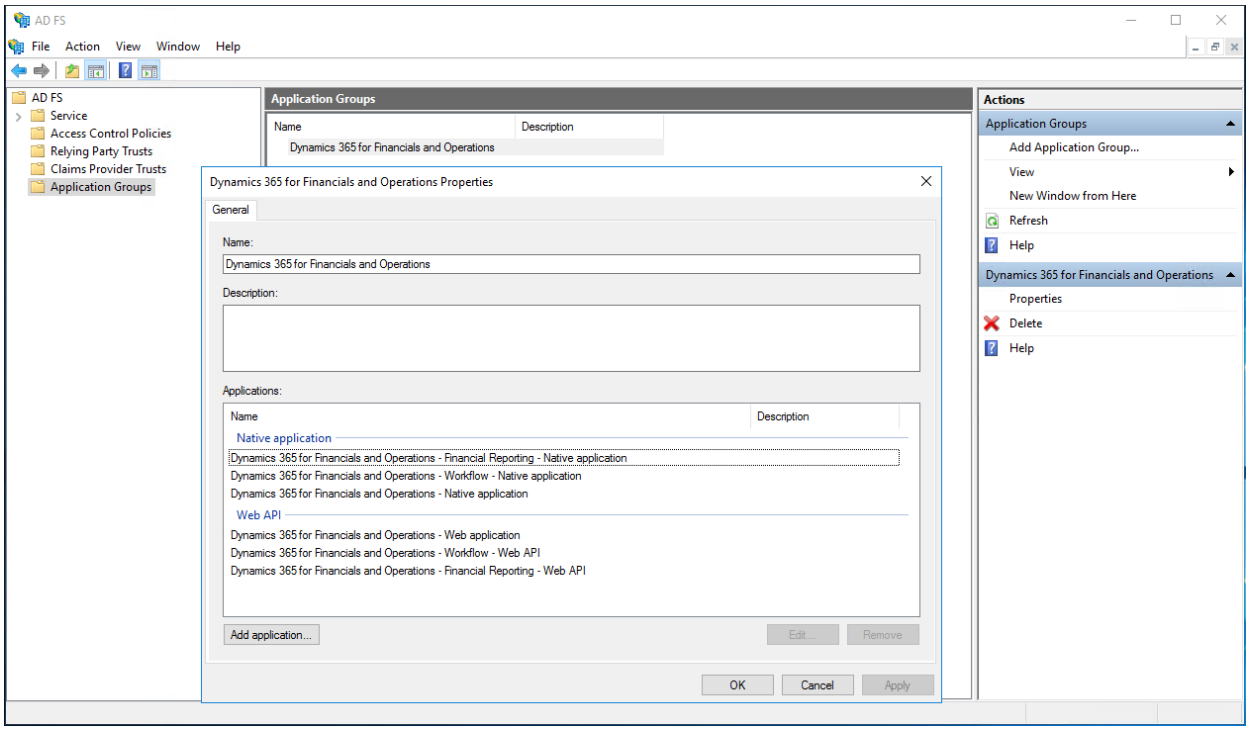
   ```
   Add-Type -AssemblyName System.Net
   $fqdn = ([System.Net.Dns]::GetHostEntry('localhost').HostName).ToLower()
   $domainName = $fqdn.Substring($fqdn.IndexOf('.')+1)
   Set-AdfsClaimsProviderTrust -TargetIdentifier 'AD AUTHORITY' -AlternateLoginID mail -LookupForests
   $domainName
   ```

In order for AD FS to trust Finance + Operations for the exchange of authentication, various application entries must be registered in AD FS under an AD FS application group. To speed up the setup process and help reduce errors, you can use the following script for registration. Copy the Publish-ADFSApplicationGroup.ps1 script and D365FO-OP directory to a machine where the AD FS role service is installed. Then run the script by using a user account that has enough permissions to administer AD FS. (For example, use an administrator account.)

For more information about how to use the script, see the documentation that is listed in the script. Make a note of the client IDs that are specified in the output, because you will need this information in LCS in a later step. Should you lose the client IDs, log in to the machine which has AD FS installed, open **Server Manager** > **Tools** > **AD FS Management** > **Application Groups** > **Microsoft Dynamics 365 for Operations On-premises** and find the client IDs under the native applications.

> **NOTE**
>
> If you want to reuse your previously configured AD FS server for additional environments, see Reuse the same AD FS instance for multiple environments.

```
# Host URL is your DNS record\host name for accessing the AOS
.\Publish-ADFSApplicationGroup.ps1 -HostUrl 'https://ax.d365ffo.onprem.contoso.com'
```



Finally, make sure that you can access the AD FS OpenID Configuration URL on a Service Fabric node of the **AOSNodeType** type. To perform this check, try to open `https://<adfs-dns-name>/adfs/.well-known/openid-configuration` in a web browser. If you receive a message that states that the site isn't secure, you haven't added your AD FS SSL certificate to the Trusted Root Certification Authorities store. This step is described in the AD FS deployment guide, and if you are using remoting, you can use the following script to install the certificate on all nodes in the Service Fabric cluster:

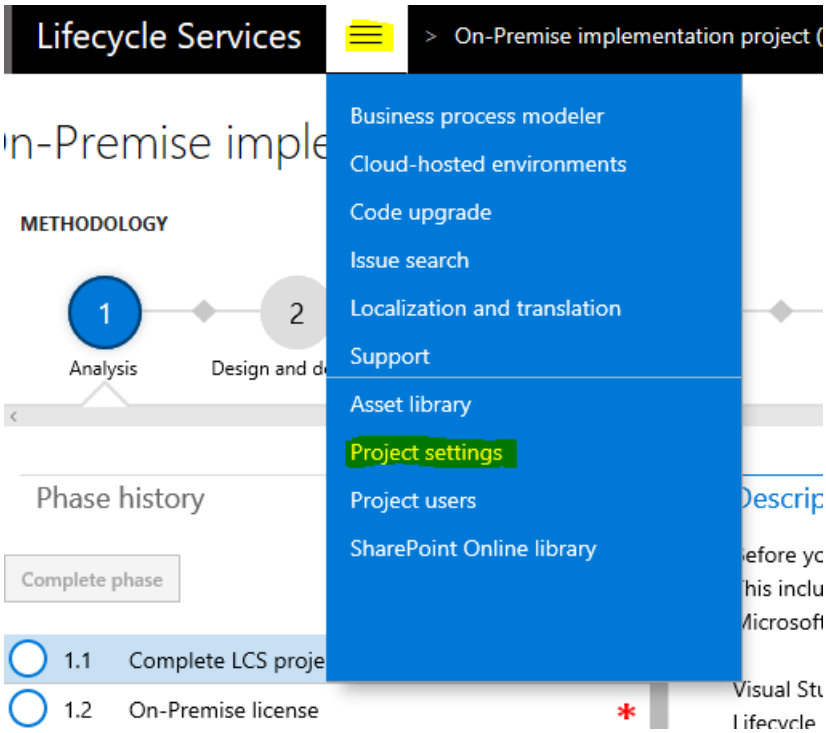
```
# If remoting, execute
.\Install-ADFSCert-AllVMs.ps1 -ConfigurationFilePath .\ConfigTemplate.xml
```

If you successfully access the URL, a JavaScript Object Notation (JSON) file is returned that contains your AD FS configuration, and you will see that your AD FS URL is trusted.

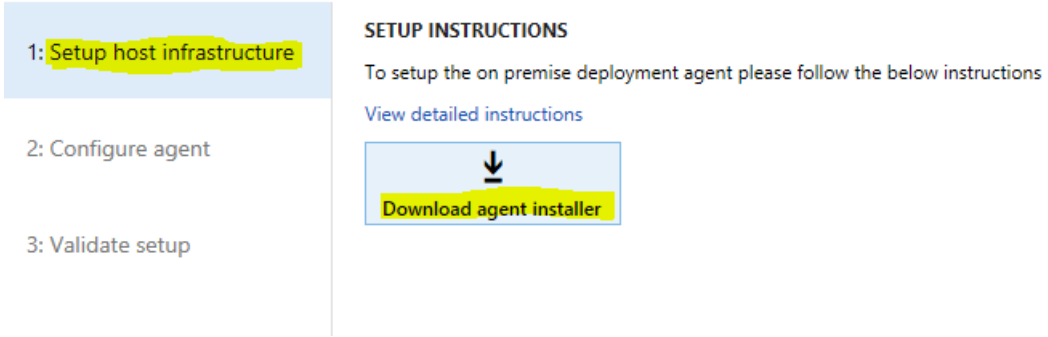
You've now completed the setup of the infrastructure. The following sections describe how to navigate to LCS to set up your connector and deploy your Finance + Operations environment.

19. Configure a connector and install an on-premises local agent

1. Sign in to LCS, and open the on-premises implementation project.
2. On the hamburger menu, select **Project settings**.



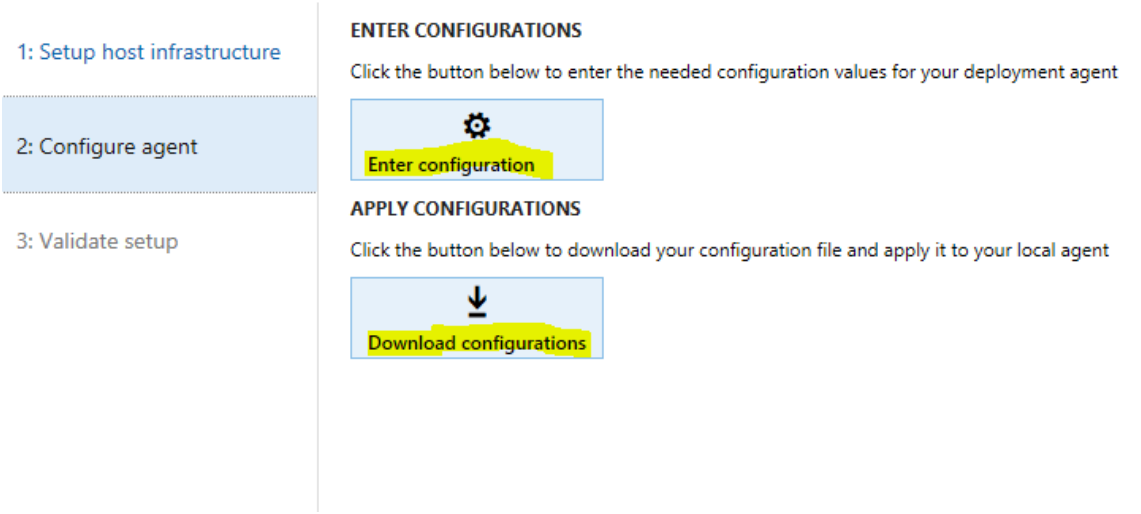
3. Select **On-premises connectors**.
4. Select **Add** to create a new connector.
5. On the **Setup host infrastructure** tab, download the agent installer.



6. Verify that the zip file is unblocked. Right-click the file, and then select **Properties**. In the dialog box, select **Unblock**.
7. Unzip the agent installer on one of the Service Fabric nodes of the **OrchestratorType** type.
8. On the **Configure agent** tab, enter the configuration settings. Execute the following script on any machine with access to it and the configuration file, to get the needed values.

```
.\Get-AgentConfiguration.ps1 -ConfigurationFilePath .\ConfigTemplate.xml
```

9. Save the configuration, and then select **Download configurations** to download the localagent-config.json configuration file.

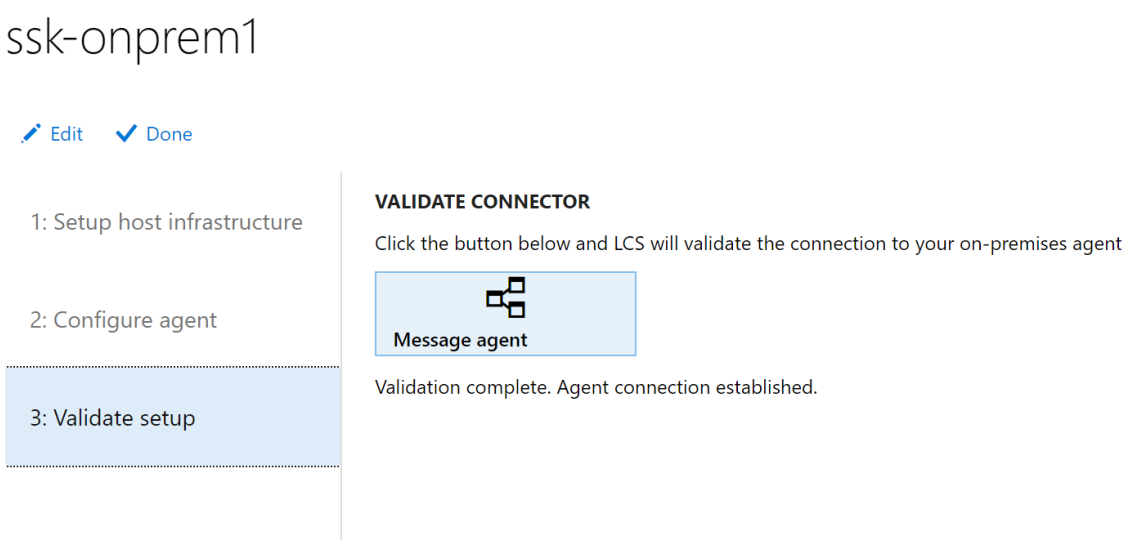


10. Copy the localagent-config.json file to the machine where the agent installer package is located.
11. In a **Command Prompt** window, run the following command by navigating to the folder that contains the agent installer.

```
LocalAgentCLI.exe Install <path of config.json>
```

> **NOTE**
> The user who runs this command must have **db_owner** permissions on the OrchestratorData database.

12. After the local agent is successfully installed, navigate back to your on-premises connector in LCS.
13. On the **Validate setup** tab, select **Message agent** to test for LCS connectivity to your local agent. When a connection is successfully established, the page will resemble the following illustration.



20. Tear down CredSSP, if remoting was used

If any of the remoting scripts were used during setup, be sure to execute the following script when there are breaks in the setup process, or the setup has finished.

```
.\Disable-CredSSP-AllVMs.ps1 -ConfigurationFilePath .\ConfigTemplate.xml
```

If the previous remoting PowerShell window was accidentally closed and CredSSP was left enabled, the script

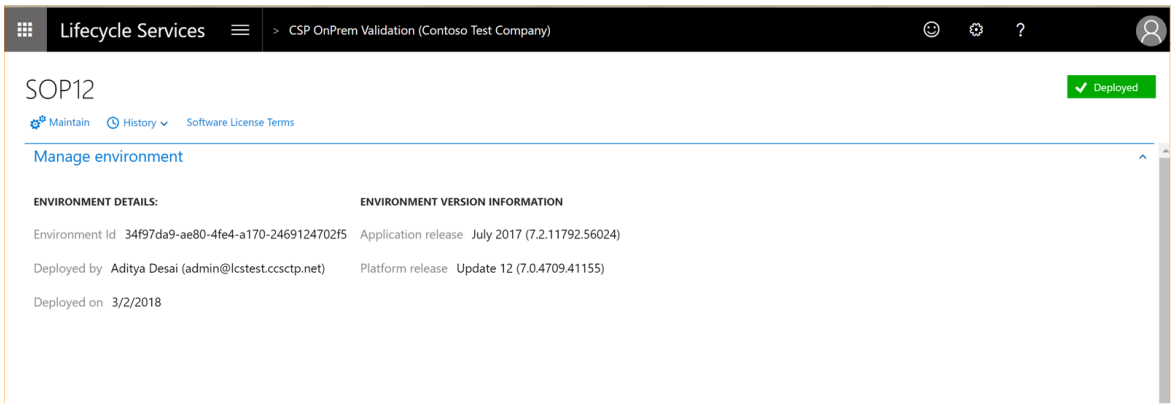
will disable it on all the machines specified in the configuration file.

21. Deploy your Finance + Operations environment from LCS

1. In LCS, navigate to your on-premises project, go to **Environment** > **Sandbox**, and then select **Configure**. Execute the following script on the primary domain controller VM, which must have access to ADFS and the DNS server settings, to get the needed values.

```
.\Get-DeploymentSettings.ps1 -ConfigurationFilePath .\ConfigTemplate.xml
```

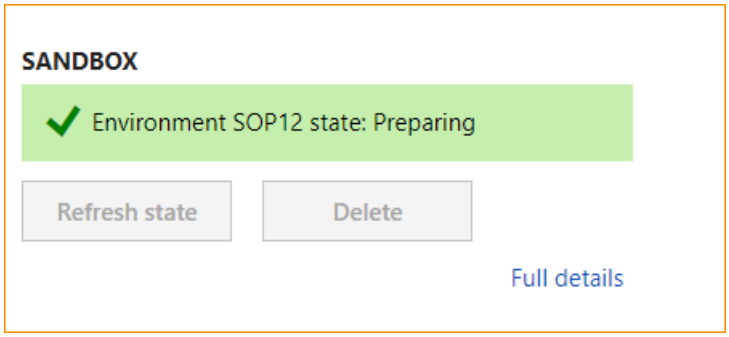
2. For new deployments, select your environment topology, and then complete the wizard to start your deployment.



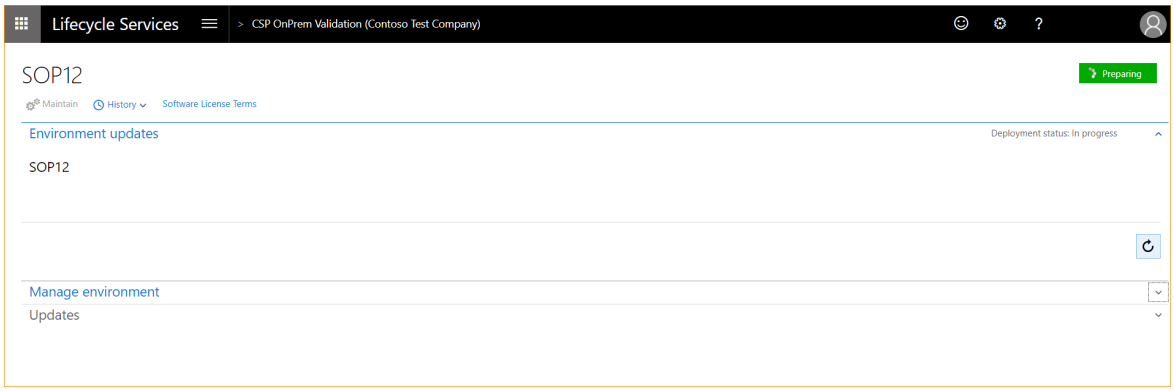
3. If you have an existing Platform update 8 or Platform update 11 deployment:

   - Update the local agent. See Update the local agent for more details.
   - Validate the local agent from LCS.
   - Deploy Platform update 12 while going through the steps in Reconfigure environments to take a new platform or topology.

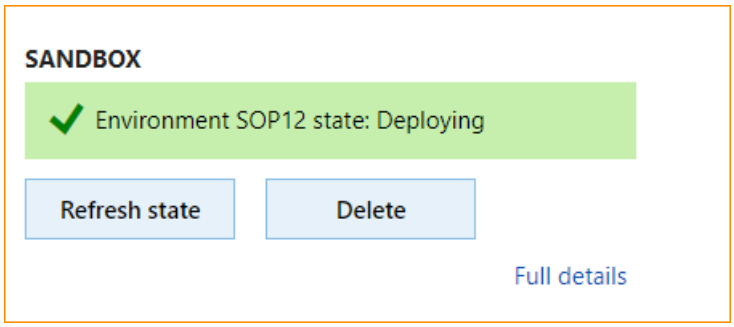
4. LCS will assemble the Service Fabric application packages for your environment during the preparation phase. It then sends a message to the local agent to start deployment. You will notice the **Preparing** status as below.

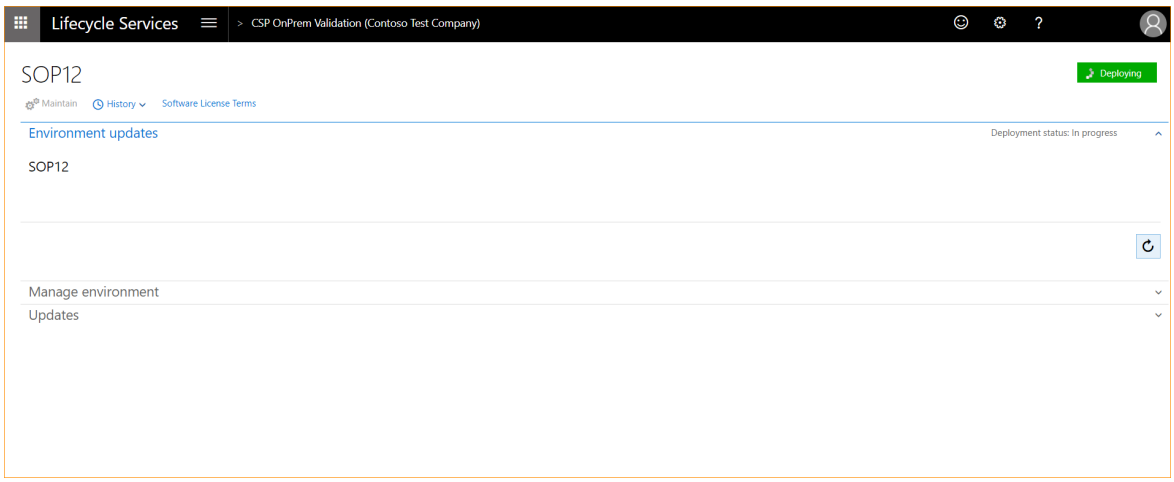


   Click **Full details** to take you to the environment details page, as shown below.

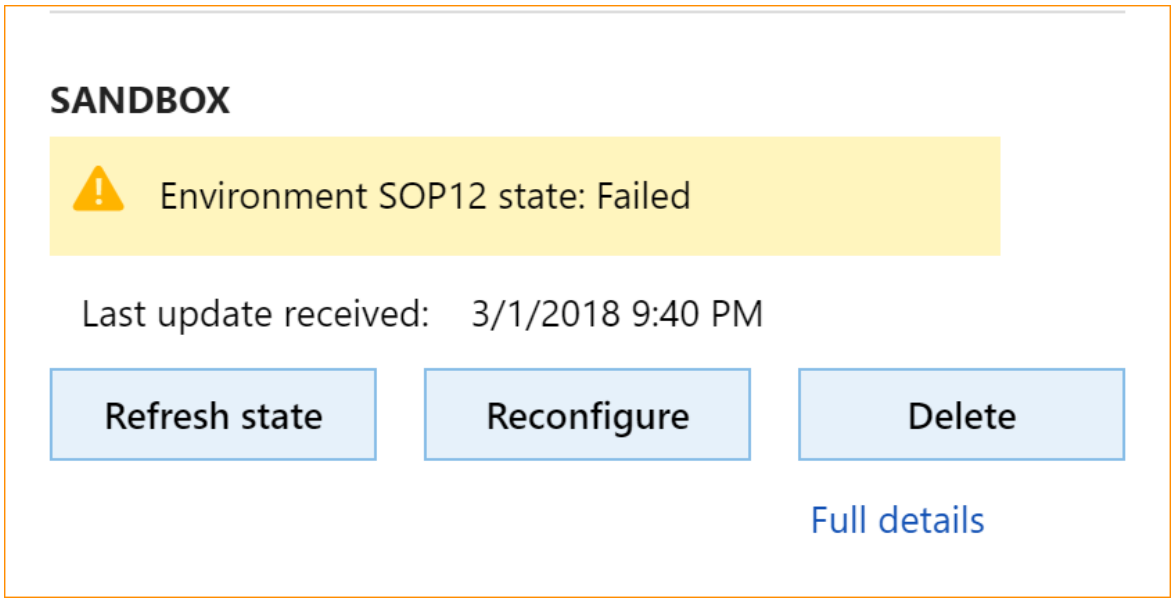


5. The local agent will now pick up the deployment request, start the deployment, and communicate back to LCS when the environment is ready. When deployment starts, the status will change to **Deploying**, as shown.

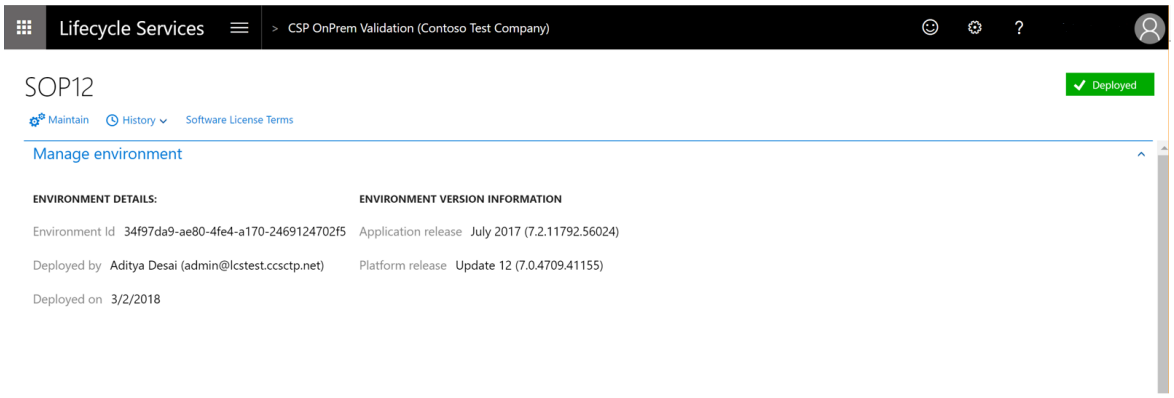




If the deployment fails, the **Reconfigure** button will become available for your environment in LCS, as shown below. Fix the underlying issue, click **Reconfigure**, update any configuration changes, and click **Deploy** to retry the deployment.



See the Reconfigure environments to take a new platform or topology topic for details about how to reconfigure. The following graphic shows a successful deployment.



22. Connect to your Finance + Operations environment

In your browser, navigate to https://[yourD365FOdomain]/namespaces/AXSF, where yourD365FOdomain is the domain name that you defined in the Plan your domain name and DNS zones section of this topic.

Additional resources

- Apply updates to on-premises deployments
- Redeploy on-premises environments
- Configure document management
- Import Electronic reporting (ER) configurations
- Document generation, publishing, and printing in on-premises deployments
- Configure proxies for on-premises environments
- Set up technical support for Finance and Operations apps
- Client internet connectivity

Known issues

Error "Key does not exist" when running the New-D365FOGMSAAccounts cmdlet

If this is your first time creating and generating group Managed Service Account passwords in your domain, you need to first create the **Key Distribution Services KDS Root Key**. For more information, see Create the Key Distribution Services KDS Root Key.

Error "The WinRM client cannot process the request" when running the remoting script Configure-Prereqs-AllVms cmdlet

You need to follow the instructions in the error message to enable the computer policy **Allow delegation fresh credentials** in all machines of Service Fabric cluster.

Error "Not process argument transformation on parameter 'Test'. Cannot convert value "System.String" to type "System.Management.Automation.SwitchParameter" when running the Config-Prereqs-AllVms cmdlet

To work around this error, remove "-Test:$Test" in line 56 of Config-Prereqs-AllVms.ps1, which is found under the **Infrastructure** folder.

Error "Not process argument transformation on parameter 'Test'. Cannot convert value "System.String" to type "System.Management.Automation.SwitchParameter" when running the Complete-Prereqs-AllVms cmdlet

To work around this error, remove "-Test:$Test" in line 56, 61 and 66 of Complete-Prereqs-AllVms.ps1 which is found under the **Infrastructure** folder.

Error "Install-WindowsFeature: The request to add or remove features on the specified server failed" when running Configure-Prereqs on MRType and ReportServerTyoe servers

.NET Framework 3.5 is required in MRType and ReportServerType servers. By default, however, .NET Framework 3.5 source files aren't included in your Windows Server 2016 installation. To work around this error, install it and specify the source files using the **source** option when you manually add new features by server manager.

Error "MSIS7628: Scope names should be a valid Scope description name in AD FS configuration" when running the Publish-ADFSApplicationGroup cmdlet

This error occurs because of an OpenID scope **allatclaims** that is required by the D365FO-OP-ADFSApplicationGroup, but it might be missing in some Windows Server 2016 installation. To work around this error, add the scope description **allatclaims** through AD FS Management\Service\Scope Descriptions.

Error "ADMIN0077: Access control policy does not exist: Permit everyone" when running the Publish-ADFSApplicationGroup cmdlet

When your AD FS is installed with a non-English version of Windows Server 2016, the permit everyone access control policy is created with your local language. Invoke the cmdlet by specifying AccessControlPolicyName parameter as: .\Publish-ADFSApplicationGroup.ps1 -HostUrl 'https://ax.d365ffo.onprem.contoso.com' -AccessControlPolicyName ''.

Installation steps for Retail channel components in an on-premises environment

2/18/2021 • 5 minutes to read • Edit Online

This topic covers the installation steps for Commerce channel components in an on-premises environment.

Overview

Channel functionality, in an on-premises environment, is enabled exclusively via use of Commerce Scale Unit (self-hosted). For an overview, see Commerce Scale Unit (self-hosted).

Unlike a cloud deployment, an on-premises environment does not enable seamless, high-availability deployment of channel components via Lifecycle Services (LCS). The only way to use channel components is by installing Commerce Scale Unit (self-hosted).

Prerequisites

Before you can start installation of channel components, you must first complete all prior installation steps for an on-premises environment. These steps are listed in Set up and deploy on-premises environments (Platform update 12 and later). In addition, version 8.1.1 must be installed in order for Commerce have full functionality. We recommend that you update to version 8.1.2.

> **NOTE**
> It is critical to ensure that a secure network, that is not publicly accessible, is used to connect Commerce Scale Unit to Headquarters. You must also restrict network access to Headquarters, so access is only allowed to known Commerce Scale Unit devices via network filtering or other means. This means that a firewall must exist and using a safe list is highly recommended.

Installation steps

1. On the previously created Application share, (not the **LocalAgent** share folder), create a new folder called **selfservicepackages** in the root directory of the share location.
2. On each AOS computer, create an easily accessible directory, such as **C:/selfservicepackages**.
3. On one AOS computer (which one does not matter), run the following PowerShell script.

```
.\RetailUpdateDatabase.ps1 -envName '<Environment name>' -AosUrl 'https://<My Environment
Name>.com/namespaces/AXSF/' -SendProductSupportTelemetryToMicrosoft
```

IMPORTANT

The above steps apply to version 10.0 and later. For the original 8.1.3 release of Retail on-premises functionality, the original version of the script delimiters must be used.

```
.\RetailUpdateDatabase.ps1 -DatabaseServer '<Database server name for AOS database>' -
DatabaseName '<Database name for AOS database>' -envName '<Environment name>' -
RetailSelfServicePackages '<Local path of Retail self-service packages, such as
**C:/selfservicepackages**>' -SendProductSupportTelemetryToMicrosoft
```

- The parameter **-envName** should be known based on creation when the environment is generated.
- The legacy parameters **-DatabaseServer** and **-DatabaseName** should be known based on the environment setup.
- The parameter **-SendProductSupportTelemetryToMicrosoft** is a required value to enable telemetry to Microsoft. This is critical to maximize support from Microsoft.
- This script will perform a variety of actions, including updating the Service user and role and updating registry keys.

4. On each AOS computer, run the following PowerShell script.

```
.\RetailUpdateDatabase.ps1 -RetailSelfServicePackages 'C:\RetailSelfService\Packages'
```

NOTE

The parameter **-RetailSelfServicePackages** is the full path location created in the beginning of this step (**C:/selfservicepackages**).

5. Download the appropriate binary update from LCS to have the Commerce installers. For instructions, see Download updates from Lifecycle Services (LCS).

6. Extract the zip file and copy all self-service installers into the folder **C:/selfservicepackages** defined and created in step 2 in each of the AOS machines. The six self-service installers include:

   - AsyncServerConnectorServiceSetup.exe
   - RealtimeServiceAX63Setup.exe
   - HardwareStationSetup.exe
   - ModernPosSetup.exe
   - ModernPosSetupOffline.exe
   - StoreSystemSetup.exe

NOTE

Cloud environments can synchronize self-service installers through Headquarters from what is available in LCS (Synchronize self-service installers in Dynamics 365 Commerce). On-premises environments cannot utilize this functionality, however, these environments can still download from LCS. The SDK is available in the deployable package zip file. The self-service installers are available from the LCS **Asset library**. You can utilize the upload and download mechanism from within LCS, but the Headquarters synchronization functionality will not work.

7. Navigate to the AD FS machine, then go to the InfrastructureScripts folder. This is the same file directory where the previously run PowerShell script was located (**RetailUpdateDatabase.ps1**). Find the PowerShell script **Create-ADFSServerApplicationForRetail.ps1**.

8. On the AD FS machine that you're currently using, run this script in a new PowerShell window using the

command **.\Create-ADFSServerApplicationForRetail -HostUrl 'https://ax.d365ffo.onprem.contoso.com'**, where the **HostUrl** value can be found in Service Fabric. To find the **HostUrl** value, go to **Service Fabric** > **Application fabric:/AXSF** > **Details** > **Aad_AADValidAudience**.

9. Access the newly generated Server application from the **Application Groups** in AD FS Management.
10. Edit the newly generated Server application and select **Reset the Secret**.

> **NOTE**
>
> It is an important security measure to run this script for each Commerce Scale Unit. This maximizes security and minimizes the workload in case of a security breach.
>
> It is critical to keep this secret safe. This secret should only be copied once and never stored on the system. The Client ID and Secret generated will be used during the Commerce Scale Unit installer, so it is required to be used at a later time. You can always reset the secret again, but it must then be updated on any Commerce Scale Unit that used the previous secret.

11. Go to **Retail and Commerce** > **Headquarters setup** > **Commerce scheduler** > **Connector for Microsoft Dynamics AX**.
12. Select **Edit** on the Action pane.
13. In the **Profile** field, enter the value **Default**. If needed, enter a description in the **Description** field.

> **NOTE**
>
> It is possible for the following fields in steps 12 through 14 to already have values. If this occurs, skip those steps and continue from there. What is important is to have a selectable profile title (default in this case).

14. In the **Web application name** field, enter **RetailCDXRealTimeService**.
15. In the **Protocol** field, select **https**.
16. In the **Common name** field, enter **AXServiceUser@contoso.com**.
17. Select **Save** on the Action Pane.
18. In Headquarters, go to **Retail and Commerce** > **Headquarters setup** > **Parameters** > **Commerce shared parameters**.
19. Select the **Security** tab.
20. Under the sub-heading **Transaction service legacy properties**, select the **Real-time Service profile** field, and then select the newly created **Default** value.
21. Select the **Identity providers** tab.
22. On the **Identity providers** FastTab, select **Add**.
23. In the new **Issuer** row, enter the new Identity provider value **https://sts.windows.net/** in the field.
24. Select **Save** on the Action Pane.
25. Go to **Retail and Commerce** > **Headquarters setup** > **Parameters** > **Commerce parameters**.
26. On the **General** tab, select the **Initialize** link to configure seed data for Commerce functionality.

NOTE

The installers will not download from their relevant pages the first time a download is attempted. This is because the installers have only just been placed into the download location and the associated database values do not yet exist. In Headquarters, when the **Download** functionality is attempted (for example, Commerce Scale Unit or Modern POS), an error will display and then an automated upload functionality will be initiated to allow the installers to be downloaded the second time that the download is attempted. (Wait one minute before attempting to download the installer again).

The Peripheral Simulator (downloaded on the Hardware profile page in headquarters) will not be available until at least one Hardware profile has been created and is functional. After that point has been achieved, the following script can be run.

```
.\RetailUpdateDatabase.ps1 -envName 'LBDenv1' -UpdateRetailHardwareProfileSelfServicePackage
```

27. Follow the installation steps for installing the Commerce Scale Unit. For instructions, see Configure and install Commerce Scale Unit (self-hosted). At multiple locations in this document there will be notes referencing changes to the instructions for an on-premises deployment. It is important to note each of these changes.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Develop and deploy custom models to on-premises environments

2/18/2021 • 3 minutes to read • Edit Online

This topic describes how to develop customizations and extensions, and deploy them to an on-premises environment. On-premises environments are also referred to as local business data (LBD) environments. This topic focuses on the ways that this process differs from the process in a run-time cloud environment.

The process has the following main steps:

1. Deploy your development and build environments.
2. Create a deployable package of your code and customizations.
3. Upload the deployable package to your project in Microsoft Dynamics Lifecycle Services (LCS).
4. Configure and deploy an on-premises runtime environment that includes your deployable package. This environment can be either a sandbox environment or a production environment.

The following sections provide more information about this process.

Development tools and platform

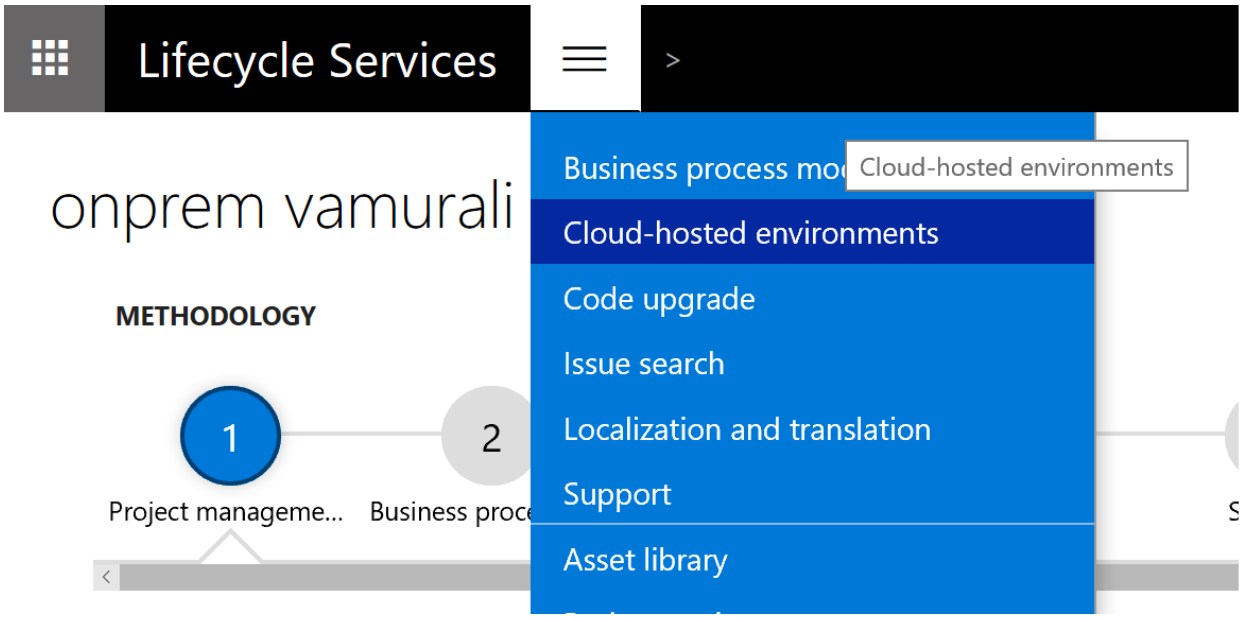
Whether you're developing, extending, or customizing cloud applications or on-premises applications, the development platform, tools, and environments (virtual machines [VMs]) are the same. Your custom code is developed on the same development VMs, regardless of whether your target runtime environments are in a cloud environment or an on-premises environment.

For detailed information about development, see the Develop and customize home page. For information about extensibility and customization, see the Extensibility home page. For information about building, testing, and continuous delivery, see the Continuous delivery home page.

Deploy development and build environments

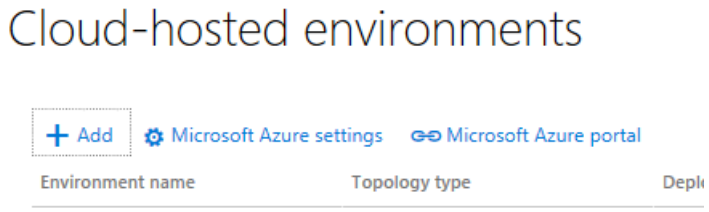
You can use an on-premises LCS project to deploy build and development environments on Microsoft Azure by using your own Azure subscription. Alternatively, you can download a virtual hard disk (VHD) for local development.

To deploy a development or build environment in your Azure subscription, or to download a development VHD, open the **Cloud-hosted environments** page in LCS.



Then follow these steps.

1. Click **Add**.

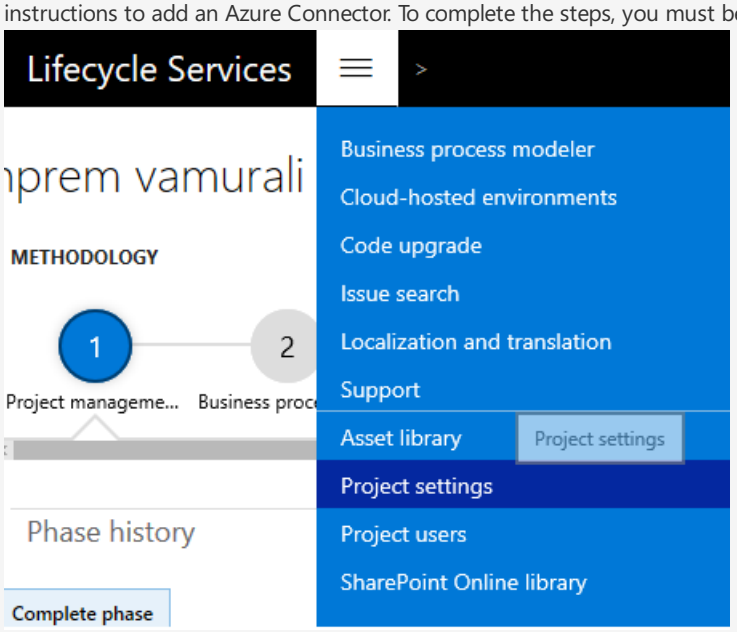


2. Select **Azure** or **Locally**. If you select **Locally**, find and download a development VHD. If you select **Azure**, you're prompted to select one of three topologies: **Build and Test**, **Demo**, or **Development**.
3. Complete the deployment steps, and deploy a VM in your Azure subscription.

For more information about how to configure a local development VHD, see Deploy and access development environments.

NOTE

To deploy environments in your own Azure subscription, you must set up at least one Azure Connector. To set up an Azure Connector, in LCS, open the **Project settings** page, and then click the **Azure connectors** tab. Then follow the instructions to add an Azure Connector. To complete the steps, you must be the tenant administrator of the organization.



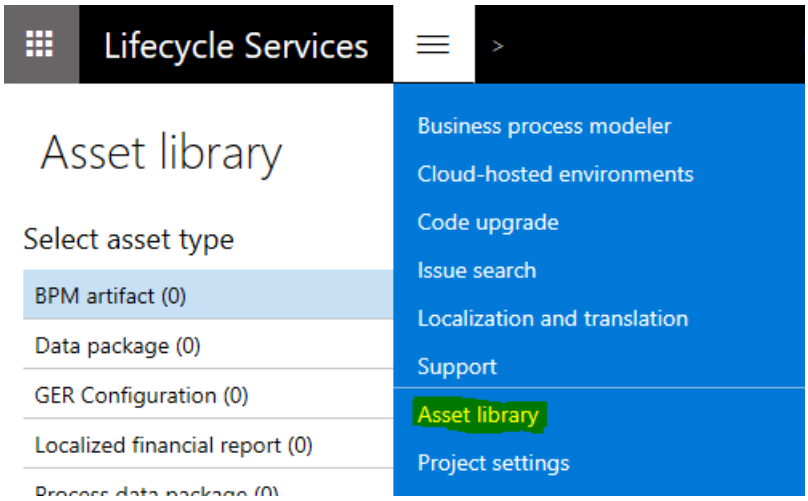
Create and upload a deployable package to the LCS Asset library

When you complete a phase of development, and are ready to deploy your code to a sandbox or production on-premises environment, you must create an application deployable package from your models. This process doesn't differ from the process for cloud environments.

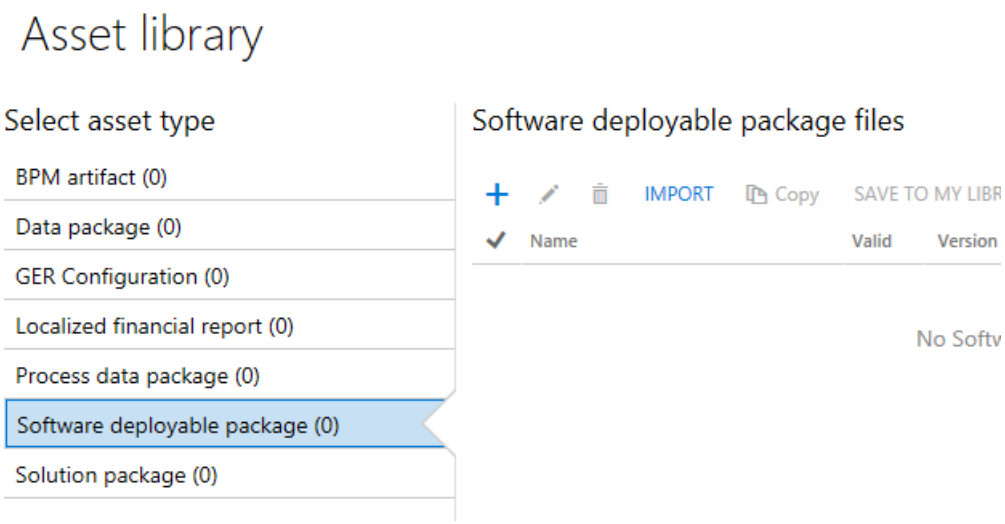
If you're using automated builds (a build environment), the build process creates an application deployable package for you. You can also create an application deployable package from Microsoft Visual Studio in your development environment. For more information on how to create an application deployable package in your development environment, see Create deployable packages of models.

When your deployable package is ready, follow these steps to upload it to your LCS project's Asset library.

1. Open the **Asset library** page.



2. Click the **Software deployable package** tab.

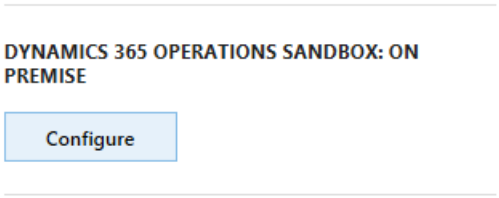


3. Click the plus sign (+) to upload the deployable package.

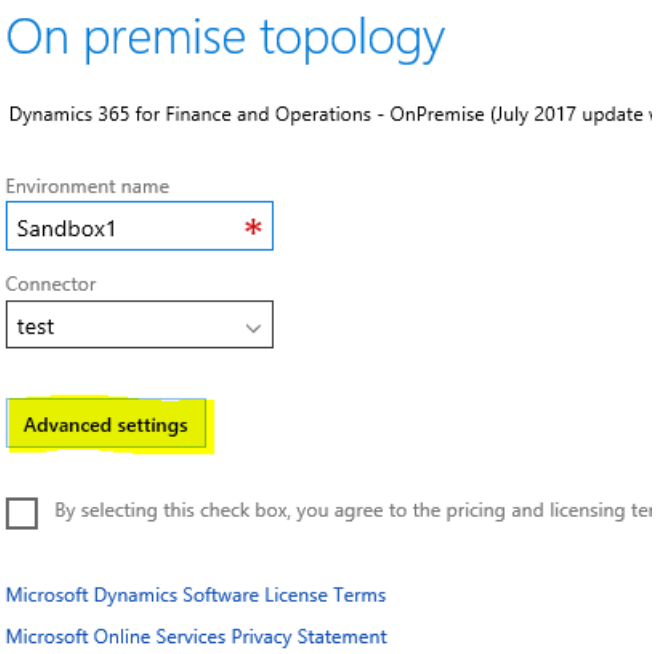
Configure an on-premises runtime environment that uses your code

As of the July 2017 release of Microsoft Dynamics 365 for Finance and Operations (on-premises), you can apply your customizations and extensions only during the deployment of a sandbox or production environment.

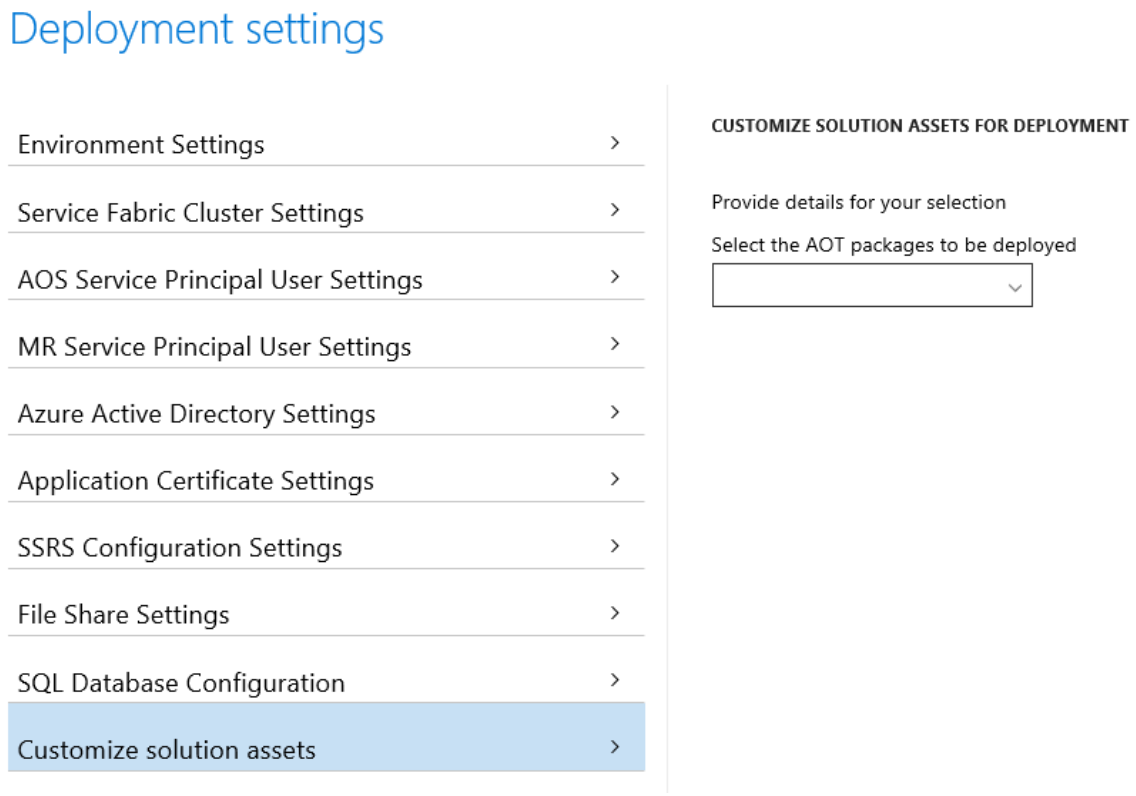
1. In your LCS project, click **Configure** to deploy your environment.



2. In the deployment tool, when you must enter the environment name, click **Advanced settings**.



3. Click the **Customize solution assets** tab.



4. In the **Select the AOT packages to be deployed** field, select the application (AOT) deployable package that contains your customizations. This field lists all the AOT packages in your Asset library.
5. Click **Done** to close the **Deployment settings** page, and then continue with the environment deployment process.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Apply updates to on-premises deployments

2/18/2021 • 8 minutes to read • Edit Online

This topic explains how to apply supported updates to Dynamics 365 Finance + Operations (on-premises). All updates to on-premises environments are done through Microsoft Dynamics Lifecycle Services (LCS).

Search for and download updates

For more information about how to find the updates that you can apply to your on-premises environment, see Issue search in Lifecycle Services (LCS). For information about how to download updates from the tiles in the **Updates** section of the **Environment details** page in LCS, see Download updates from Lifecycle Services (LCS).

> **NOTE**
>
> When you are updating an on-premises environment, always select updates from the update tiles on the **Environment** details page. If you select updates from another location, the updates might not work.

Update an on-premises deployment

You can apply updates to an on-premises environment either during deployment or after the deployment is completed.

While an on-premises environment is being deployed, you can select to deploy a custom package in the **Advanced** settings. For more information about how to apply customizations or application X++ updates, see Develop and deploy custom models to on-premises environments.

To apply updates to an on-premises environment after it has been deployed, in LCS, on the **Environment details** page for the environment, under **Maintain**, select **Apply updates**.

> **NOTE**
>
> You can apply updates after deployment only on environments that have Platform update 12 for Finance and Operations or later. The environment must also have the latest version of the local agent available in LCS. For more information, see Update the local agent. If you're on a platform version that is older than Platform update 12, you can reconfigure an environment that is already deployed to update the customizations or update to the latest platform release. For more information about how to redeploy an environment, seeRedeploy on-premises environments.

Apply application or binary updates through LCS

The following steps can be used to apply X++, All Binary, or Platform binary updates.

> **IMPORTANT**
>
> The application of updates requires downtime for your environment. Therefore, no business transactions can be performed in the environment during the update. When you complete the following steps, verify that the system isn't being used, and that an official downtime notice has been communicated to all system users.

> **IMPORTANT**
> To move to the latest platform, always select the platform update from the **Platform Binary Updates** tile on the **Environment** details page. If you select updates from another location, the updates might not work.

Prerequisites

- Before you begin, complete a full backup of the Management Reporter (MR), Microsoft Dynamics AX, and Microsoft SQL Server Reporting Services (SSRS databases). Although the code is restored through LCS, the database must be manually restored to help guarantee that there is no data loss.
- Update your environment to the latest build of Platform update 12.
- Update the local agent to the latest version. For more information, see Update the local agent.
- Depending on the type of update, complete the following steps to generate a deployable package:
  - **Platform binary updates** – Download or save the update directly to the Asset library in LCS by following the steps in Download updates from Lifecycle Services (LCS).
  - **Application binary updates** – Download or save the update directly to the Asset library in LCS by following the steps in Download updates from Lifecycle Services (LCS).
  - **Application X++ updates** – Download the required hotfix to your development environment, and then follow the steps in Create deployable packages of models.
  - **Customizations** – Follow the steps in Develop and deploy custom models to on-premises environments.

Update a sandbox environment

1. In the LCS Asset library, upload the deployable package that was generated in the "Prerequisites" section of this topic to the**Software deployable packages**tab.
2. In LCS, open the on-premises implementation project, and then open the **Environment details** page of the environment to update.
3. Under **Maintain**, select **Apply updates**. A slider shows the updates that were uploaded to the Asset library. Note that only packages that are marked as **Valid** in the Asset library appear.

If you are on local agent version 2.1.0 and higher, complete the following steps.

1. Select the update, and then click **Prepare**. Clicking on **Prepare** will prepare your on-premises environment for servicing.

   > **NOTE**
   > During preparation, the environment state will be **Deployed** but the Deployment status field will show the progress of Preparation. Steps such formatting the package and downloading the package are executed during preparation. The environment is not directly touched during preparation and hence there is no downtime during the preparation phase. Users can continue to use the system during preparation.

2. After the preparation is complete, you will see **Abort** and **Update Environment** buttons. To start applying the update, click **Update Environment**. If preparation fails, see the "Resolve a failed update application" section later in this topic.
3. In the confirmation message, select **Yes**. The servicing operation has started on this environment. This is the start of the downtime on your environment.
4. The environment state is changed from **Deployed** to **Deploying**.
5. After the update is completed, the environment state is changed back to **Deployed**. If application of the

update fails, the environment state is changed to **Failed**. For information about what to do if package application fails, see the "Resolve a failed update application" section later in this topic.

6. Open the **History** and **Environment details** pages to view the operations that were performed on the environment. You can also view a record of major actions that were performed on the environment, such as deployments, servicing, and rollbacks.

If you are on local agent version lower than 2.1.0, complete the following steps.

1. Select the update, and then click **Apply**.
2. In the confirmation message, select **Yes**. The servicing operation has started on this environment. This is the start of the downtime on your environment.
3. Environment state changes from **Deployed** to **Preparing**.

> **NOTE**
> During preparation, steps such formatting the package and downloading the package are executed during preparation. The environment is not directly touched during preparation and hence there is no downtime during the preparation phase. Users can continue to use the system during preparation. However, we recommend that the downtime starts when the environment enters the Preparing state.

4. After preparation is complete, the environment state is changed from **Preparing** to **Deploying**.
5. After the update is completed, the environment state is changed back to **Deployed**. If application of the update fails, the environment state is changed to **Failed**. For information about what to do if package application fails, see the "Resolve a failed update application" section later in this topic.
6. Open the **History** and **Environment details** pages to view the operations that were performed on the environment. You can also view a record of major actions that were performed on the environment, such as deployments, servicing, and rollbacks.

Update a production environment

Before you update a production environment, you must successfully complete the package application update on a sandbox environment.

1. In the project for the sandbox environment that you applied the package to, open the Asset library, and then, on the **Software deployable packages** tab, select the package, and mark it as a **Release candidate**.
2. On the **Environment details** page, under **Maintain**, select **Apply updates**. In the dialog box, only packages that are marked as a **Release candidate** are shown.
3. Select the Release candidate package to be applied to the Production environment.
4. The rest of the Update flow is the same as that of a sandbox environment. Your update experience will differ based on the version of the local agent running on your environment. We recommend that you always run with the latest version.

Resolve a failed update application

When preparation fails, the environment state is **Deployed**. When the application of an update fails, the environment state is **Failed**. The first step is to determine why there is a failure. The location of the logs varies, depending on the stage where the failure occurred:

- **Preparation stage:** If the operation fails during the **Preparation** stage, the logs are uploaded to LCS. In the log files, select **Download logs** to download the log files. If the package has any merge issues, the error is included in the log file.
- **Deploying stage:** If the operation fails during the **Deploying** stage, the logs are located in the on-premises

environment. You must sign in to the environment, and then access the logs and event viewer.

For more information about how to use the troubleshooting logs, see Troubleshoot on-premises deployments.

After you review the logs and determine the cause of the failure, complete one of the following operations to restore the environment to a healthy state. No actions can be performed on an environment that is in a **Failed** state. The environment must first be restored to a healthy state.

- **Retry failed operation** – If update application fails, select **Retry** to recover from the failed operation.
- **Abort failed operation** – Because there is no change made to the on-premises environment, if the preparation fails, you have the option to cancel the operation. Select **Abort** to cancel the preparation.
- **Roll back the update** – To roll back the update that failed, select **Rollback**. Before you start the rollback, you must restore the database to the last known good state. When you select **Rollback**, the environment is restored to the last known good state. The environment state is then changed to **Preparation**, then to **Deploying**, and then to either **Deployed** or **Failed**.

  > **NOTE**
  >
  > The **Rollback** button doesn't roll back the database. You're responsible for restoring the database to the last known backup that was made before update application. This step is critical to help guarantee that there is no data loss.

- **Refresh the state** – If update application fails during the **Preparation** stage, the failure is on the LCS side, and update application hasn't yet started. Therefore, the on-premises environment is in a good state. To restore the LCS environment state to **Deployed**, on the project dashboard page, select **Refresh**.
- **Delete and redeploy an environment** – If the retry and rollback options don't work, you must delete and redeploy the environment. To delete the environment, on the project dashboard page, select **Delete**. You then see the option to configure the environment.

  > **IMPORTANT**
  >
  > This option should **not** be used on a production environment. However, it can be used on a sandbox deployment to restore the environment to a healthy state.

  > Because this option requires that you do a fresh deployment of the environment, you lose any updates that were previously applied. Any customizations and binary updates must be reapplied to the environment.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Certificate rotation

2/18/2021 • 9 minutes to read • Edit Online

You may need to rotate the certificates used by your Dynamics 365 Finance + Operations (on-premises) environment as they approach their expiration date. In this topic, you will learn how to replace the existing certificates and update the references within the environment to use the new certificates.

WARNING

The certificate rotation process should be initiated well before the certificates expire. This is very important for the Data Encryption certificate, which could cause data loss for encrypted fields. For more information, see After certificate rotation.

Old certificates must remain in place until the certificate rotation process is complete, removing them in advance will cause the rotation process to fail.

Caution

The certificate rotation process should not be carried out on Service Fabric clusters running 7.0.x and 7.1.x.

Upgrade your Service Fabric cluster to 7.2.x or later before attempting certificate rotation.

Preparation steps

1. Rename the original **Infrastructure** folder that you created during the process to Download setup scripts from LCS. Rename the folder to **InfrastructureOld**.
2. Download the latest setup scripts from Download setup scripts from LCS. Unzip the files into a folder that is named **Infrastructure**.
3. Copy **ConfigTemplate.xml** and **ClusterConfig.json** from **InfrastructureOld** to **Infrastructure**.
4. Configure certificates as needed in **ConfigTemplate.xml**. Follow the steps in Configure certificates, specifically these steps.

```
# Create self-signed certs
.\New-SelfSignedCertificates.ps1 -ConfigurationFilePath .\ConfigTemplate.xml
```

Alternatively, if you have or would like to switch to Active Directory Certificate Services (AD CS) certificates, use this information.

```
# Only run the first command if you have not generated the templates yet.
.\New-ADCSCertificates.ps1 -ConfigurationFilePath .\ConfigTemplate.xml -CreateTemplates
.\New-ADCSCertificates.ps1 -ConfigurationFilePath .\ConfigTemplate.xml
```

NOTE

The AD CS scripts need to run on a domain controller, or a Windows Server computer with Remote Server Admin Tools installed. The AD CS functionality is only available with Infrastructure scripts release 2.7.0 and later.

> Self-signed certificates should never be used in production environments. If you're using publicly trusted certificates, manually update the values of those certificates in the ConfigTemplate.xml file.

```
# Export Pfx files into a directory VMs\<VMName>, all the certs will be written to
infrastructure\Certs folder
.\Export-PfxFiles.ps1 -ConfigurationFilePath .\ConfigTemplate.xml
```

5. Continue to Setup VMs. The specific steps that are needed for this process include:

   a. Export the scripts that must be run on each VM.

   ```
   # Export the script files to be executed on each VM into a directory VMs\<VMName>
   .\Export-Scripts.ps1 -ConfigurationFilePath .\ConfigTemplate.xml
   ```

   b. Copy the contents of each infrastructure\VMs folder into the corresponding VM (if remoting scripts are used, they will automatically copy the content to the target VMs), and then run the following scripts, if they exist. Perform these steps as an Administrator.

   ```
   # If remoting, only execute
   # .\Complete-PreReqs-AllVMs.ps1 -ConfigurationFilePath .\ConfigTemplate.xml -
   ForcePushLBDScripts

   .\Import-PfxFiles.ps1
   .\Set-CertificateAcls.ps1
   ```

   c. Run the following script to validate the VM setup.

   ```
   # If remoting, only execute
   # .\Test-D365FOConfiguration-AllVMs.ps1 -ConfigurationFilePath .\ConfigTemplate.xml
   .\Test-D365FOConfiguration.ps1
   ```

6. If axdataenciphermentcert certificates are rotated, you need to regenerate the credentials.json file. For more information, see Encrypt credentials.

7. Run the following PowerShell command to have values that can be used in LCS later. For more information, see Deploy your on-premises environment from LCS.

```
.\Get-DeploymentSettings.ps1 -ConfigurationFilePath .\ConfigTemplate.xml
```

Activate new certificates within Service Fabric cluster

Service Fabric with certificates that aren't expired

1. Open the **Clusterconfig.json** file for editing, and find the following section. If a secondary thumbprint is defined, go to Clean up old Service Fabric certificates before you go any further.

```
"security": {
    "metadata":  "The Credential type X509 indicates this cluster is secured using X509 Certificates.
    The thumbprint format is - d5 ec 42 3b 79 cb e5 07 fd 83 59 3c 56 b9 d5 31 24 25 42 64.",
    "ClusterCredentialType":  "X509",
    "ServerCredentialType":  "X509",
    "CertificateInformation":  {
        "ClusterCertificate":  {
                                "X509StoreName":  "My",
                                 "Thumbprint": "*Old server thumbprint(Star/SF)*"
                            },
        "ServerCertificate":   {
                                 "X509StoreName":  "My",
          "Thumbprint": "*Old server thumbprint(Star/SF)*"
                            },
        "ClientCertificateThumbprints":  [
                                {
                                     "CertificateThumbprint": "*Old client thumbprint*",
                                     "IsAdmin":  true
                                }
                                      ]
                            }
           },
```

2. Replace that section in the file with following section.

```
"security":  {
    "metadata":  "The Credential type X509 indicates this cluster is secured using X509 Certificates.
    The thumbprint format is - d5 ec 42 3b 79 cb e5 07 fd 83 59 3c 56 b9 d5 31 24 25 42 64.",
    "ClusterCredentialType":  "X509",
    "ServerCredentialType":  "X509",
    "CertificateInformation":  {
        "ClusterCertificate":  {
                                "X509StoreName":  "My",
                                 "Thumbprint": "*New server thumbprint(Star/SF)*",
                                 "ThumbprintSecondary": "Old server thumbprint(Star/SF)"
                            },
        "ServerCertificate":   {
                                 "X509StoreName":  "My",
          "Thumbprint": "*New server thumbprint(Star/SF)*",
                                 "ThumbprintSecondary": "Old server thumbprint(Star/SF)"
                            },
        "ClientCertificateThumbprints":  [
                                {
                                     "CertificateThumbprint": "*Old client thumbprint*",
                                     "IsAdmin":  false
                                },
                                {
                                     "CertificateThumbprint": "*New client thumbprint*",
                                     "IsAdmin":  true
                                }
                                      ]
                            }
           },
```

3. Edit the new and old thumbprint values.

4. Change clusterConfigurationVersion to the new version, for example 2.0.0.

```
{
"name": "Dynamics365Operations",
"clusterConfigurationVersion": "2.0.0",
"apiVersion": "10-2017",
```

5. Save the new ClusterConfig.json file.

6. Run the following PowerShell command.

```
# Connect to the Service Fabric cluster
Connect-ServiceFabricCluster

# Get path of ClusterConfig.json for following command
# Note that after running the following command, you need to manually cancel using the red button
(Stop Operation) in Windows PowerShell ISE or Ctrl+C in Windows PowerShell, otherwise you will
receive the following notification, "Start-ServiceFabricClusterConfigurationUpgrade : Operation timed
out.". Be aware that the upgrade will proceed.
Start-ServiceFabricClusterConfigurationUpgrade -ClusterConfigPath ClusterConfig.json

# If you are using a single Microsoft SQL Server Reporting Services node, use
UpgradeReplicaSetCheckTimeout to skip PreUpgradeSafetyCheck check, otherwise it will timeout
Update-ServiceFabricClusterUpgrade -UpgradeReplicaSetCheckTimeoutSec 30

# To monitor the status of the upgrade, run the following and note UpgradeState and
UpgradeReplicaSetCheckTimeout
Get-ServiceFabricClusterUpgrade

# While monitoring the status of the upgrade, if UpgradeReplicaSetCheckTimeout was reset to the
default (example 49710.06:28:15), run the following command again
Update-ServiceFabricClusterUpgrade -UpgradeReplicaSetCheckTimeoutSec 30

# When UpgradeState shows RollingForwardCompleted, the upgrade is finished
```

Service Fabric with or without expired certificates (cluster not accessible)

Continue this process following Troubleshoot on-premises deployments.

Update the LocalAgent certificate

You must reinstall the LocalAgent if:

- You changed the service fabric cluster/server certificate.
- You changed the service fabric client certificate.
- You changed the LocalAgent certificate.

1. Update your current localagent-config.json by replacing the **serverCertThumprint** and **clientCertThumbprint** values with the new thumbprints.

```
{
"serviceFabric": {
    "connectionEndpoint": "192.168.8.22:19000",
    "clusterId": "Orch",
    "certificateSettings": {
        "serverCertThumbprint": "New server thumbprint(Star/SF)",
        "clientCertThumbprint": "New client thumbprint"
    }
},
```

2. Run the following PowerShell command on one of the Orchestrator nodes.

```
.\LocalAgentCLI.exe Cleanup <path of localagent-config.json>
```

3. Run the following PowerShell command and note the new LocalAgent thumbprint.

```
.\Get-AgentConfiguration.ps1 -ConfigurationFilePath .\ConfigTemplate.xml
```

4. Follow the steps in Configure LCS connectivity for the tenant.

> **NOTE**
> If you receive the error **Update to existing credential with KeyId '<key>' is not allowed**, follow the instructions in Error: "Updates to existing credential with KeyId '' is not allowed".

5. Continue with Configure a connector and install an on-premises local agent, specifically the following changes:

   - Client certificate thumbprint
   - Server certificate thumbprint
   - Tenant service principle certificate thumbprint

Update your current deployment configuration

Because you've updated your certificates, the configuration file that is present in your environment is outdated and must be manually updated. Otherwise, the cleanup job will probably fail. (This manual update must be done just this one time.)

1. Open your configuration file. You can find the location of this file by running the following command.

```
select Location from DeploymentInstanceArtifact where AssetId='config.json' and DeploymentInstanceId
= 'LCSENVIRONMENTID'
```

> **NOTE**
> Replace **LCSENVIRONMENTID** with the ID of your environment. You can obtain this ID from the page for your environment in LCS.

   The beginning of the file should resemble the following example.

```
{
"serviceFabric": {
    "connectionEndpoint": "192.168.8.22:19000",
    "clusterId": "Orch",
    "certificateSettings": {
        "serverCertThumbprint": "Old server thumbprint(Star/SF)",
        "clientCertThumbprint": "Old client thumbprint"
    }
},
```

2. Replace the **serverCertThumprint** and **clientCertThumbprint** values with the new thumbprints.

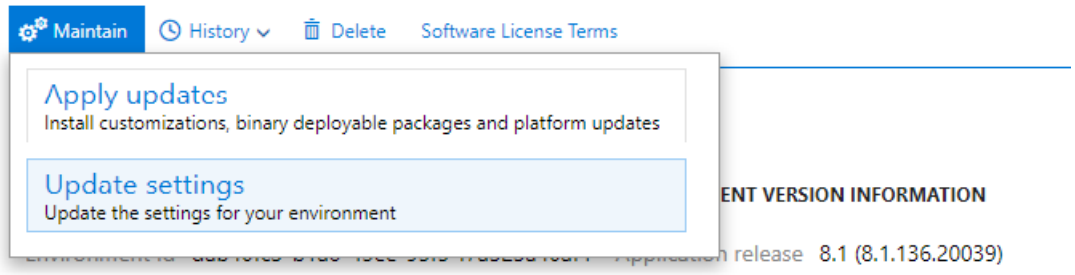
```
{
"serviceFabric": {
    "connectionEndpoint": "192.168.8.22:19000",
    "clusterId": "Orch",
    "certificateSettings": {
        "serverCertThumbprint": "New server thumbprint(Star/SF)",
        "clientCertThumbprint": "New client thumbprint"
    }
},
```

3. Save and close the file. Remember to close any programs that are accessing this network location. Otherwise, the cleanup process might fail.

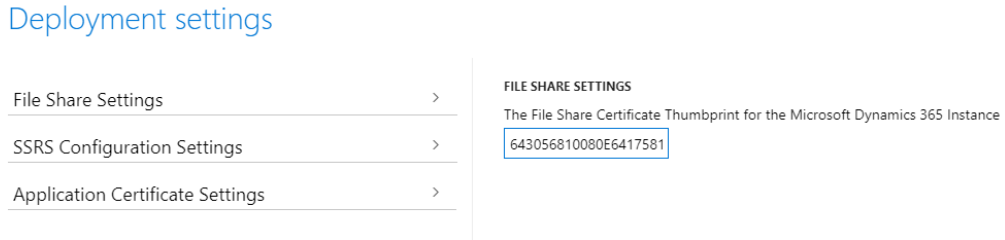
Update deployment settings in LCS

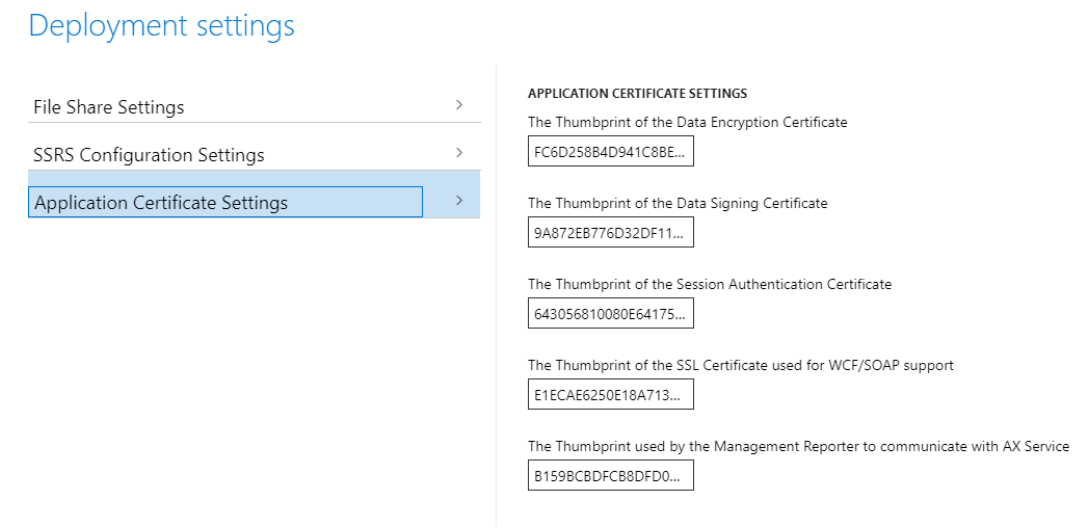
> **NOTE**
>
> Note that the Client, Data Signing, and Encipherment certificates will only be replaced. You will also need to recreate the Credentials.json file, as described in Encrypt credentials.
>
> Before you continue, you need to make a backup of the local Dynamics database.

1. In LCS, select the "Full Details" link for the environment where you want to change the certificates.
2. Select **Maintain** and then select **Update Settings**.

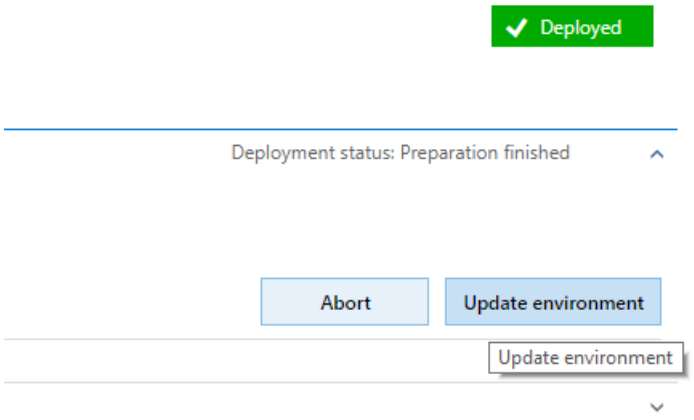


3. Change the thumbprints to the new thumbprints that you previously configured. You can find them in the ConfigTemplate.xml file in the InfrastructureScripts folder.



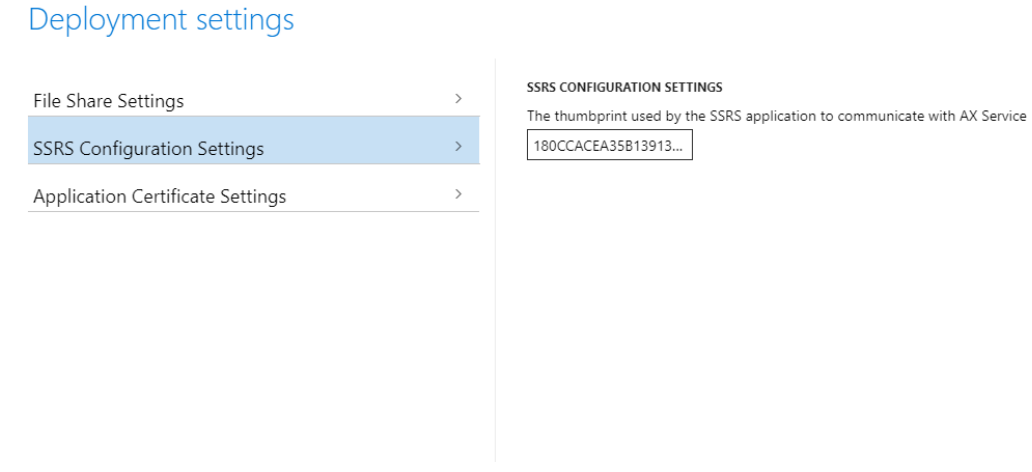


4. Select **Prepare**.
5. After downloading and preparation is complete, the **Update environment** button will display.



6. Select **Update environment** to start updating your environment.
7. During the update, the environment will be unavailable.
8. After the environment is successfully updated with the new certificates, you can view the new thumbprints in Service Fabric Cluster Explorer. The names of the thumbprints in Service Fabric Explorer might differ from the names in LCS. However, the values should be the same.

   Here is an example of how the name of the same thumbprint might differ.



Update other certificates as needed

1. Always check if the SQL server certificate has expired. For more information, see Set up SQL Server.
2. Check to be sure that the Active Directory Federation Service (ADFS) certificate has not expired.

Clean up old Service Fabric certificates

This procedure should be completed either after a successful certificate rotation or before the next certificate rotation.

1. Remove the old/secondary thumbprints from the cluster configuration. After you've removed them, the appropriate section should resemble the following example.

```
"security": {
    "metadata":  "The Credential type X509 indicates this is cluster is secured using X509
Certificates.
    The thumbprint format is - d5 ec 42 3b 79 cb e5 07 fd 83 59 3c 56 b9 d5 31 24 25 42 64.",
    "ClusterCredentialType":  "X509",
    "ServerCredentialType":  "X509",
    "CertificateInformation":  {
        "ClusterCertificate":  {
                                    "X509StoreName":  "My",
                                     "Thumbprint": "server thumbprint(Star/SF)"
                                },
        "ServerCertificate":   {
                                     "X509StoreName":  "My",
          "Thumbprint": "server thumbprint(Star/SF)"
                                },
        "ClientCertificateThumbprints":  [
                                    {
                                         "CertificateThumbprint": "client thumbprint",
                                         "IsAdmin":  true
                                    }
                                          ]
                                }
            },
```

2. Follow steps 4 through 6 in the Service Fabric with certificates that are not expired section earlier in this topic.

After certificate rotation

Data encryption certificate

This certificate is used to encrypt data stored in the database. By default there are certain fields that are encrypted with this certificate, you can check those fields in Document the values of encrypted fields. However, our API can be used to encrypt other fields that customers deem should be encrypted.

In Platform update 33 and later, the batch job that is named "Encrypted data rotation system job" will use the newly rotated certificate to re-encrypt data. This batch job crawls through your data to re-encrypt all the encrypted data by using the new certificate. It will run for two hours per day until all of the data has been re-encrypted. In order to enable the batch job, a flight and a configuration key need to be enabled. Execute the following commands against your business database (for example, AXDB).

```
IF (EXISTS(SELECT * FROM SYSFLIGHTING WHERE [FLIGHTNAME] = 'EnableEncryptedDataCrawlerRotationTask'))
  UPDATE SYSFLIGHTING SET [ENABLED] = 1 WHERE [FLIGHTNAME] = 'EnableEncryptedDataCrawlerRotationTask'
ELSE
  INSERT INTO SYSFLIGHTING ([FLIGHTNAME],[ENABLED],[FLIGHTSERVICEID]) VALUES
('EnableEncryptedDataCrawlerRotationTask', 1, 0)

IF (EXISTS(SELECT * FROM SECURITYCONFIG WHERE [KEY_] = 'EnableEncryptedDataRotation'))
  UPDATE SECURITYCONFIG SET [VALUE] = 'True' WHERE [KEY_] = 'EnableEncryptedDataRotation'
ELSE
  INSERT INTO SECURITYCONFIG ([KEY_], [VALUE]) VALUES ('EnableEncryptedDataRotation', 'True')
```

After the above commands have been executed, restart your AOS nodes from Service Fabric Explorer. The AOS will detect the new configuration and will schedule the batch job to run during off hours. After the batch job has been created, the schedule can be modified from the user interface.

WARNING

Make sure that the old Data Encryption certificate is not removed before all encrypted data has been re-encrypted and it has not expired. Otherwise, this could lead to data loss.

Client internet connectivity

2/18/2021 • 2 minutes to read • Edit Online

The configuration of the local network for a deployment of Dynamics 365 Finance + Operations (on-premises) can affect the features that are available in the web client. In particular, if the network configuration does not allow a client machine to access the internet, several degradations in the web client will occur. These include:

- The Office app launcher and Dynamics 365 areas in the navigation bar will no longer be clickable.
- The Help pane will not be accessible.
- The Ideas portal will not be accessible from the web client.
- Users will see their initials instead of a user image.
- Skype integration will not be available.
- The favorite icon shown in the browser tab will be the browser's default favorite icon instead of the application icon.
- The Open in Excel options are hidden because the Excel Add-in will not run.

In addition to platform features that may not be accessible when the client can't access the internet, there may also be application features that rely on an internet connection that developers will need to hide or turn off. To facilitate this, developers can use the **clientHasRestrictedInternet()** method that has been added to the **Session** class. This method will return true if the client does not have access to the internet.

Client internet connectivity options

Client internet connectivity options allow an administrator to manually turn off the external connections that the client makes even when internet connectivity is available. These can be used for troubleshooting issues or to see what the client will look like when internet connectivity is not available. These client internet connectivity options are available out-of-the-box starting in Platform update 16 but are also available in Platform update 15 (via KB 4091764) and Platform update 12 (via KB 4091763).

The client internet connectivity options can be found on the **System administration** > **Setup** > **Client performance options** page.

- **Internet connectivity enabled** - Allows an administrator to turn off all external connections that the web client would otherwise make.
- **Skype presence enabled** - Allows an administrator to turn off external connections to Skype that the web client would otherwise make.

Why does the client connect to the Skype for Business API when it first loads?

When the client loads, it performs a quick call (ping) to the Skype for Business API to check if an internet connection is available. If it isn't available then the client functions in a disconnected fashion. An environment doesn't need to have Skype for Business visible/enabled for this check to be made.

Configure Batch-only and Interactive-only AOS nodes in on-premises deployments

2/18/2021 • 3 minutes to read • Edit Online

IMPORTANT

This feature is supported starting in application update 10.0.12, Platform update 36.

This topic explains how to configure your environment so that you can deploy batch-only and interactive-only Application Object Server (AOS) nodes.

To make this feature available, Microsoft has introduced two new Microsoft Azure Service Fabric node types. For batch-only AOS nodes, the new node type is **BatchOnlyAOSNodeType**. For interactive-only AOS nodes, the new node type is **InteractiveOnlyAOSNodeType**.

NOTE

The traditional deployment option, where an AOS node is interactive and is running batch jobs, is still supported and isn't affected by these changes.

Sizing

For sandbox environments, we recommend that you have at least two nodes of each type.

For production environments, there should be at least three nodes of each type.

New deployments

1. When you're describing your configuration as explained in Set up and deploy on-premises environments, edit the **configtemplate.xml** file to add the new node types. When you've finished, the new **NodeType** sections should resemble the following example.

```
<NodeType name="InteractiveOnlyAOSNodeType" primary="false" namePrefix="AOS" purpose="AOS">
    <VMList>
        <VM name="LBDE05FS1AOS01" ipAddress="192.168.5.51" faultDomain="fd:/fd0" updateDomain="ud0"
/>
        <VM name="LBDE05FS1AOS02" ipAddress="192.168.5.52" faultDomain="fd:/fd1" updateDomain="ud1"
/>
        <VM name="LBDE05FS1AOS03" ipAddress="192.168.5.53" faultDomain="fd:/fd0" updateDomain="ud2"
/>
    </VMList>
</NodeType>
<NodeType name="BatchOnlyAOSNodeType" primary="false" namePrefix="AOS" purpose="AOS">
    <VMList>
        <VM name="LBDE05FS1AOS04" ipAddress="192.168.5.54" faultDomain="fd:/fd0" updateDomain="ud0"
/>
        <VM name="LBDE05FS1AOS05" ipAddress="192.168.5.55" faultDomain="fd:/fd1" updateDomain="ud1"
/>
        <VM name="LBDE05FS1AOS06" ipAddress="192.168.5.56" faultDomain="fd:/fd0" updateDomain="ud2"
/>
    </VMList>
</NodeType>
```

2. Follow the remaining instructions in Set up and deploy on-premises environments in the usual way.

Existing deployments

1. Follow the instructions in Remove an AOS node up through the point where you save the configuration file.

> **IMPORTANT**
>
> You must use option 1, "Use a configuration file (preferred option)." Do **not** use option 2.

2. Continue to edit the **ClusterConfig.json** file to add the new node types to the **NodeTypes** section. When you've finished, the **NodeTypes** section should resemble the following example.

```
"NodeTypes": [
    {
        "Name": "AOSNodeType",
        "PlacementProperties": {
            "IsAosEnabled": "True"
        },
        "ClientConnectionEndpointPort": 19000,
        "HttpGatewayEndpointPort": 19080,
        "ReverseProxyEndpointPort": 19081,
        "LeaseDriverEndpointPort": 19002,
        "ClusterConnectionEndpointPort": 19001,
        "ServiceConnectionEndpointPort": 19003,
        "ApplicationPorts": {
            "StartPort": 20001,
            "EndPort": 20031
        },
        "IsPrimary": false
    },
    {
        "Name": "BatchOnlyAOSNodeType",
        "PlacementProperties": {
            "IsAosEnabled": "True"
        },
        "ClientConnectionEndpointPort": 19000,
        "HttpGatewayEndpointPort": 19080,
        "ReverseProxyEndpointPort": 19081,
        "LeaseDriverEndpointPort": 19002,
        "ClusterConnectionEndpointPort": 19001,
        "ServiceConnectionEndpointPort": 19003,
        "ApplicationPorts": {
            "StartPort": 20001,
            "EndPort": 20031
        },
        "IsPrimary": false
    },
    {
        "Name": "InteractiveOnlyAOSNodeType",
        "PlacementProperties": {
            "IsAosEnabled": "True"
        },
        "ClientConnectionEndpointPort": 19000,
        "HttpGatewayEndpointPort": 19080,
        "ReverseProxyEndpointPort": 19081,
        "LeaseDriverEndpointPort": 19002,
        "ClusterConnectionEndpointPort": 19001,
        "ServiceConnectionEndpointPort": 19003,
        "ApplicationPorts": {
            "StartPort": 20001,
            "EndPort": 20031
        },
        "IsPrimary": false
    },
    ...
]
```

3. Continue to follow the instructions in Remove an AOS node from the point where you stopped until you've finished removing the nodes from your cluster.

4. On the machines that you've removed from the Service Fabric cluster, follow these steps:

    a. Delete the contents of your Service Fabric data root (**C:\ProgramData\SF**) and your Service Fabric log root (**C:\ProgramData\SF\Log**).

    b. Copy or download the standalone package for Service Fabric for Windows Server to the virtual machine (VM) or machine.

c. Unzip the package (**C:\Temp\ServiceFabricStandalone**).

d. Open Windows PowerShell as an admin.

e. Run the following commands.

> **IMPORTANT**
> Replace parameters as appropriate for your machine configuration.

```
cd C:\Temp\ServiceFabricStandalone
.\AddNode.ps1 -NodeName AOS_12 -NodeType BatchOnlyAOSNodeType -NodeIpAddressOrFQDN
192.168.5.12 -ExistingClientConnectionEndpoint 192.168.5.21:19000 -UpgradeDomain ud0 -
FaultDomain fd:/fd0 -X509Credential -ServerCertThumbprint
1A1A1A1A1A1A1A1A1A1A1A1A1A1A1A1A1A1A1A1A -AcceptEULA -StoreLocation LocalMachine -StoreName MY
-FindValueThumbprint 1A1A1A1A1A1A1A1A1A1A1A1A1A1A1A1A1A1A1A1A
```

> **NOTE**
> When the node first reappears in Service Fabric Explorer, it might show the old node type. However, it should show the updated node type after a few minutes.

5. Query the Service Fabric cluster again for the **ClusterConfig.json** file.

```
Get-ServiceFabricClusterConfiguration -UseApiVersion -ApiVersion 10-2017 > C:\Temp\ClusterConfig.json
```

6. In the **fabricSettings** section, remove the **NodesToBeRemoved** parameter. When you've finished, the **fabricSettings** section should resemble the following example.

```
"fabricSettings": [
    {
        "name": "Setup",
        "parameters": [
            {
                "name": "FabricDataRoot",
                "value": "C:\\\\ProgramData\\\\SF"
            },
            {
                "name": "FabricLogRoot",
                "value": "C:\\\\ProgramData\\\\SF\\\\Log"
            }
        ]
    }
]
```

7. Remove the following lines from the **Security** section.

```
"WindowsIdentities": {
    "\$id": "3"
},
```

> **NOTE**
>
> If you don't remove these lines, you will receive the following error message later:
>
> *ValidationException: Authentication type can't be changed from unsecured to Windows.*

8. Increment the version number of the configuration file. Make this change at the lowest increment. In the following example, the version number went from **1.0.1** to **1.0.2**.

```
"ClusterConfigurationVersion": "1.0.2"
```

9. Save the configuration file.
10. Open Windows PowerShell as an admin, and run the following commands.

```
Connect-ServiceFabricCluster
Start-ServiceFabricClusterConfigurationUpgrade -ClusterConfigPath C:\Temp\ClusterConfig.json
Update-ServiceFabricClusterUpgrade -UpgradeReplicaSetCheckTimeoutSec 30
```

11. To monitor the upgrade, you can run the following command.

```
Get-ServiceFabricClusterUpgrade
```

12. Once the upgrade is finished from LCS use the Update Settings button without changing any settings to trigger an environment refresh. Alternatively you could also apply the latest hotfix.

> **IMPORTANT**
>
> This step causes downtime so be sure that your environment can be unavailable for some time.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Configure proxies for on-premises environments

2/18/2021 • 2 minutes to read • Edit Online

You may want to secure the Dynamics 365 Finance + Operations (on-premises) environment behind a proxy. Proxy is a server that hides the actual servers that are serving traffic from the clients. The proxy server accepts requests from the clients on behalf of the environment and forwards the traffic to it. The clients are not aware of the actual servers that compose the environment. This adds another measure of security and enables load balancing.

Configure the proxy

Perform the following steps in **each** node of type **OrchestratorType** in the Microsoft Azure Service Fabric cluster.

1. Use remote access to connect to the Orchestrator virtual machine (VM).
2. Execute the following PowerShell script to retrieve the path of the `machine.config` file.

```
[System.Runtime.InteropServices.RuntimeEnvironment]::SystemConfigurationFile
```

3. Edit the `machine.config` file to add the following code example.

```
<system.net>
 <defaultProxy enabled="true" >
  <proxy <<<SET YOUR PROXY SETTINGS>> />
    </defaultProxy>
</system.net>
```

4. Save the file.
5. Restart the virtual machine.

The above procedure must be performed for all Orchestrator node VMs.

Safe list URLs

The LocalAgent needs to communicate with Azure resources. As a result, the following URLs should be added to a safe list on the proxy or firewalls so that all **OrchestratorType** nodes can access them:

```
- lcsapi.lcs.dynamics.com
- login.windows.net
- uswelcs1lcm.queue.core.windows.net
- www.office.com
- login.microsoftonline.com
- dc.services.visualstudio.com
- uswelcs1lcm.blob.core.windows.net
```

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

On-premises diagnostics

2/18/2021 • 12 minutes to read • Edit Online

The Microsoft Dynamics 365 team monitors the health and performance of the Azure Services that provide functionality for our cloud-based customers by using state-of-the-art Azure diagnostic tools. For customers who have implemented Finance + Operations (on-premises) and would like to have the ability to monitor the health and performance of their on-premises solution, there are several third-party offerings available.

This topic describes the setup and configuration of Elastic Stack, a third-party product, and one of many choices that can provide diagnostic monitoring of your on-premises solution.

When you consider a diagnostic solution, consider the following fundamentals of your implementation:

- Your diagnostic system should be able to collect and store 30 days' worth of diagnostic information.
- Your diagnostic repository should be set up in a central location that is sharable among many client computers.
- Create structured diagnostics events, including event type, classification, and data.
- Events stored in raw text (deserialized) can be easily queried and searched.
- Avoid storing sensitive or personal data in events.

> **NOTE**
>
> By default, communication in an Elastic Stack cluster is not sent over HTTPS. Don't set up the Elastic Stack unless you've considered the risks, and prepared or implemented mitigations for those risks. The paid version of X-Pack can be used to encrypt communication in the Elastic Stack. For setup information, see Setting up TLS on a cluster. There is also an open source Elasticsearch plug-in. Although Microsoft hasn't tested this plug-in, according to the documentation, it can enable HTTPS. Microsoft recommends that you always utilize encrypted communication using HTTPs, VPN, or another secure, encrypted protocol. Many industry certifications and laws require the use of encrypted transmission if your content includes end user, customer, personal, or sensitive data.

Diagnostic data guidelines

To diagnose the deployment and execution of Finance + Operations (on-premises), you must have access to diagnostic data. For a cloud deployment, Microsoft stores and monitors the diagnostic data from services to help keep the environment healthy. For an on-premises deployment, the customer is responsible for this task.

You can select the diagnostic data store and query tool that you prefer to use. However, at a minimum, the tool should perform the following tasks:

- The store should be able to store 30 days' worth of diagnostic data.
- The events should be stored in a centralized location, so that support engineers don't have to switch between multiple machines to find events that are relevant to an issue.
- The events should be discoverable based on event type and event data.
- The event data (in XML format) should be deserialized so that the event data can be queried on and traversed.

Elastic Stack example

To meet the diagnostic data guidelines that are listed in the previous section, Microsoft tested the Elastic Stack setup. This setup includes the following components:

- **Elasticsearch** – For storage, event indexing, and event querying. For more information about Elasticsearch, see the Elastic website.
- **Logstash** – For load distribution and event data transformation.
- **Winlogbeat** – For diagnostic data collection.
- **Kibana** – An interface for querying the data that is stored in Elasticsearch.

> **NOTE**
>
> By default, communication in an Elastic Stack cluster is **not** sent over HTTPS. Don't set up the Elastic Stack unless you've considered the risks, and prepared or implemented mitigations for those risks. The paid version of X-Pack can be used to encrypt communication in the Elastic Stack. For setup information, see Setting up TLS on a cluster. There is also an open source Elasticsearch plug-in. Although Microsoft hasn't tested this plug-in, according to the documentation, it can enable HTTPS.

If you deploy the Elastic Stack, your experience might vary if you follow the steps that are described in this topic. For its tests, Microsoft used version 6.2.3 of the Elastic Stack components and Microsoft Dynamics 365 for Finance and Operations 7.3 with platform update 12.

This topic describes how Microsoft handled the setup and configuration steps that are required for the Elastic Stack to work for an on-premises deployment. For guidance that isn't related to Finance + Operations (on-premises), see the documentation on Elastic.co.

Install and configure the Elastic Stack

All hosted components of the Elastic Stack, except Winlogbeat, run on Java. For the test scenario, Microsoft first downloaded and installed the latest version of Java Runtime Environment (JRE) 8 (64-bit) on each node that will run Elasticsearch, Logstash, or Kibana (that is, all the Orchestrator nodes). You can get Java 8 from https://www.oracle.com/technetwork/java/javase/downloads/index.html.

As of June 2018, the Elastic Stack runs on Java 8. Any attempt to run it on a newer version of Java might not work.

> **NOTE**
>
> The whole Elastic Stack, except Winlogbeat, can be hosted on Linux. For its tests, Microsoft hosted the stack on Microsoft Windows Server 2016 virtual machines (VMs).

Remember to open ports in the firewall for the various components on each node.

If you get stuck during setup, Elastic.co has extensive and well-written documentation about the installation and configuration of the Elastic Stack. For help with specific types of errors, web searches yield reliable results from both the Elastic.co forum and StackOverflow.

Component matrix

For its tests, Microsoft used the following setup for a small to medium-sized deployment.

NODE	ELASTICSEARCH	LOGSTASH	KIBANA	WINLOGBEAT
Orchestrator #1	X			X
Orchestrator #2	X	X		X
Orchestrator #3		X	X	X

NODE	ELASTICSEARCH	LOGSTASH	KIBANA	WINLOGBEAT
AOS #1...*n*				X

> **IMPORTANT**
> For testing purposes, Microsoft used the Orchestrator machines for the ELK installation. Because it can take up critical resources from the Orchestration services, don't use the Orchestrator machines for ELK installations on production environments or critical Sandbox Environments. Instead, use separate machines to host the ELK services.

Elasticsearch

The installation of Elasticsearch is fairly straightforward. For its tests, Microsoft downloaded the Microsoft Windows Installer (MSI) file onto the Orchestrator #1 and Orchestrator #2 nodes. Most of the default settings in the installer can be left as is. This section describes the settings that Microsoft changed.

To facilitate Elasticsearch to start running again if the operating system (OS) is restarted, Microsoft installed it as a service on Windows. The installer can be used to set up the service.

On the **Configuration** page of the installer, Microsoft used the same cluster name when it installed each Elasticsearch node in the cluster.

Microsoft set every Elasticsearch node to perform all three roles: Data, Master, and Ingest.

Depending on the amount that you expect Kibana and Elasticsearch to be used, consider increasing the memory usage. You can change this setting later by modifying the -Xm options in the C:\ProgramData\Elastic\Elasticsearch\config\jvm.options file and restarting Elasticsearch.

Depending on the number of Elasticsearch nodes that you set up, you can set the Discovery minimum master nodes appropriately. If you aren't sure, you can keep the master nodes empty. For more information about discovery and nodes, see Node.

For discoverability, in **Network settings**, Microsoft set the **Network host** value of each node to that node's IP address and added the IP addresses of all Elasticsearch nodes to the **Unicast Hosts** list for each node. For example, for Orchestrator #1, which has the IP address 10.0.0.12, Microsoft set the **Network host** value to **10.0.0.12** and added the following IP addresses to the **Unicast Hosts** list: 10.0.0.12 and 10.0.0.13, where 10.0.0.13 is Orchestrator #2.

If you're installing Elasticsearch version 6.3 or higher, you can disregard this paragraph. You can install X-Pack either now or later. For more information about setup and whether you should install X-Pack, see the "X-Pack" section of this topic. For now, unless you know what X-Pack is for, don't install it.

> **IMPORTANT**
> Open the HTTP port (by default, port 9200) and the node communication port (by default, port 9300) in your firewall.

To verify that the installation was successful, start a browser, and open the application address. You should see some JavaScript Object Notation (JSON) output.

Logstash

In its test setup, Microsoft found that some events from Winlogbeat required adjustments. Logstash provides that functionality.

Microsoft downloaded Logstash to C:\ELK\Logstash on the Orchestrator #2 and Orchestrator #3 nodes.

To help ensure that Logstash runs on startup, we used the Non-Sucking Service Manager (NSSM) to set up a

service for the Logstash batch script.

1. Copy nssm.exe to the Logstash bin folder (for example, C:\ELK\Logstash\6.2.4\bin\).
2. Open Windows PowerShell from the bin folder, and run the following command.

   ```
   .\nssm.exe install Logstash
   ```

3. On the **Application** tab, set the following fields, and then click **Save**:

   - **Path:** C:\ELK\Logstash\6.2.4\bin\logstash.bat
   - **Startup directory:** C:\ELK\Logstash\6.2.4
   - **Arguments:** -f C:\ELK\Logstash\config\logstash-dyn365finops.conf

   (There are more settings that you can set. However, these settings suffice for now.)

4. Run the following command.

   ```
   .\nssm.exe start Logstash
   ```

In the tests that Microsoft performed, NSSM had trouble restarting the installed services. Because NSSM wasn't 100-percent reliable for Logstash and Kibana, the service was treated as an OS startup service and little else.

Microsoft created a configuration file for Logstash and put it in C:\ELK\Logstash\6.2.4\config. This file performs useful transformations on diagnostics.

To make the configuration work for your setup, you must change the **hosts** fields in the **output** section so that they point to the Elasticsearch nodes in your cluster. For example, change **hosts** to **["ORCH1:9200", "ORCH2:9200"]**.

The configuration was tested by using the Winlogbeat configuration from the next section.

Remember to open the Winlogbeat port (by default, port 5044) in your firewall on the machine that is hosting Logstash, so that Beats can send data to Logstash.

Winlogbeat

Microsoft downloaded Winlogbeat to each Application Object Server (AOS) and Orchestrator node at C:\ELK\Winlogbeat, and configured the winlogbeat.yml file.

To make the configuration work for your set up, you must change the **output.logstash.hosts** fields so that they point to all your Logstash nodes. Winlogbeat handles the load balancing.

When Winlogbeat runs on an Orchestrator node, the **Tags** field can be changed from **AOS** to **ORCH** or a similar value. Microsoft also used the **fields.env** field to set the environment of the deployment (sandbox, sandbox-n, or production). In this way, there is a cleaner separation when data from multiple environments and node types is queried.

Winlogbeat includes a service installer. Microsoft used this installer to set up Winlogbeat as a service on each node. Press the Windows logo key+R to start the Run tool, and then run the following command.

```
powershell.exe -ExecutionPolicy Bypass -File C:\ELK\Winlogbeat\install-service-winlogbeat.ps1
```

Kibana

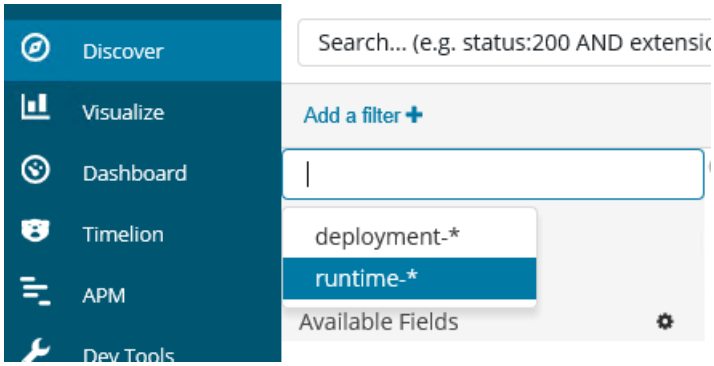
Kibana provides the interface to query the diagnostic data in Elasticsearch.

Microsoft downloaded Kibana to C:\ELK\Kibana and configured the kibana.yml file in the following manner.

```
server.host: "10.0.0.14"
server.name: "Dyn365FinOps On-Premises Diagnostics"
elasticsearch.url: "http://ORCH1:9200"
```

From Kibana, Microsoft had to define index patterns on the **Management** tab. Because index patterns group indexes by name, an index pattern was required for the two indexes that were made: deployment-* and runtime-*. The index names are case-sensitive.

Microsoft set the runtime-* index pattern as the default pattern. When you're looking at the index patterns on the **Management** tab, click the asterisk (*). The index pattern will then appear on the **Discover** tab.



Microsoft ran Kibana as a service in the same manner as Logstash, so that Kibana is started at OS startup. Unlike Logstash, kibana.bat doesn't need the path of the configuration files. Therefore, you can just install an NSSM service that points to C:\ELK\Kibana\6.2.4\bin\kibana.bat.

If you want users to browse Kibana on your network, remember to open the port for Kibana. The default port is 5601.

Example queries on the Discover tab in Kibana

The following sample queries can help you start probing the diagnostic data. If you require something more than the examples show, you can try one of the following queries:

- **Find slow database queries:** Enter **slow** in the search field to find events that have the word "slow" somewhere in the event data. If you want to be more precise, you can find events that have a task name of **AosDatabaseSlowQuery** and then enter **TaskName:AosDatabaseSlowQuery** in the search field.
- **Find recent exceptions:** Enter **exception** in the search field to find events that have either thrown an exception, or handled an exception and logged it. In the upper-right corner of Kibana, you can select the time frame that the search should be limited to. The time frame that you set there is persisted between tabs. Therefore, the data on the **Visualize** tab will reflect the selected time frame.

- **Find events from an AOS node:** Enter **host:AOS1** in the search field to find all events from that node.
- **Find events with proximity, in time, to another:** When you've found an event that you're interested in, click **View surrounding documents** next to the header of that event to find events that occurred at the same time. If you see events that occurred at around the same time but from different AOS nodes, you can add additional filtering to view only events from the node that you want.

Thirty-day data retention

To keep its hard disks free from stale data, Microsoft used Curator v5.5 to clean up indexes that were older than 30 days.

Microsoft downloaded Curator to one of the Orchestrator nodes at C:\ELK\Curator. Microsoft then used the configuration file that was put in C:\ELK\Curator to connect Curator to its Elasticsearch cluster.

Curator runs actions, and Microsoft created an action to clean up 30-day-old indexes that were put in C:\ELK\Curator.

Microsoft created a basic task in Windows Task Scheduler. This task has a weekly trigger on Saturday and Sunday, and the trigger has the following settings to start a program:

- **Program/script:** C:\ELK\Curator\curator.exe
- **Add arguments:** --config curator.yml .\30day_data_retention_actions.yml
- **Start in:** C:\ELK\Curator

X-Pack

> **IMPORTANT**
> As of June 2018, Elastic Stack components have been released that start with version 6.3. This updated version handles X-Pack in a more graceful manner, by enabling the free features of X-Pack by default, without requiring that you update the license every year, and by letting you opt in to the paid features afterward. If you install an Elastic Stack version that is earlier than 6.3, the content in this section only partially applies to the setup.

You can select to install X-Pack when you install Elasticsearch. Alternatively, you can install it later.

X-Pack has a free basic license that must be updated every year.

Microsoft installed the free version to enable query data to be exported from Kibana in comma-separated value (CSV) format. There are other useful features in X-Pack, but some are available only in a paid subscription.

To enable only the free features and avoid using other X-Pack trial features, Microsoft added the following settings to our elasticsearch.yml and kibana.yml configuration files:

```
xpack.graph.enabled: false
xpack.ml.enabled: false
xpack.security.enabled: false
xpack.watcher.enabled: false
```

These settings also stop Kibana and Elasticsearch from asking for credentials because the security module is no longer enabled.

For X-Pack to work, the logstash.yml configuration file must also be configured in the following manner.

```
xpack.monitoring.elasticsearch.url: "http://orch1:9200"
```

The paid version of X-Pack includes HTTPS encryption for connections throughout the cluster, password-protected data access, and more. For more information about X-Pack, see the Elastic website.

Export a query to a CSV file

In Kibana, on the **Discover** tab, write a query, and save it. After you save the query, on the **Reporting** tab at the top of the **Discover** page, click **Generate CSV**.

Troubleshooting

You don't receive any data in Kibana

If you don't receive any data in Kibana, review the logs from Winlogbeat to Logstash, Elasticsearch, and Kibana. Note that the Winlogbeat installation puts its logs in C:\ProgramData\winlogbeat\Logs, whereas the other Elastic Stack components put their logs close to the installation path (for example, in C:\ELK\Elasticsearch\logs).

On-premises disaster recovery configuration

2/18/2021 • 8 minutes to read • Edit Online

Disaster recovery is an important consideration for on-premises deployments of Dynamics 365 Finance + Operations (on-premises) to protect from events that could put your organization's operations at risk. Examples of such events include equipment failures, datacenter break downs due to cyberattacks, electrical, physical, or other disasters.

The core concept of disaster recovery involves the use of a second datacenter including a data recover environment. We recommend that you plan, document, and test disaster recovery as carefully as your production setup.

Limitations of this content

This topic does not cover specific configuration details for disaster recovery of the following components:

- Active Directory Federation Services (AD FS)
- File storage
- SQL Server

> **NOTE**
> High availability configuration isn't covered in this topic. For more information about the minimum setup required for high availability, see System requirements for on-premises deployments.

Recommendations

Remember to keep your disaster recovery environment updated with the latest Windows Updates. Your environment should have the latest security updates and not require updates during a disaster event.

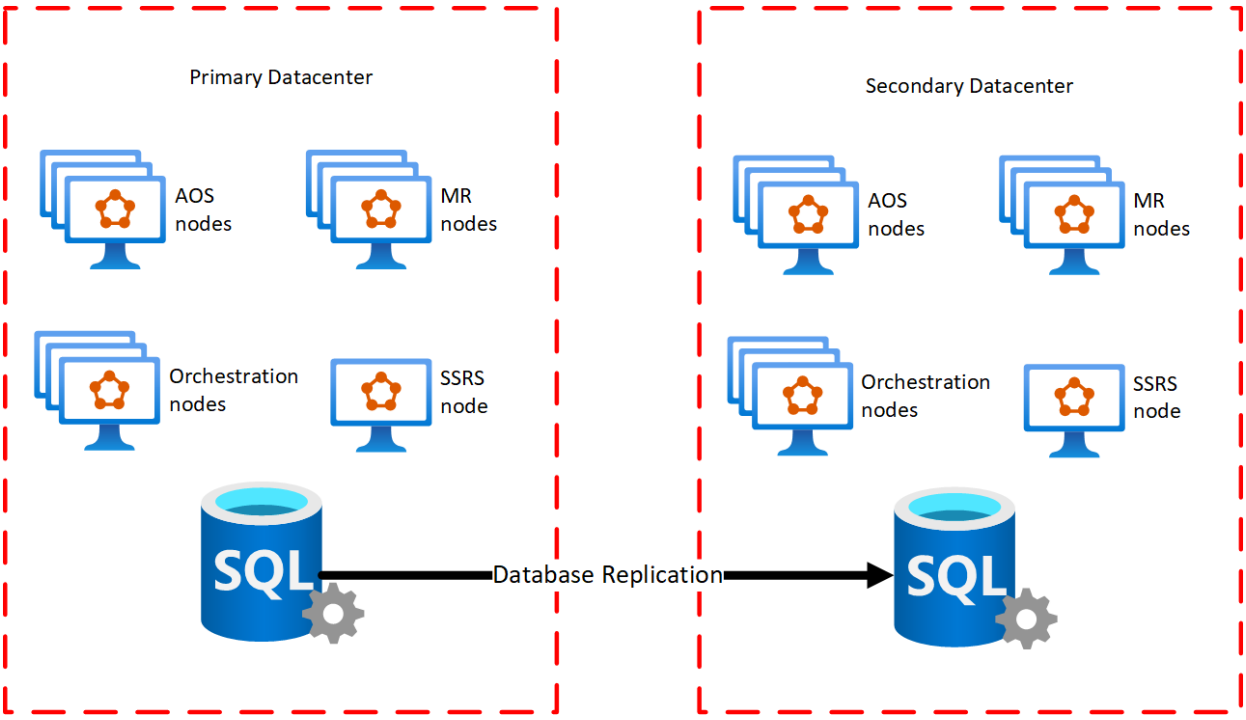
Ensure that you're applying new prerequisites that are specified by Microsoft. Also, keep your Service Fabric cluster updated and perform certificate rotations as required.

After you've read through this topic, document the steps that need to be taken by your team. After you've done that, go through the steps multiple times to ensure that you don't encounter unexpected problems and you minimize the potential downtime.

Overview

The basic configuration for disaster recovery involves deploying a duplicate of the production environment within another datacenter (the secondary datacenter) and replicating databases to that datacenter. If a disaster event takes place, a few manual steps can be taken to bring the environment that is within the secondary datacenter online.

The following diagram illustrates the required setup, at a high level.



Environment configuration

In Lifecycle Services (LCS), the production environment should be deployed using the environment slot named **Production**. Your disaster recovery environment will not use an additional environment slot in LCS. It will instead reuse the slot for your production environment.

Finance and Operations AOS nodes and SQL Server must be co-located within the same datacenter. For more information, see System requirements for on-premises deployments.

Deploying code packages to production

When code packages are deployed to the production environment, they don't need to be deployed to the disaster recovery environment. That environment should be unused and no Service Fabric services should be deployed.

Environment deployment settings

The disaster recovery environment should have a similar configuration as the production environment. The following table illustrates the shared and specific settings for disaster recovery.

ENVIRONMENT SETTINGS	DISASTER RECOVERY ENVIRONMENT
Active Directory settings	
Administrator user	Same as production
AD FS URL	Same as production
AD FS OpenId Connect client ID for AOS	Same as production
AD FS OpenId Connect client ID for Financial Reporting	Same as production
SQL Database configuration	

ENVIRONMENT SETTINGS	DISASTER RECOVERY ENVIRONMENT
SQL Server name	Same as production
AX database name	Same as production
Financial Reporting database name	Same as production
File share settings	
File share for document store	Same as production
File share certificate thumbprint	Same as production
SSRS configuration settings	
IP address of SSRS instance	Can be different [1]
SSRS certificate thumbprint	Same as production
Configure service settings	
DNS host name of Dynamics 365 instance	Can be different [2]
AOS service user	Same as production
MR application service user	Same as production
MR process service user	Same as production
MR click-once service user	Same as production
Application certificate settings	
Data encryption certificate thumbprint	Same as production
Data signing certificate thumbprint	Same as production
Session authentication certificate thumbprint	Same as production
SSL certificate thumbprint	Same as production
Management reporter certificate thumbprint	Same as production

[1] SSRS is referenced by IP. If the exact machine IP can't be configured in the disaster recovery environment, the IP can be different.

[2] This depends on your network configuration. If you have a load balancer that can handle diverting traffic to the other environment, then the host name can be the same. If you're unable to do that, then use a different host name.

SQL Server Always-On Availability configuration

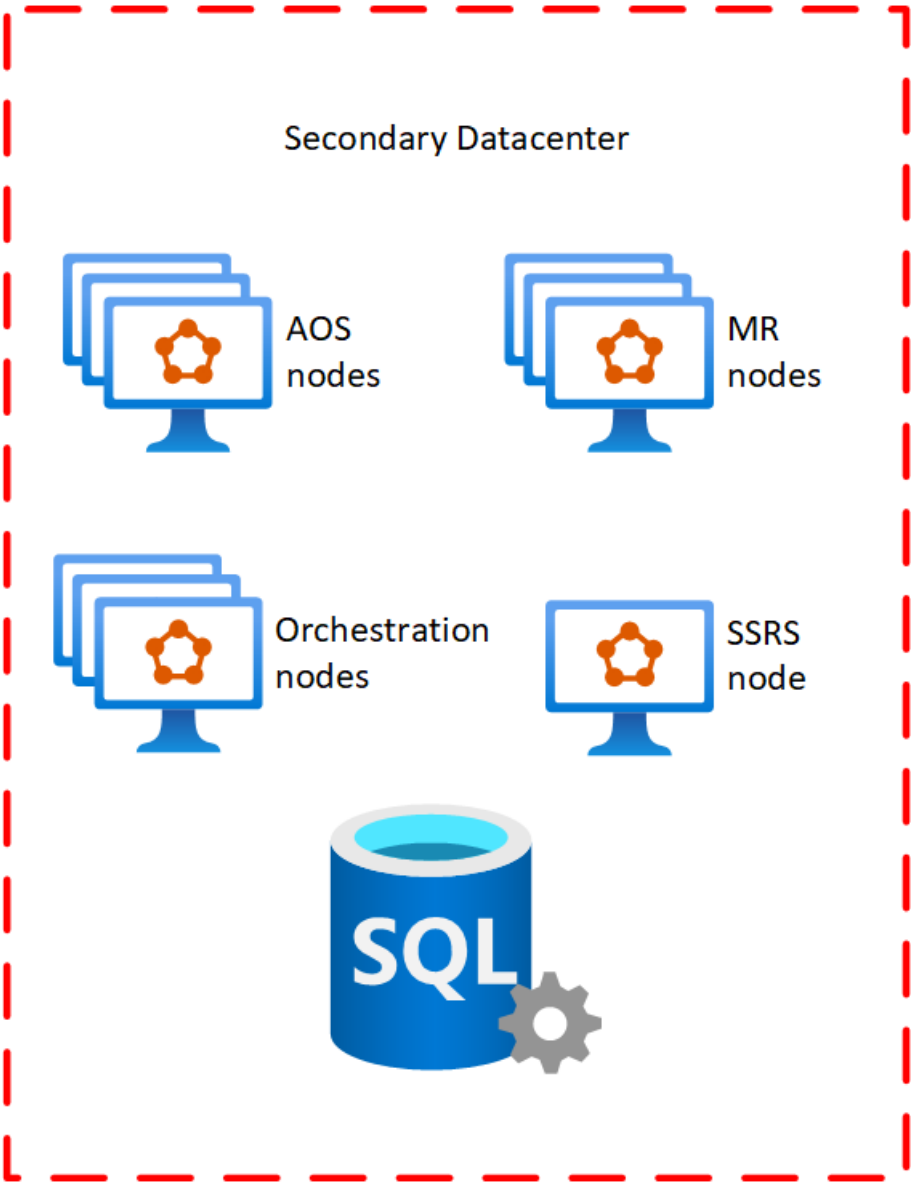
The business data database (AXDB) should be replicated to the secondary datacenter, typically using SQL Server Always-On availability groups feature. For more information, see Always On availability groups.

DATABASE	REPLICATED
Business data (AXDB)	Yes
Financial Reporting	Yes
BYODB	Yes
OrchestratorData	Yes

Failing over to disaster recovery

Overview

When a disaster event occurs, the primary datacenter may be unavailable but within the secondary datacenter, the following components will be available.



At the initial moment of the disaster event, the disaster recovery environment will be empty. The only thing present will be a configured Service Fabric cluster and SQL Server, which contain all of the replicated production data.

To bring the disaster recovery environment online, you'll need to have LCS deploy what is currently available in your Production environment into the disaster recovery environment.

> **IMPORTANT**
>
> Before you continue, ensure that no Dynamics Service Fabric services are running in your production environment (in case you're only failing due to a partial disaster event).

Deploy the LocalAgent

Download the LocalAgent installer and configuration file from LCS to your disaster recovery environment. After you have the configuration file, open it. Ensure that the connectionEndpoint under the serviceFabric section points to the IP or FQDN of a server in the disaster recovery environment. After modifying the file, save it locally and deploy the LocalAgent as you typically would.

> **IMPORTANT**
>
> Do not make changes to your connector settings in LCS.

Until your main production environment comes back online, this LocalAgent will process all requests that LCS puts into the message queue. That's why it's important that you ensure no services are running in your production environment. Eventually, when your orchestrator nodes come back up in your primary datacenter, unprovision the LocalAgent from the cluster.

Caution

The LocalAgent must only be running in one datacenter at a time. At this point it should only be running in your secondary datacenter.

Prepare your pre-deployment scripts (optional)

Pre-deployment scripts are necessary when changes to the deployment configuration are required. This script will have to modify the config.json file with the values you specify. It will be the customers' responsibility to create this script.

You can find the location of the config.json file by running the following command.

```
  select Location from DeploymentInstanceArtifact where AssetId='config.json' and DeploymentInstanceId =
'LCSENVIRONMENTID'
```

> **NOTE**
>
> Replace **LCSENVIRONMENTID** with the ID of your environment. You can obtain this ID from the full details page for your environment in LCS.

If the SSRS node IP is different, you'll have to modify the following values.

```
        "biReporting": {
          "persistentVirtualMachineIPAddressSSRS": {
            "value": "192.168.5.31"
          },
          "reportingServers": {
            "value": "192.168.5.31"
          },
```

If you are changing the host name, the following modifications are required.

```
    "name": "AOS",
      "parameters": {
        "activeDirectory": {
          ...
          "aadValidAudience": {
            "value": "https://ax.contosoen05.com/"
          },
          ...
        "infrastructure": {
          "hostName": {
            "value": "ax.contosoen05.com"
          },
          ...
        }
    ...
    "name": "FinancialReporting",
      "parameters": {
        ...
        "aad": {
         ...
         "cookieDomain": {
            "value": "ax.contosoen05.com"
          },
          "validAudiences": {
            "value": "https://ax.contosoen05.com/"
          },
          ...
```

IMPORTANT

If you are changing the hostname URL for your deployment, ensure that your AD FS server is configured to accept the new URL. For more information, see Reuse the same AD FS instance for multiple environments.

If the file share is shared between the production and disaster recovery environments, this pre-deployment script should be disabled. Only enable it when deploying to your disaster recovery environment.

Ensure reports get deployed

Because the database has previously been synchronized successfully, synchronization typically would be skipped. However, to synchronize the reports because the SSRS node is empty, perform the following actions.

Version 10.0.13 or later

Run the following command against your business data database (AXDB):

```
 UPDATE SF.synclog SET STATE=5, SyncStepName = 'ReportSyncstarted' WHERE CODEPACKAGEVERSION in (SELECT
TOP(1) CODEPACKAGEVERSION from SF.SYNCLOG ORDER BY CREATIONDATE DESC)
```

Version 10.0.12 or earlier

Run the following command against your business data database (AXDB):

```
    DELETE FROM SF.synclog WHERE CODEPACKAGEVERSION in (SELECT TOP(1) CODEPACKAGEVERSION from SF.SYNCLOG
ORDER BY CODEPACKAGEVERSION DESC)
```

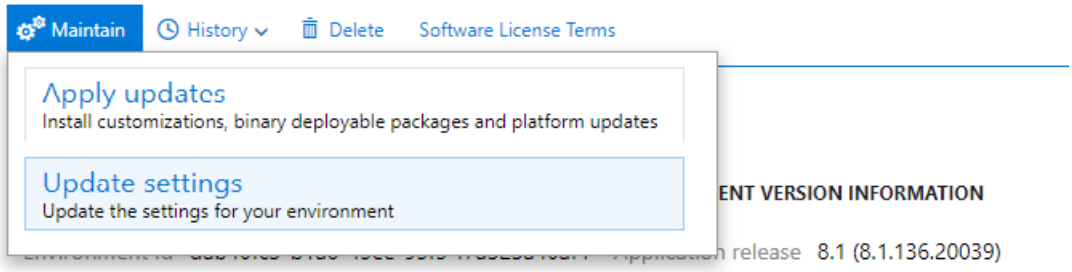
NOTE

If you are using version 10.0.12 or earlier, a full database synchronization will be executed.

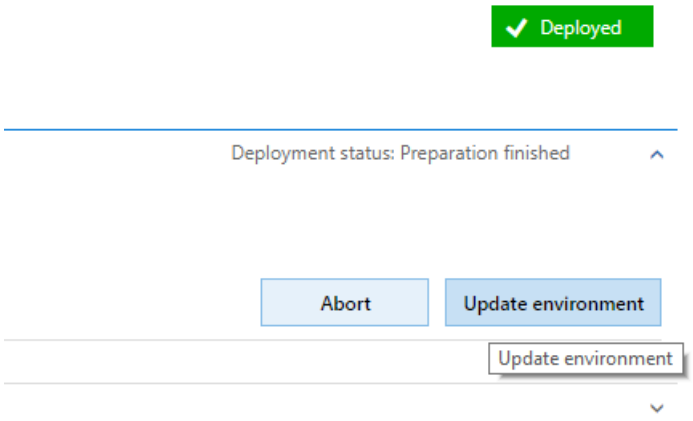
Deploy your environment

To deploy your environment, follow these instructions.

1. In LCS, go to the environment page for your production environment.
2. Select **Maintain** and then select **Update Settings**.



3. Don't change any values. Select **Prepare**.
4. After downloading is finished and preparation is completed, the **Update environment** button will be displayed. Select this button to start updating your environment



5. After the environment is deployed, the disaster recovery environment is ready for use.

Using your disaster recovery environment

You can use your disaster recovery environment as you typically would, except that updates or hotfixes shouldn't be applied to the environment. If you must apply updates to your environment, your failback process will differ from the one described below. Failing back under this condition is not covered in this topic.

Failing back to your production environment

> **IMPORTANT**
> At this point, no Dynamics Service Fabric services should be running in your production environment.

Secure a downtime window in which you can switch operation from the disaster recovery environment to the Production environment. In the downtime window, disable all non-Orchestrator nodes in the disaster recovery environment through Service Fabric Explorer. Once all nodes are disabled, failover your SQL Server to the production datacenter.

After the failover has occurred, start the AOS, SSRS, and MR nodes in your primary datacenter. Carry out validation tests to ensure that your environment is functioning as expected. When you determine that your environment is working as expected, remove the LocalAgent from your disaster recovery environment and

reinstall it on your Production environment.

Clean up your DR environment by manually unprovisioning all Dynamics Service Fabric services.

Caution

Do not use the Cleanup functionality in LCS to perform the clean up of your disaster recovery environment.

Failback checklist

- Non-orchestrator nodes are disabled in disaster recovery datacenter.
- SQL Server is failed back to primary datacenter.
- LocalAgent is uninstalled in your disaster recovery datacenter.
- All Dynamics Service Fabric services (including LocalAgent) are running in your primary datacenter.
- No Dynamics Service Fabric services are deployed in your disaster recovery datacenter.

IMPORTANT

Your primary environment will be functioning as usual and can be serviced after you ensure that all items in the checklist are verified.

PowerBI.com integration with on-premises environments

2/18/2021 • 5 minutes to read • Edit Online

The cloud version provides several features that allow for deeper integration with Microsoft Power BI. Some of these features haven't yet been implemented for on-premises deployments. However, the availability of Entity Store in on-premises deployments lets customers use PowerBI.com to report on and analyze data.

This topic outlines the analytical capabilities that are available in on-premises deployments that run Microsoft Dynamics 365 for Finance and Operations platform update 26 (May 2019) and later.

FEATURE/CAPABILITY	CLOUD	ON-PREMISES (PLATFORM UPDATE 26 OR LATER)
Analytical workspaces	Available	**Not yet implemented** Customers can use the reports that are built on Entity Store together with PowerBI.com.
Pin reports, tiles, and dashboards from PowerBI.com into the client.	Available	**Not yet implemented**
Author and distribute Power BI reports that use application data.	Available	**Available** Customers can modify ready-made reports and create new reports by using Entity Store on-premises.
Extract application data into data warehouses.	Available	**Available**

Enable Entity Store on-premises

This topic supplements the Set up and deploy on-premises environments (Platform update 12 and later) topic. The section numbers that follow correspond to the section numbers in that topic.

3. Plan your users and service accounts

USER ACCOUNT	TYPE	PURPOSE	USER NAME
AOS SQL AXDW DB Admin user	SQL user	The application uses this user to enter information in the AXDW database. This user is optional and must be created only if you want Entity Store support.	axdwadmin

USER ACCOUNT	TYPE	PURPOSE	USER NAME
AOS SQL AXDW FB Admin user	SQL user	The application uses this user to enter information in the AXDW database. This user is optional and must be created only if you want Entity Store support.	axdwruntimeuser

14. Configure the databases

The steps in this section are optional.

If you want to create a database that can be used for Entity Store, you must first modify the ConfigTemplate.xml file.

1. Under **DbServer – Security**, set the **generateUser** flags for **axdwadmin** and **axdwruntimeuser** to **True**. The scripts that you run in the next step will then create those two users. You will be prompted to set passwords for the users.

2. Run the following scripts.

   ```
   .\Initialize-Database.ps1 -ConfigurationFilePath .\ConfigTemplate.xml -ComponentName EntityStore
   .\Configure-Database.ps1 -ConfigurationFilePath .\ConfigTemplate.xml -ComponentName EntityStore
   ```

   The Initialize-Database.ps1 script performs the following actions:

   - Create an empty database that is named **AXDW**. This database is used for Entity Store.
   - Create a new user that is named **axdwadmin** and that has SQL authentication enabled, and prompt you for the user's password.
   - Create a new user that is named **axdwruntimeuser** and that has SQL authentication enabled, and prompt you for the user's password.
   - Grant the axdwadmin and axdwruntimeuser users **db_owner** permissions on the database.

   The Configure-Database.ps1 script performs the following actions:

   - Set the specified database file and log settings.
   - GRANT VIEW SERVER STATE TO axdwadmin.
   - GRANT VIEW SERVER STATE TO axdwruntimeuser.

15. Encrypt credentials

Create a Credentials.json file as shown here. The **AosDWAuth** category is optional and is used only if Entity Store is enabled.

```
{
    "AosPrincipal": {
        "AccountPassword": "<encryptedDomainUserPassword>"
    },
    "AosSqlAuth": {
        "SqlUser": "<encryptedSqlUser>",
        "SqlPwd": "<encryptedSqlPassword>"
    },
    "AosDWAuth": {
        "DWUser": "<encryptedDWUser>",
        "DWPwd": "<encryptedDWPassword>",
        "DWRuntimeUser": "<encryptedDWRuntimeUser>",
        "DWRuntimePwd": "<encryptedDWRuntimePassword>"
    }
}
```

Here is an explanation of the preceding code lines:

- **AccountPassword** is the encrypted domain user password for the Application Object Server (AOS) domain user (contoso\axserviceuser).
- **SqlUser** is the encrypted SQL user (axdbadmin) that has access to the application database (AXDB). **SqlPwd** is the encrypted SQL password.
- **DWUser** is the encrypted SQL user (axdwadmin) that has access to the Entity Store database (AXDW). **DWPwd** is the encrypted SQL password that is entered when the Initialize-Database.ps1 script is run.
- **DWRuntimeUser** is the encrypted SQL user (axdwruntimeuser) that has access to the Entity Store database (AXDW). **DWRuntimePwd** is the encrypted SQL password that is entered when the Initialize-Database.ps1 script is run.

More information

Entity Store was enabled in Platform update 26.

Creation of the Entity Store database and users is optional. When you configure a deployment in Microsoft Dynamics Lifecycle Services (LCS), you can leave the Entity Store customizations blank.

If Entity Store should be enabled in an environment, the following tasks must be completed:

- Create an **AXDW** database by using the scripts that are described in step 14.
- Create **axdwadmin** and **axdwruntimeuser** users that have appropriate privileges, by using the scripts in that are described step 14. Users are defined in the following files: ConfigTemplate.xml and DatabaseTopologyDefinition.xml.
- Create a **Credentials.json** file as described in step 15. User names and passwords should be defined for axdwadmin and axdwruntimeuser as described in step 14, and they should be encrypted.
- The Entity Store SQL Server name and Entity Store database name must be filled in during deployment in LCS.
    - The database name is created in step 14 and is defined in the DatabaseTopologyDefinition.xml file. The default name is **AXDW**.
    - The SQL Server name is the fully qualified domain name (FQDN) of the Microsoft SQL Server or Always on listener. An example of this FQDN is **sqlinstance.onprem.contoso.com**. It's the server that the AXDW database is created on.

Currently, you can enable Entity Store only during initial deployment. If it must be enabled in an existing environment, that environment must be deleted and then deployed again by using a platform update that supports Entity Store.

Authoring and distributing reports by using Entity Store on-premises

Entity Store is an operational data warehouse that is included. It lets power users and business analysts author reports that use simplified and enriched data. Entity Store contains aggregate measurements, which are simplified star schemas. For more information about how to author reports by using Entity Store, see Create analytical reports by using Power BI Desktop.

Note the following additional steps that are specific to on-premises deployments:

- For on-premises environments, you don't have to use LCS to deploy Power BI reports to production or sandbox environments. Because admins can point PowerBI.com datasets to specific Entity Store databases in on-premises environments, you don't have to use the functionality that LCS offers. However, you might have to configure the on-premises gateway to enable PowerBI.com to access data on-premises. For more information about the gateway, see Power BI gateway documentation.
- Although cloud-based application environments support only reports that are authored by using the DirectQuery option, on-premises Entity Store supports both DirectQuery reports and Import mode reports.
- Analytical workspaces aren't yet implemented in on-premises deployments. Instead of viewing reports in analytical workspaces, you can deploy them to PowerBI.com environments. The reports can then be used by users who have access to PowerBI.com. Your users might require appropriate licenses to access reports on PowerBI.com.

Reconfigure environments to take a new platform or topology

2/18/2021 • 2 minutes to read • Edit Online

This topic provides information about how to reconfigure your environment with a new platform or topology and how to update the configuration of your existing environment.

Prerequisites

Before you complete the steps in this topic, you must update your local agent. For more information, see the topic, Update the local agent. The procedure in this topic will only work with local agents that are on or above version 1.1.0.

Reconfigure your environment

1. In Lifecycle Services (LCS), navigate to your on-premises project and open the **Environments** blade.
2. Do one of the following based on whether you are going to reconfigure or take a new deployment.

- If you need to reconfigure your environment, click **Reconfigure**.
- If you need to take a new deployment or topology, select the new topology for your platform and enter the environment name. You can use the same name or enter a new one.

3. Click **Advanced Settings** to update your configuration. The configuration from your previous deployment will be saved.

Redeploy on-premises environments

2/18/2021 • 3 minutes to read • Edit Online

At some point, you might have to redeploy your on-premises environment. This could be to apply a new platform update or because of changes or issues in your implementation. Before you delete the environment you are currently working with, you should save your configuration setting information to use when you redeploy. This topic describes how to save configuration settings and how to redeploy your environment.

Save your configuration

Before you delete the environment you plan to update, use the following steps to save your configuration.

1. In LCS, navigate to **Project Settings** > **On-prem Connectors**.
2. Select the connector to your environment, and then click **Edit**.
3. On the **Edit connector** tab, navigate to **Configure Agent** > **Enter Configuration**.
4. Copy the value of the Download Fileshare location in the **Configuration Settings** section. You will need this later.
5. Log in to the on-premises environment file share machine and copy the **\agent\wp\StandaloneSetup\config.json**. You can use the configuration settings in this json file to redeploy your environment.

Configuration settings

The following tables provide information about configuration settings. Use the **Configuration setting** value from the .json file that you saved in the previous procedure.

Active Directory Federation Services settings

FIELD	CONFIGURATION SETTING
The email address of the user who will be the initial administrator (such as, `adminuser@yourdomain.com`)	components.(AOS).parameters.provisioning.adminPrincipleName.value
ADFS OpenID metadata endpoint for the Dynamics 365 Application group. (such as, https://[federation-service-name]/adfs/.well-known/openid-configuration)	components.(AOS).parameters.activeDirectory.adfsMetadata.value
ADFS OpenID Connect client ID for the AOS application group	components.(AOS).parameters.activeDirectory.adfsClientId.value
ADFS OpenID Connect client ID for the Financial Reporting application group	components.(FinancialReporting).parameters.aad.nativeClientAuthentication.clientId.value

SQL database configuration

FIELD	CONFIGURATION SETTING
SQL SERVER	components.(AOS).parameters.database.dbServer.value
AX DATABASE	components.(AOS).parameters.database.dbName.value

FIELD	CONFIGURATION SETTING
FINANCIAL REPORTING DATABASE	components.(FinancialReporting).parameters.mrdb.dbName.value

File share settings

FIELD	CONFIGURATION SETTING
The file share path for the Microsoft Dynamics 365 instance. This share is used as the document store for files uploaded by users.	components.(AOS).parameters.storage.fileSharePath.value
The File share certificate thumbprint for the Microsoft Dynamics 365 instance.	components.(AOS).parameters.storage.sharedAccessThumbprint.value

> **NOTE**
>
> When you copy the file path configuration value from .json file to LCS UI, make sure to remove the extra backslashes. For example, configuration value **\\DC1\D365FFOStorage** from the .json file should be **\DC1\D365FFOStorage** in the LCS UI.

SSRS configuration settings

FIELD	CONFIGURATION SETTING
The IP Address of the SSRS instance.	components.(AOS).parameters.biReporting.persistentVirtualMachineIPAddressSSRS.value
The thumbprint used by the SSRS application to communicate with AX Service.	components.(ReportingServices).parameters.reportingClientCertificateThumbprint.value

Configure service settings

FIELD	CONFIGURATION SETTING
DYNAMICS 365 DNS INFORMATION - The DNS host name of the Microsoft Dynamics 365 instance, such as ax.d365ffo.onprem.contoso.com.	components.(AOS).parameters.infrastructure.hostName
AOS SERVICE PRINCIPAL USER SETTINGS - The domain user account to run the AX service, such as yourdomain\axserviceuser.	components.(AOS).parameters.infrastructure.principalUserAccountName *
MR SERVICE PRINCIPAL USER SETTINGS - The group managed service account (gMSA) to run the MR application service, such as yourdomain\Svc-FRAS$.	components.(FinancialReporting).parameters.ApplicationServicePrincipalUser.accountName.value *
The group managed service account (gMSA) to run the MR process service, such as yourdomain\Svc-FRPS$.	components.(FinancialReporting).parameters.ProcessServicePrincipalUser.accountName.value *

FIELD	CONFIGURATION SETTING
The group managed service account (gMSA) to run the MR click-once service, such as yourdomain\Svc-FRCO$.	components.(FinancialReporting).parameters.ClickOnceServicePrincipalUser.accountName.value *

> **NOTE**
> Remove the extra backslash from the Principal username cofiguration value in the .json file before entering in the LCS UI. For example, contoso\\AXServiceUser should be entered as contoso\AXServiceUser in LCS.

Application certificate settings

FIELD	CONFIGURATION SETTING
The thumbprint of the Data Encryption certificate.	components.(AOS).parameters.database.dataEncryptionCertificateThumbprint.value
The thumbprint of the Data Signing certificate.	components.(AOS).parameters.database.dataSigningCertificateThumbprint.value
The thumbprint of the Session Authentication certificate.	components.(FinancialReporting).parameters.sessionAuthenticationCertificateThumbprint.value
The thumbprint of the SSL certificate used for WCF/SOAP support.	components.(AOS).parameters.infrastructure.sslCertificateThumbprint.value
The thumbprint used by the Management Reporter to communicate with AX service.	components.(FinancialReporting).parameters.tokenSpec.certThumbprint.value

Redeploy your environment

The following instructions provide information about how to update or redeploy your environment with a new platform or topology.

1. In LCS, navigate to the **Environments** blade in your on-premises project.
2. Click **Delete** to delete your environment.

   > **NOTE**
   > Deleting the environment will not delete the database, infrastructure or Local agent. Only the Service Fabric applications are deleted.

3. Wait for a few minutes and verify that the deployment is deleted. To confirm the deployment is deleted, log in to the on-premises environment and navigate to the Service Fabric Explorer.

   The following applications should be deleted:

   - AXBootstapperAppType

- AXSFType
- FinancialReportingType
- RTGatewayAppType
- ReportingService

  The following on-premises service fabric agent applications should not be deleted:

- LocalAgentType
- MonitoringAgentAppType

4. After all of the applications in step 3 are deleted, go back to LCS and click **Configure**.
5. Select the new topology for your platform.
6. Enter the environment name. You can use the same name or enter a new one.
7. Click **Advanced Settings**. You can now use the relevant configurations from the .json file that you saved to configure your environment.

Remove and reinstall, or add an AOS node

2/18/2021 • 9 minutes to read • Edit Online

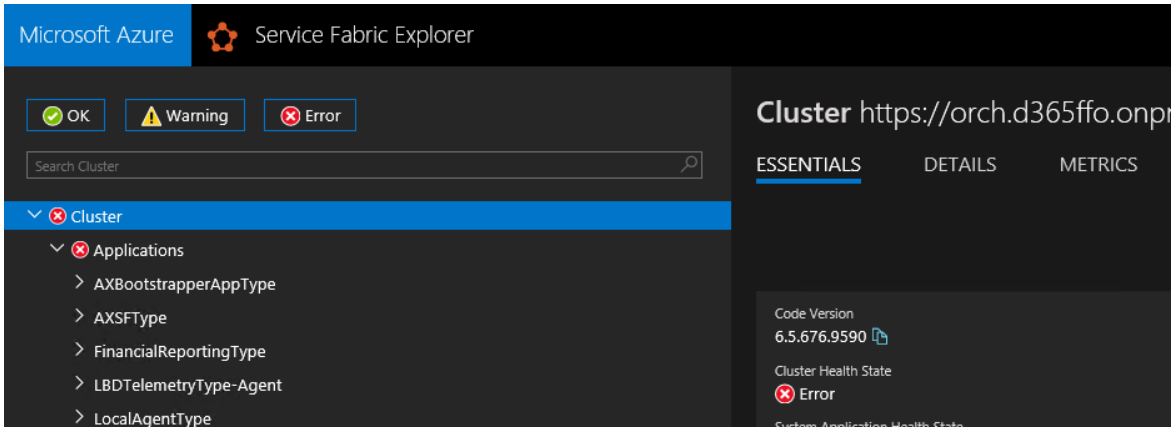
This topic explains how to remove an Application Object Server (AOS) node in your on-premises environment to reduce or replace a failed node. It also explains how to add a new AOS node for scale-out performance.

Remove a node

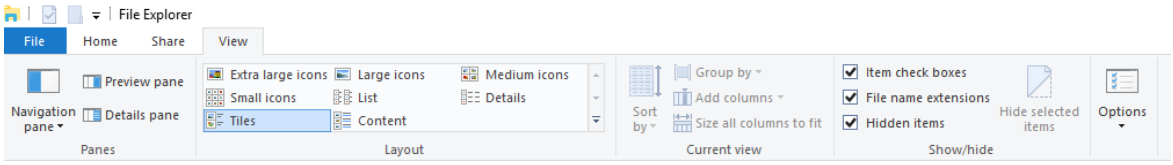
Option 1: Use a configuration file (preferred option)

Reference document: Add or remove nodes to a standalone Service Fabric cluster running on Windows Server

1. In Service Fabric Explorer, select **Cluster**, and make a note of the Microsoft Service Fabric cluster version. For this example, the cluster version is **6.5.676.9590**.



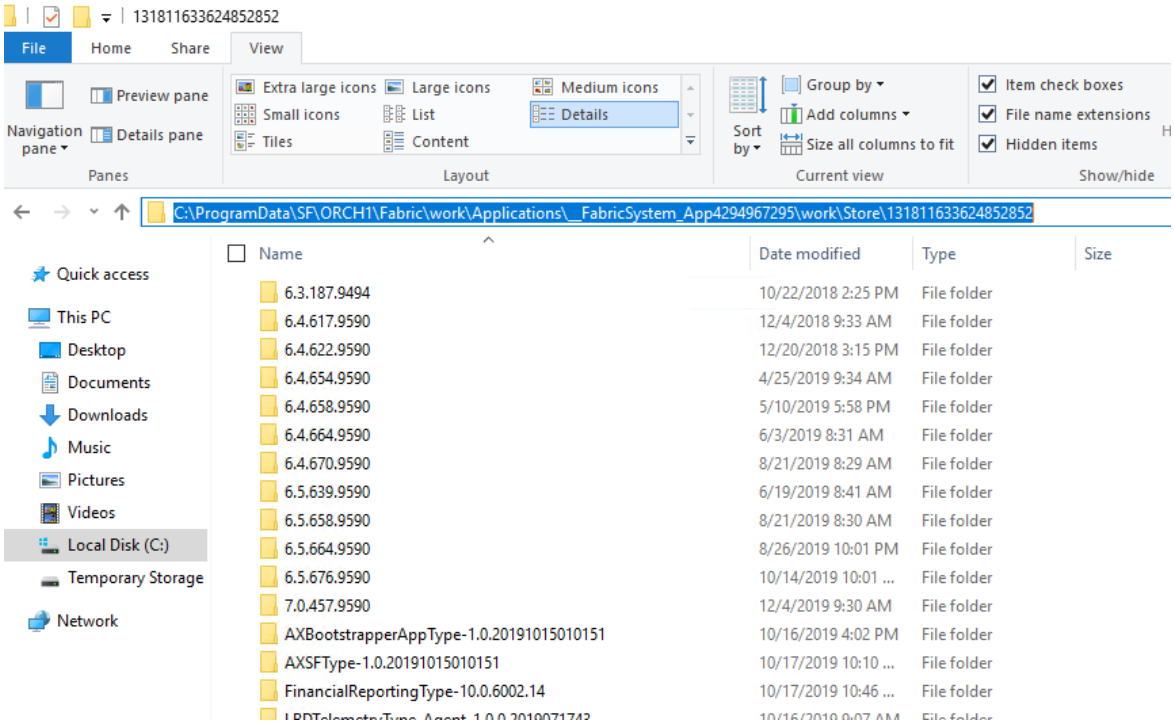
2. On one of the orchestrator nodes, open File Explorer. On the **View** tab, in the **Show/hide** group, make sure that the **File name extensions** and **Hidden items** check boxes are selected.



3. Expand drive C, and then drill down into the following folder. (Note that the bold parts of the path will vary, depending on the node name and setup.)

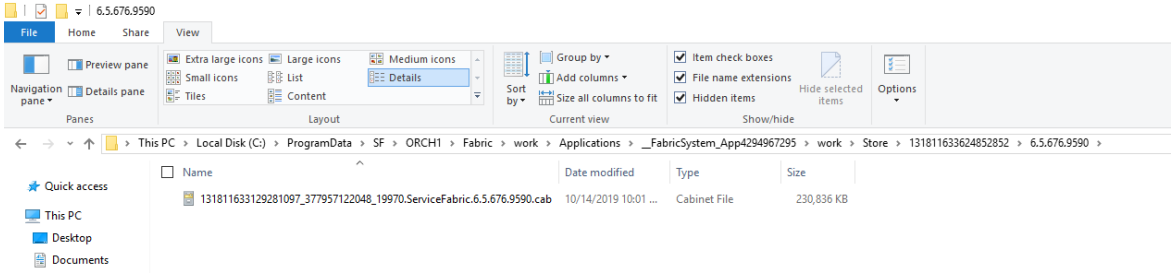
   C:\ProgramData\SF**ORCH1**\Fabric\work\Applications__FabricSystem_**App4294967295**\work\Store**131811633624852852**

   In the folder, you should see a list of folders for various versions of Microsoft Service Fabric. Here is an example.

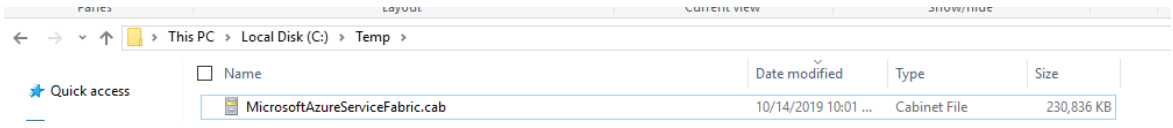


4. Open the folder with the name the same as the version of Microsoft Service Fabric cluster you that you made a note of earlier. For this example, the folder is named **6.5.676.9590**.

5. In the folder, you should see a .cab file.



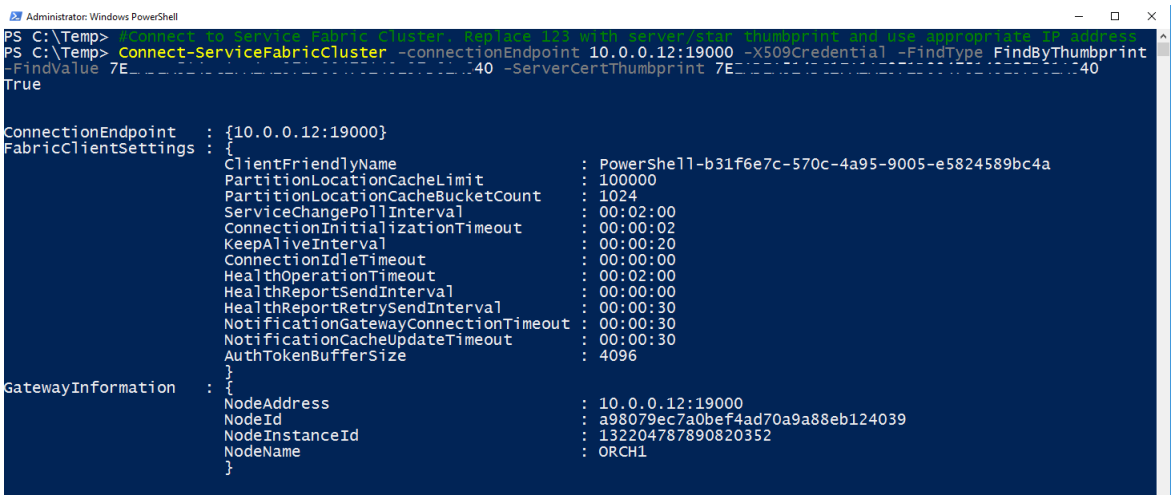
6. Copy the .cab file to C:\Temp, and rename the copied file **MicrosoftAzureServiceFabric.cab**. (If you don't have a Temp folder, create it.)



7. Open a Windows PowerShell Command Prompt window as an admin.

8. Run the following command to connect to the Service Fabric cluster.

```
#Connect to Service Fabric Cluster. Replace 123 with server/star thumbprint and use appropriate IP
address
Connect-ServiceFabricCluster -connectionEndpoint 10.0.0.12:19000 -X509Credential -FindType
FindByThumbprint -FindValue 123 -ServerCertThumbprint 123
```



9. Run the following command to save the configuration file to C:\Temp\ClusterConfig.json. (Make sure that the C:\Temp path exists.)

```
Get-ServiceFabricClusterConfiguration -UseApiVersion -ApiVersion 10-2017 >C:\Temp\ClusterConfig.json
```

10. In the configuration file that you saved in the previous step, in the **fabricSettings** section, in the **Setup** section, add a section for the **NodesToBeRemoved** parameter. The parameter value should be a comma-separated list of names of the nodes that must be removed.

> **NOTE**
>
> Be sure to add a comma to the end of the line that precedes the new section.

```
"fabricSettings": [
    {
        "name": "Setup",
        "parameters": [
            {
                "name": "FabricDataRoot",
                "value": "C:\\\\ProgramData\\\\SF"
            },
            {
                "name": "FabricLogRoot",
                "value": "C:\\\\ProgramData\\\\SF\\\\Log"
            },
            {
                "name": "NodesToBeRemoved",
                "value": "AOS1"
            }
        ]
    }
]
```

11. Remove the node from the **Nodes** section. In the following example, the **AOS1** node was removed.

```
"Nodes": [
    {
        "NodeName": "AOS2",
        "NodeTypeRef": "AOSNodeType",
        "IPAddress": "10.0.0.10",
        "FaultDomain": "fd:/fd1",
        "UpgradeDomain": "ud1"
    },
    {
    "NodeName": "AOS3",
    "NodeTypeRef": "AOSNo…
```

12. Remove the following lines from the **Security** section.

```
"WindowsIdentities": {
    "\$id": "3"
},
```

> **NOTE**
>
> If you don't remove these lines, you will receive the following error message later:
>
> ValidationException: Authentication type can't be changed from unsecured to Windows.

13. Increment the version number of the configuration file. Make this change at the lowest increment. In the following example, the version number went from **1.0.0** to **1.0.1**.

```
"ClusterConfigurationVersion": "1.0.1"
```

14. Save the configuration file.

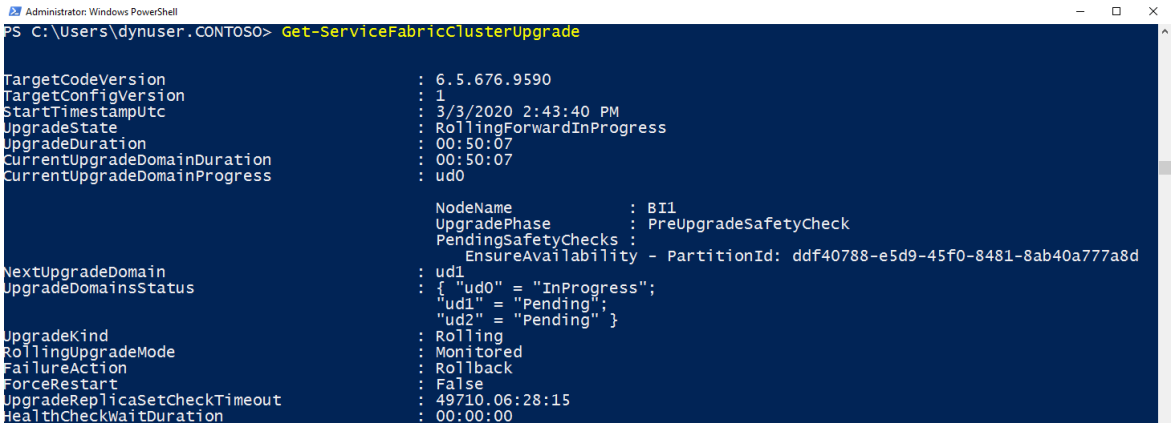
15. Run the following command to remove the node.

```
Start-ServiceFabricClusterConfigurationUpgrade -ClusterConfigPath C:\Temp\ClusterConfig.json
```

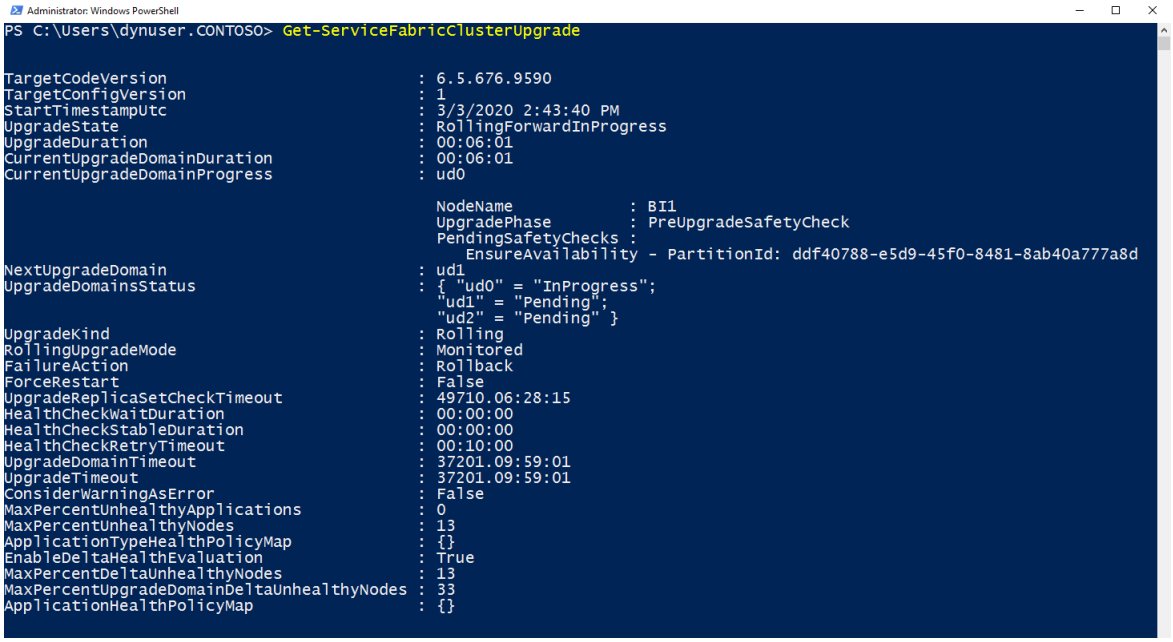
16. Run the following command to monitor the progress.

```
Get-ServiceFabricClusterUpgrade
```

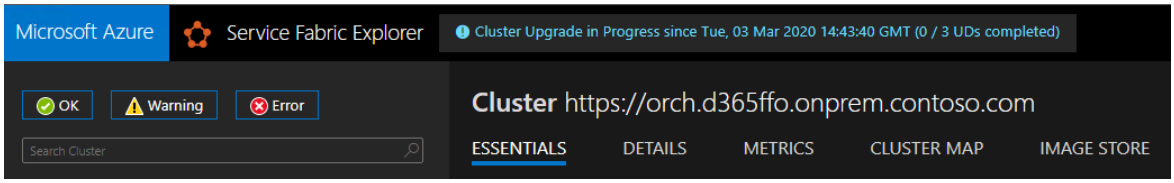
    If the upgrade stops responding at "UpgradePhase: PreUpgradeSafetyCheck," make a note of the **NodeName** value, and restart that node from Service Fabric Explorer. In the following illustration, the upgrade has stopped responding. It was running for 50 minutes at the same status on node BI1.



During upgrade of the cluster configuration, if you receive an error message that states that you previously added a node through the **Add-ServiceFabricNode** command, you will need to run a configuration upgrade without making any changes to the configuration file except for the version number. You can use the **Get-ServiceFabricClusterConfiguration** and **Start-ServiceFabricClusterConfigurationUpgrade** commands for this purpose.

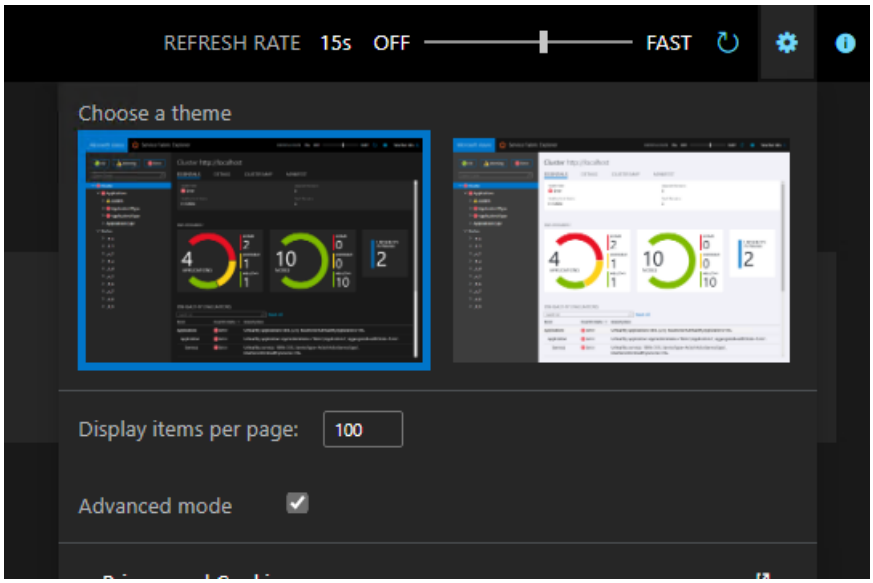


You can also view the progress in Service Fabric Explorer.

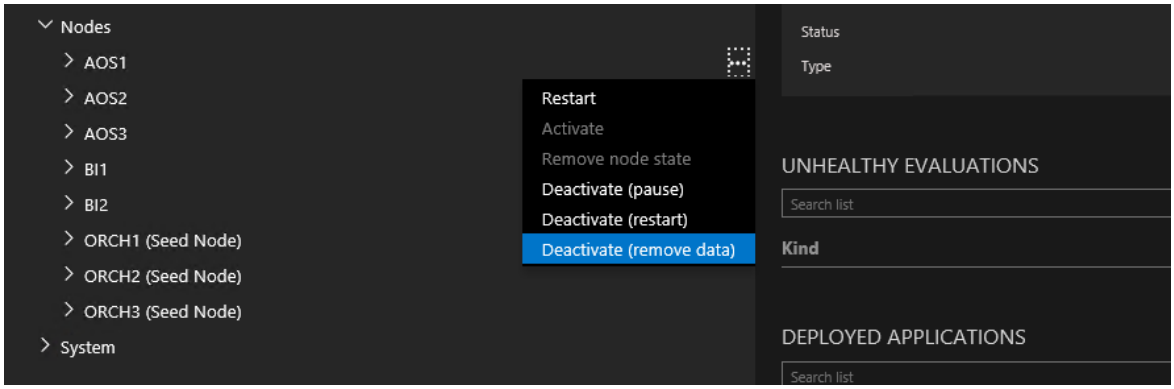


Option 2: Use Service Fabric Explorer

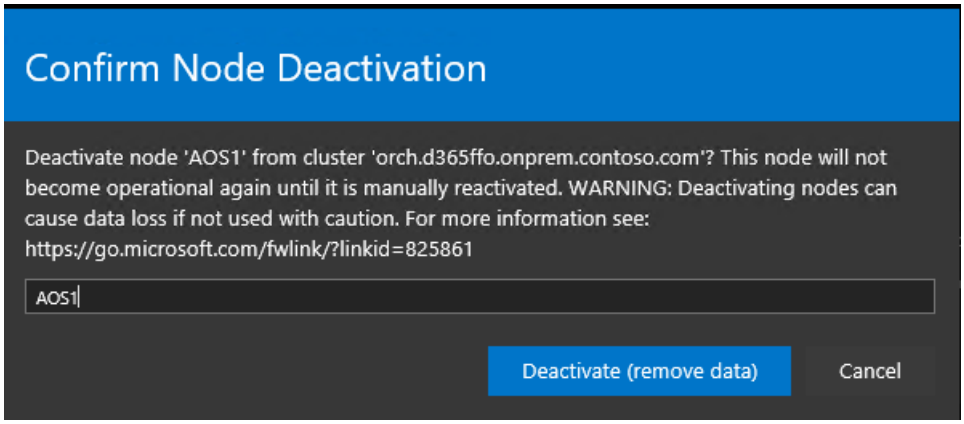
1. Sign in to Service Fabric Explorer.
2. Select the **Settings** button (gear symbol), and make sure that **Advanced** mode is turned on.



3. Expand **Nodes**, select the ellipsis (...) button next to the node that you want to remove, and then select **Deactivate (remove data)**. Note that this option might not be available if the node is already down (for example, if the node server can't be started).



4. When you're prompted to confirm deactivation, enter the name of the node, and then select **Deactivate (remove data)**.

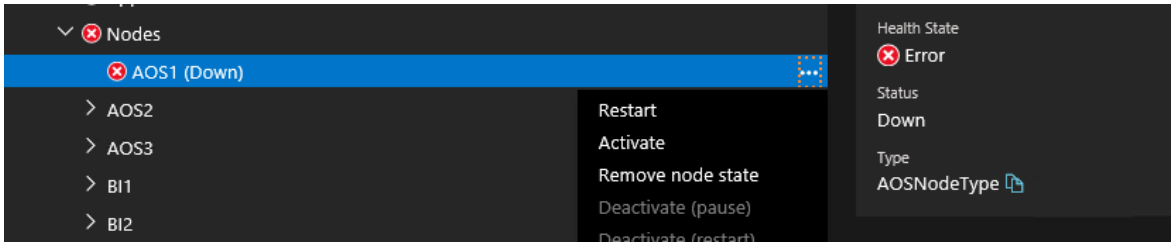


After the node has been deactivated, its status is shown as **Disabled**.

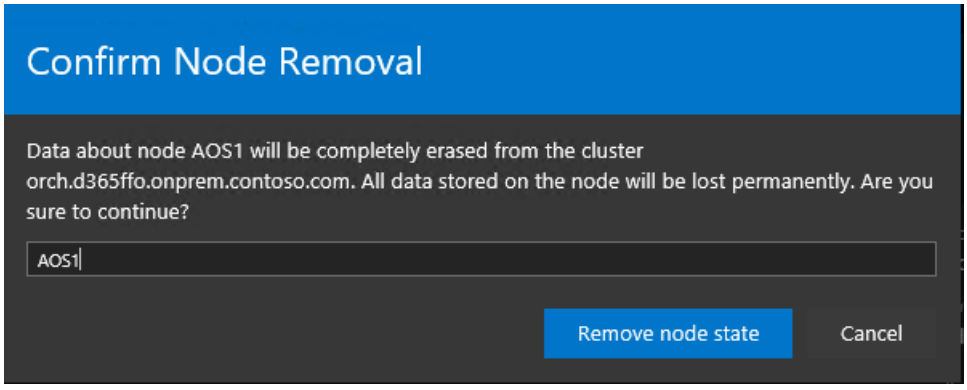
5. If the server is still active and connected to the domain, you might have to follow these steps if you will be replacing the deactivated node with a new server:

   a. Sign in to the server.
   b. Remove the server from the domain.
   c. Rename the server.
   d. Make a note of the IP address, and then change the IP address to a free address that you have in your range.
   e. Shut down the server.

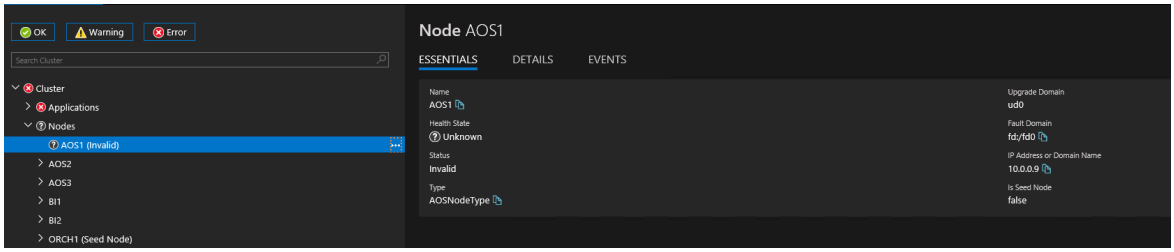
6. After the server has been shut down, or if it was already down, Service Fabric Explorer reflects its status. Select the ellipsis (...) button again next to the node, and then select **Remove node state**.



7. Confirm removal of the node.



After the node has been removed, its status is shown as **Invalid**.



8. Make a note of the node name and type. For this example, the node name is **AOS1**, and the type is **AOSNodeType**. Remember that the node name might not match the network name. Also make a note of the **Upgrade Domain** and **Fault Domain** settings, and the IP address. The previous illustration shows all these values.

Add a node

The next step is to start a new AOS server.

1. Follow these steps if you're replacing an existing server that was removed:

   a. Give the server the network name of the previous AOS server.
   b. Assign the original IP address. For this example, that IP address is **10.0.0.9**.
   c. Join the server to the domain.

2. If you're adding a new server to an existing cluster, update the ConfigTemplate.xml file so that it contains the additional information. This information will be used when you push out the prerequisites and apply settings through Windows PowerShell scripts.

3. Make sure that you've added the **AXServiceUser** and **svc-AXSF$** group Managed Service Accounts (gMSAs) to the local admin group on the AOS server.

   After the server is connected to the domain, you must follow the prerequisite steps for on-premises environments in Set up and deploy on-premises environments (Platform update 12 and later). The following steps summarize those prerequisite steps.

4. Copy the contents of each infrastructure\VMs<VMName> folder into the corresponding virtual machine (VM). (If you use remoting scripts, they will automatically copy the contents to the target VMs.) Then run the following Windows PowerShell scripts as an admin.

   > **NOTE**
   > If you're running remotely and repairing an existing server, you must delete the lbdscripts_remote_status.json file from the infrastructure folder to ensure the file copy process is run against all servers again.

   ```
   # Install pre-req software on the VMs.
   # If Remoting, execute
   # .\Configure-PreReqs-AllVMs.ps1 -MSIFilePath <share folder path of the MSIs> -ConfigurationFilePath
   .\ConfigTemplate.xml
   .\Configure-PreReqs.ps1 -MSIFilePath <share folder path of the MSIs>
   ```

5. Restart the computer every time that you're prompted. Make sure that you rerun the **.\Configure-PreReqs.ps1** script after every restart, until all the prerequisites are installed. In the case of remoting, rerun the **AllVMs** script when all the computers are back online.

6. If you use the remoting script, make sure that the current user has access to the share folder of Microsoft Windows Installer package files (.msi files).

7. If you use the remoting script, make sure that no user is accessing computers of the **AOSNodeType**, **MRType**, and **ReportServerType** types. Otherwise, the remoting script won't be able to restart the computer because users are signed in to it.

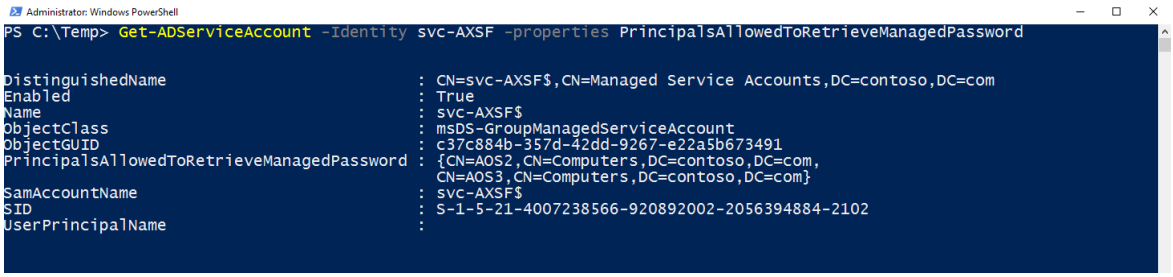
8. Run the following scripts, if they exist, to complete the VM setup.

   ```
   # If Remoting, only execute
   # .\Complete-PreReqs-AllVMs.ps1 -ConfigurationFilePath .\ConfigTemplate.xml
   .\Add-GMSAOnVM.ps1
   .\Import-PfxFiles.ps1
   .\Set-CertificateAcls.ps1
   ```

9. If errors occur while you run **Add-GMSAonVM.ps1**, you must run the following command. (Edit the command if your service account differs. Note that you remove the dollar sign [$] from the service

account name.)

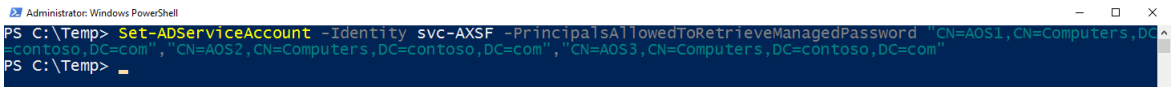
```
Get-ADServiceAccount -Identity svc-AXSF -properties PrincipalsAllowedToRetrieveManagedPassword
```



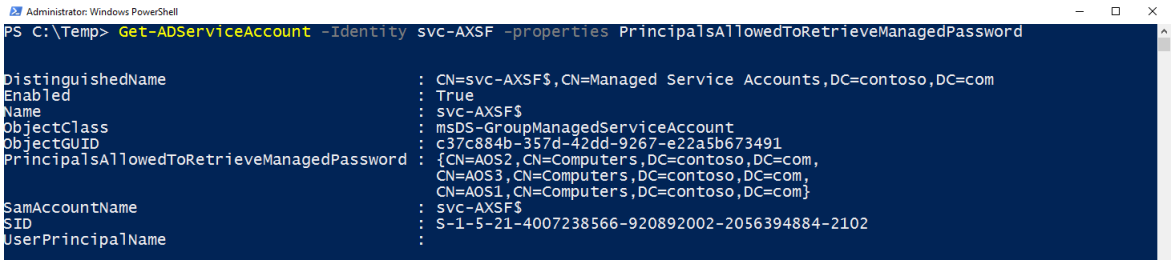
You see a list of the servers that have permission to retrieve the password for the **svc-AXFS$** gMSA. If you see a globally unique identifier (GUID) value for the server that was removed, ignore it.

10. Copy the list of principals from the result, and use them to edit or amend the following command. (Note that, because the **Set** command isn't additive, you must add all references back in.)

```
Set-ADServiceAccount -Identity svc-AXSF -PrincipalsAllowedToRetrieveManagedPassword
"CN=AOS1,CN=Computers,DC=contoso,DC=com","CN=AOS2,CN=Computers,DC=contoso,DC=com","CN=AOS3,CN=Compute
rs,DC=contoso,DC=com"
```



11. Run the original **Get** command to verify that the new AOS node was added back in. (Note in the example screenshot below you can see that AOS1 was added to the list of PrincipalsAllowedToRetrieveManagedPassword.)

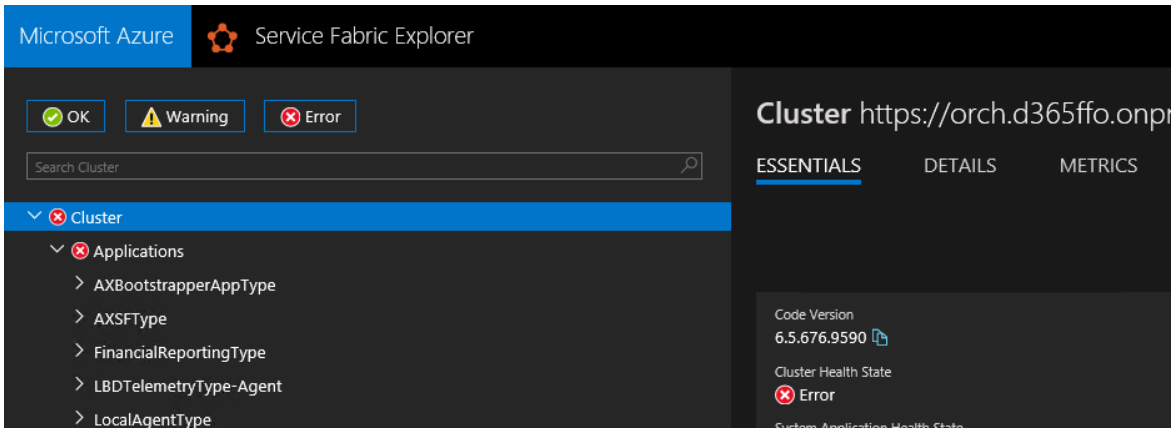


12. Run the following script to validate the VM setup.

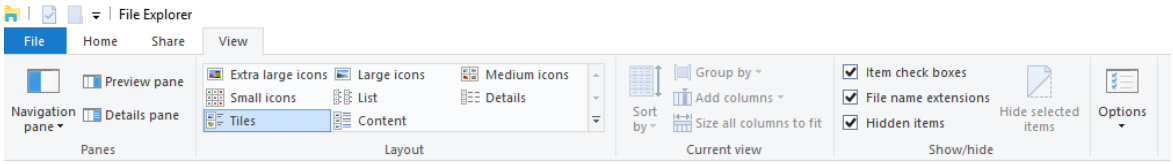
```
# If Remoting, execute
# .\Test-D365FOConfiguration-AllVMs.ps1 -ConfigurationFilePath .\ConfigTemplate.xml
.\Test-D365FOConfiguration.ps1
```

13. Before you continue, fix anything that fails as part of the validation script.

14. In Service Fabric Explorer, select **Cluster**, and make a note of the Microsoft Service Fabric cluster version. For this example, the cluster version is **6.5.676.9590**.



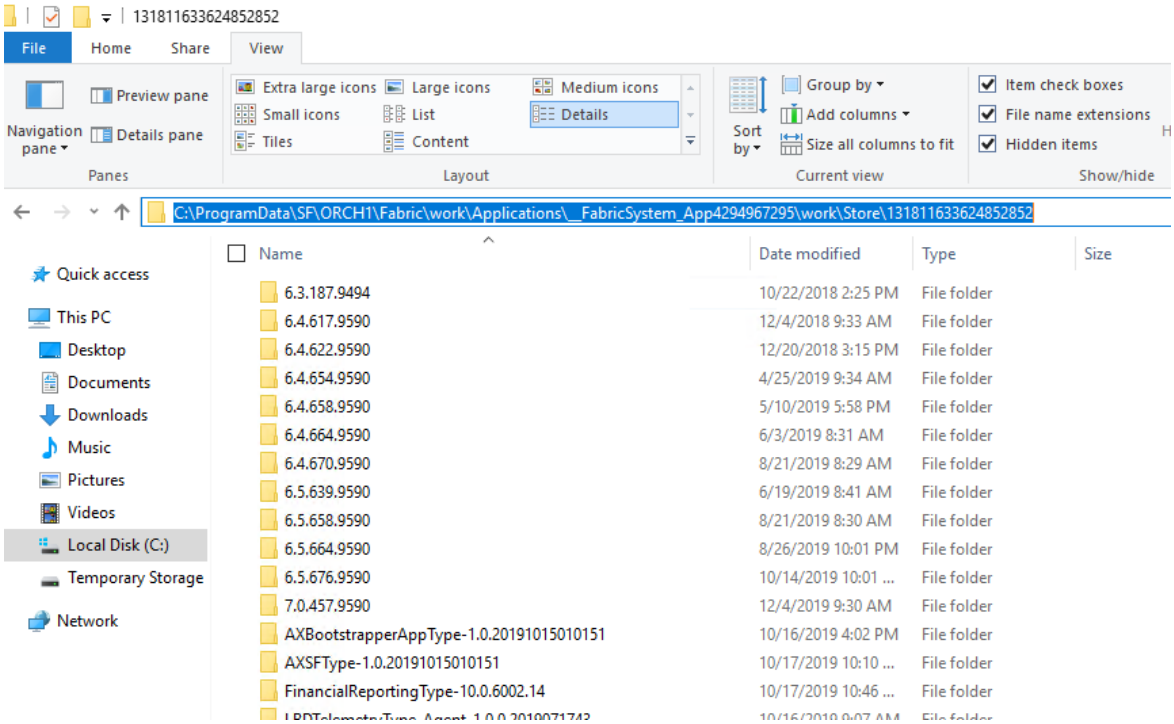
15. On one of the orchestrator nodes, open File Explorer. On the **View** tab, in the **Show/hide** group, make sure that the **File name extensions** and **Hidden items** check boxes are selected.



16. Expand drive C, and then drill down into the following folder. (Note that the bold parts of the path will vary, depending on the node name and setup.)

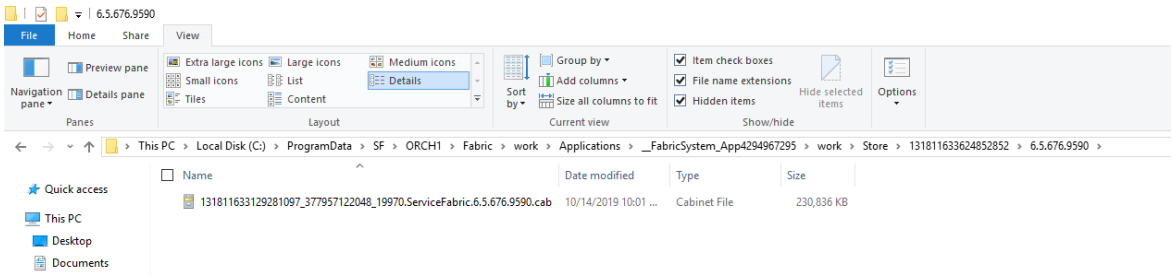
    C:\ProgramData\SF**ORCH1**\Fabric\work\Applications__FabricSystem_**App4294967295**\work\Store**131811633624852852**

    In the folder, you should see a list of folders for various versions of Service Fabric. Here is an example.

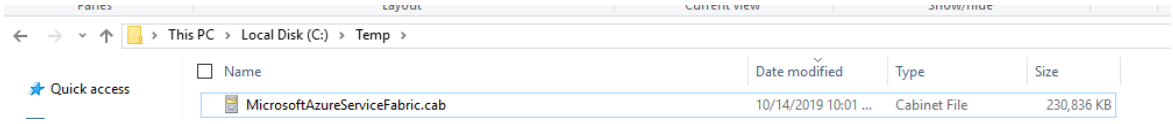


17. Open the folder that has the same name as the version of Microsoft Service Fabric cluster that you made a note of earlier. For this example, the folder is named **6.5.676.9590**.

18. In the folder, you should see a .cab file.



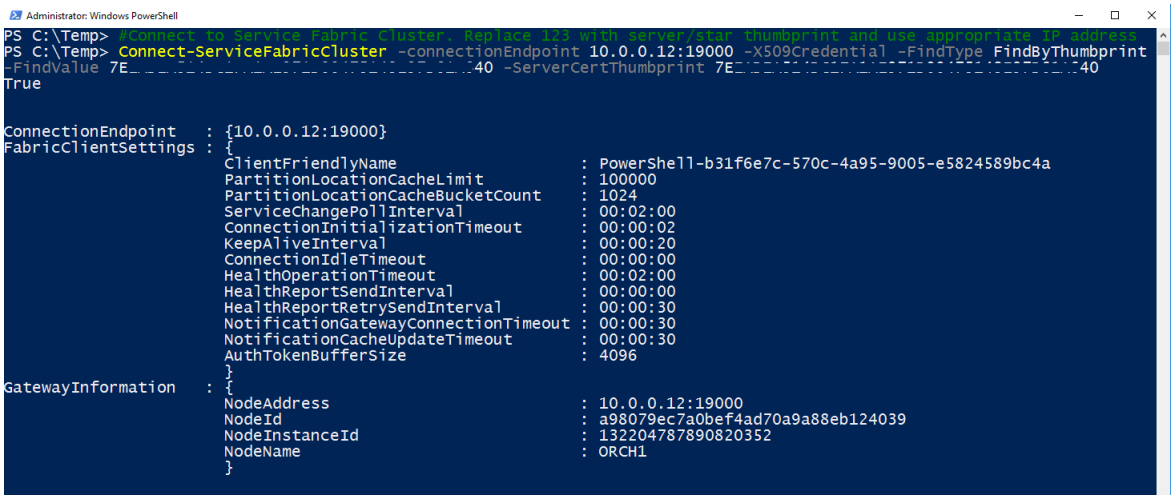
19. Copy the .cab file to C:\Temp, and rename the copied file **MicrosoftAzureServiceFabric.cab**. (If you don't have a Temp folder, create it.)



20. Open a Windows PowerShell Command Prompt windows as an admin.

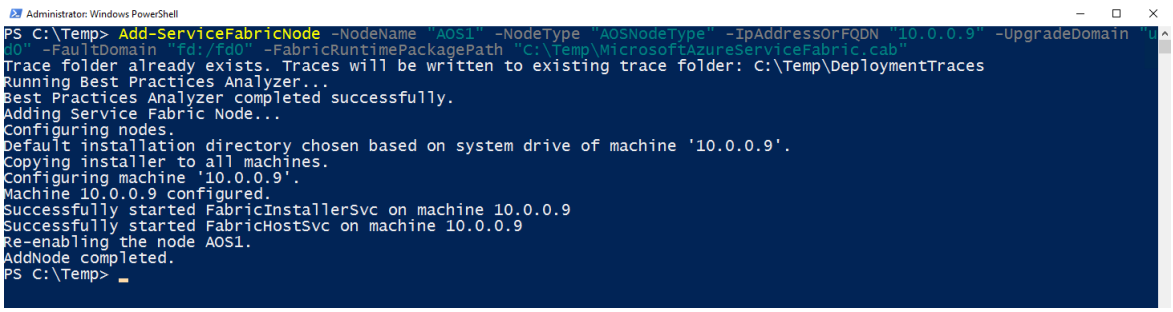
21. Run the following command to connect to your Service Fabric cluster. (Edit the command as you require.)

```
#Connect to Service Fabric Cluster. Replace 123 with server/star thumbprint and use appropriate IP
address
Connect-ServiceFabricCluster -connectionEndpoint 10.0.0.12:19000 -X509Credential -FindType
FindByThumbprint -FindValue 123 -ServerCertThumbprint 123
```



22. Run the following command to add the node back in. Before you run it, make the required edits to the **NodeName**, **IPAddress**, **UpgradeDomain**, and **FaultDomain** parameters. (If you're replacing an existing server, you should have made a note of the values earlier.)

```
Add-ServiceFabricNode -NodeName "AOS1" -NodeType "AOSNodeType" -IpAddressOrFQDN "10.0.0.9" -
UpgradeDomain "ud0" -FaultDomain "fd:/fd0" -FabricRuntimePackagePath
"C:\Temp\MicrosoftAzureServiceFabric.cab"
```



23. After the node has been added back in, return to Service Fabric Explorer, and view the application

deployment status. Several minutes will be required before all the AOS applications are restored (**AXBootstrapperAppType**, **AXSFType**, **RTGatewayAppType**, and **LBDTelemetryType-<envname>** or **MonitoringAgentAppType**) are pushed out again and installed on the node.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Reuse the same AD FS instance for multiple environments

2/18/2021 • 2 minutes to read • Edit Online

This topic explains how to use the same instance of Active Directory Federation Services (AD FS) in multiple Microsoft Dynamics 365 Finance + Operations (on-premises) environments.

Setup

> **IMPORTANT**
>
> This procedure assumes that you've previously configured AD FS for one environment by following the instructions in the Set up and deploy on-premises environments content. It also assumes that that environment is running without any issues.

1. In AD FS Manager, go to **AD FS** > **Application groups**, and open **Microsoft Dynamics 365 for Operations On-premises**.
2. In the **Native application** section, follow these steps:
   a. Open **Microsoft Dynamics 365 for Operations On-premises - Native application**, and add the redirect URI of the new environment (`https://ax.contoso.com/namespaces/AXSF`).
   b. Open **Microsoft Dynamics 365 for Operations On-premises - Financial Reporting - Native application**, and add the redirect URI of the new environment (`https://ax.contoso.com/FinancialReporting/ApplicationService/soap/`).
3. In the **Web API** section, follow these steps:
   a. Open **Microsoft Dynamics 365 for Operations On-premises - Web API**, and add two entries for the redirect URI of the new environment (`https://ax.contoso.com/namespaces/AXSF` and `https://ax.contoso.com`).
   b. Open **Microsoft Dynamics 365 for Operations On-premises - Financial Reporting Web API**, and add the redirect URI of the new environment (`https://ax.contoso.com/FinancialReporting`).
4. Optional: In the **Server** section, open **Microsoft Dynamics 365 for Operations On-premises - Retail**, and add the redirect URI of the new environment (`https://ax.contoso.com/namespaces/AXSF/`).
5. Optional: Configure the warehouse mobile app for the new environment by following the instructions in Configure the Warehousing app for on-premises deployments again. Remember to use the URI of the new environment (`https://ax.contoso.com`) as the **Resource URL** value.

   > **NOTE**
   >
   > No additional configuration is required for the workflow and retail designer applications.

6. Verify that you can reach the OpenID metadata endpoint (`https://<adfs-dns-name>/adfs/.well-known/openid-configuration`) from the **AOS** and **MR** nodes in your new environment. If you're using self-signed certificates, you might have to import the AD FS Secure Sockets Layer (SSL) certificate into the Trusted Root Certification Authorities store of each node.
7. When you deploy the new environment from Microsoft Dynamics Lifecycle Services (LCS) and are

specifying the deployment configuration, make sure that you use the same AD FS OpenID metadata endpoint and AD FS OpenID connect client IDs that you specified for the previous environment.

AD FS Microsoft 365 compatibility

2/18/2021 • 2 minutes to read • Edit Online

IMPORTANT

This feature is only fully supported starting with application update 10.0.8, Platform update 32, and LocalAgent 2.2.0. For details, see Partial support.

This topic explains how to use the same instance of Active Directory Federation Services (AD FS) for a Dynamics 365 Finance + Operations (on-premises) environment and for Microsoft 365.

Existing deployments

1. Download the new local agent version from Microsoft Dynamics Lifecycle Services (LCS). It should be version 2.2.0 or later.
2. Download the new version of the local agent configuration file, because it has additional configuration that is required for this functionality.
3. Modify the new local agent configuration file, and set the **office365AdfsCompatibility** value to **True**.
4. Run the following command to uninstall the old local agent version from your cluster.

```
.\LocalAgentCLI.exe Cleanup '<path of localagent-config.json>'
```

5. Run the following command to install the new local agent version.

```
.\LocalAgentCLI.exe Install '<path of localagent-config.json>'
```

6. Perform any servicing operation with Platform update 28 or later to make the new configuration available.
7. After servicing is completed, run the following script.

```
.\Reset-DatabaseUsers.ps1 -DatabaseServer '<FQDN of the SQL server>' -DatabaseName '<AX database
name>'
```

IMPORTANT

If you skip this step, the primary admin user won't be able to sign in.

8. Use Service Fabric Explorer to Restart applications (such as AOS).
9. Verify that you are able to sign in to the product with the system administrator user that was specified during deployment.
10. Download the newest version of the infrastructure scripts from the LCS Shared asset library.
11. Copy the Reset-SID.ps1 script from the downloaded infrastructure scripts folder into one of your AOS machines.

12. Execute the Reset-Sid.ps1 script.

```
.\Reset-SID.ps1 -AxsfCodePath
'C:\ProgramData\SF\AOS_13\Fabric\work\Applications\AXSFType_App184\AXSF.Code.1.0.20190902'
```

New deployments

1. Follow the instructions for installing the local agent in the "Configure a connector and install an on-premises local agent" section of Set up and deploy on-premises environments. However, before you actually install the local agent, complete step 2 of this procedure.
2. Modify the local agent configuration file, and set the **office365AdfsCompatibility** value to **True**.
3. Continue to follow the instructions in the "Configure a connector and install an on-premises local agent" section of Set up and deploy on-premises environments, and deploy a base version that runs Platform update 28 or later. If there is no base version that runs Platform update 28 or later, deploy the latest base version that is available. Then service it so that Platform update 28 is deployed on top.

Partial support

For partial support, it is necessary to have Local Agent 2.2.0 or later installed and to update the service with Platform update 28 or later.

With partial support, authentication against the Financial Reporting service is not supported.

Local agent pre-deployment and post-deployment scripts

2/18/2021 • 2 minutes to read • Edit Online

Local agent 2.3.0 supports the execution of pre-deployment and post-deployment scripts. Therefore, customers can now set up Microsoft Windows PowerShell scripts that are run before and after the environment is deployed. This feature applies to deployments and redeployments, and also to servicing operations.

To make this feature available, you must create a Scripts folder in the agent file share. To run a pre-deployment script, create a **PreDeployment.ps1** file in the Scripts folder. To run a post-deployment script, create a **PostDeployment.ps1** file. The following examples show the folder structure:

- \\\fileserver\agent\scripts\PreDeployment.ps1
- \\\fileserver\agent\scripts\PostDeployment.ps1

If these files don't exist, the deployment continues as usual. The following examples show the new deployment flow.

Deployment or redeployment:

1. Get unhealthy modules. In this step, the health of existing services is obtained to find which are unhealthy. This step applies only to redeployment scenarios.
2. Clean up modules. In this step, the services are removed and the contents of the **wp** folder are deleted. This step applies only to redeployment scenarios.
3. Link download artifacts. In this step, download, extraction, and processing of artifacts from Microsoft Dynamics Lifecycle Services (LCS) takes place.
4. Pre-deployment script. In this step the **PreDeployment.ps1** script is executed (if it exists).
5. Set up modules. In this step, the new services are deployed.
6. Post-Deployment script. In this step the **PostDeployment.ps1** script is executed (if it exists).

Servicing:

1. Prepare for servicing. In this step, the package is prepared in LCS and gets downloaded to the environment.
2. Clean up modules. In this step, the services are removed and the contents of the **wp** folder are deleted.
3. Link download artifacts. In this step, extraction and processing of previously downloaded artifacts from LCS takes place.
4. Pre-deployment script. In this step the **PreDeployment.ps1** script is executed (if it exists).
5. Set up modules. In this step, the new services are deployed.
6. Post-Deployment script. In this step the **PostDeployment.ps1** script is executed (if it exists).

> **NOTE**
> The pre-deployment and post-deployment scripts can contain anything. The code that the scripts run is solely the customer's responsibility. The local agent just invokes the scripts.

Customizations

The default time-out for script execution is 30 minutes. To change this value, modify the localagent-config.json file, and then reinstall the local agent by using the modified file. The following attribute defines time-out value

and must be set as shown here. (The code that is shown here is part of the **LocalAgent** component in the file.)

```
"powershellScriptRunner": {
    "timeoutMinutes": {
        "value": "30"
        }
    }
```

Logging

The outputs and error messages from the scripts are written, as .log and .err files, into the Logs folder in the Scripts folder. If a script times out, only an error message is logged. This error message has a time-out message. No other outputs are logged in this situation.

Execution of the scripts is also logged as Event Tracing for Windows (ETW) events. You can view these events in Event Viewer. If a script produces any errors, an error event is logged, but deployment continues as usual.

Update the local agent

2/18/2021 • 5 minutes to read • Edit Online

This topic explains how to update the local agent. The latest version of the local agent is version 2.6.0, which was released in October 2020.

LOCAL AGENT VERSION	CAPABILITY
Null	This initial version deploys Platform update 8.
1.0.0	This version enables the Reconfigure feature for failed deployments.
1.1.0	This version enables the Reconfigure feature for successful deployments, enables multi-model package deployments, and deploys Platform update 8 and 11.
2.0.0	This version enables servicing flows and deploys Platform update 12.
2.1.0	This version enables two-phased servicing where **Preparation** and **Update** are two separate steps.
2.1.1	This version fixes an issue that occurs when the download fails and the LCS Maintain button is not available. Additional changes include updates to Azure storage libraries to improve communication with Azure storage and enable TLS 1.2.
2.1.2	This version contains updated Azure dependencies for improved download stability and logic to correctly evaluate if files are downloaded. This fixes an issue where files are fully downloaded, but the logic would still consider them as missing a few bytes and therefore fail the download.
2.2.0	This version fixes locked dlls during cleanup and enables prerequisites for supporting Active Directory Federation Services (AD FS) that also is used for Microsoft 365.
2.3.0	This version adds support for pre- and post-deployment scripts.
2.3.1	This version fixes orchestration service crashes that may occur during clean up on some environments. Deploying version 10.0.5 with Platform update 29 or earlier requires the use of pre-deployment scripts for automatic updating of FinancialReportingDeployer.exe.config. For more information, see Troubleshoot on-premises deployments.
2.4.0	This version fixes a deployment issue and upgrades the runtime of the local agent.

LOCAL AGENT VERSION	CAPABILITY
2.5.0	This version updates dependencies and fixes a cleanup bug.
2.6.0	This version upgrades the Service Fabric SDK, fixes a bug with refresh state, and increases the application provisioning timeout.

What's new in local agent 2.6.0

- Local agent 2.6.0 uptakes a new Service Fabric SDK and runtime.
- This release fixes a bug where, if refresh state is triggered when the environment is stuck in the Downloading phase, the environment would automatically move to a deployed state without updating the environment. In this situation, the refresh state will mark the Downloading phase as failed.
- The timeout for provisioning an application has been increased.

> **IMPORTANT**
> This release is only compatible with 7.x Service Fabric clusters.

What's new in local agent 2.5.0

- Local agent 2.5.0 uptakes new versions of various dependencies. The main changes are Service Fabric and Entity Framework.
- This release also fixes a bug where, if cleanup fails without cleaning up any services, subsequent reattempts always fail during cleanup.

What's new in local agent 2.4.0

- Local agent 2.4.0 now requires .NET Framework 4.8 to uptake the newest changes from Lifecycle Services (LCS). Be sure to run the latest Infrastructure Scripts available in LCS to meet the newest requirements.
- This release also fixes an issue where the deployment of the AXService would fail in slower environments due to a hard-coded timeout.

What's new in local agent 2.3.0

- Local agent 2.3.0 enables the execution of custom pre- and post- deployment scripts.
- It fixes the problem introduced in 2.2.0 with regard to deploying older platform updates.
- This release removes the monitoring agent and introduces a new service called LBDTelemetry, which will be used to install the ETWManifests.

> **IMPORTANT**
> This release requires that a new local agent configuration file be downloaded from LCS. Refer to the Troubleshoot on-premises deployments topic if you encounter problems.

What's new in local agent 2.1.0

- Local agent 2.1.0 enables the two-phased servicing where **Environment preparation** and **Environment update** are two distinct steps and explicit actions. This reduces the total downtime customers must take when applying updates to their on-premises environments by preparing upfront and allowing users to use

the environment during preparation and then communicating the downtime when the actual update environment action is triggered.

What's new in local agent 2.0.0

- Local agent 2.0.0 can deploy Platform update 12.
- It enables the Reconfigure feature until the first deployment of Platform update 12 succeeds.
- It disables the Reconfigure feature on the first successful deployment of Platform update 12. After deployment succeeds, you can use the regular update experience to update the environment.

> **NOTE**
>
> Local agent 2.0.0 **cannot** deploy Platform update 8 and Platform update 11. You must have version 1.1.0 to deploy those platform updates.

Download the latest local agent and configuration from LCS

> **NOTE**
>
> If you require an older version of the local agent for your current deployments, download it from the Asset library in Microsoft Dynamics Lifecycle Services (LCS). To download Local agent version 1.1.0, go to **Shared Asset Library -> Model** and click on Dynamics 365 for Finance and Operations on-premises - Local agent v1.1.0**.
>
> You must have version 2.0.0 or later to deploy Platform update 12 and complete update flows.

1. In LCS, select **Project settings** > **On-prem connectors**.
2. Select the connector to your environment, and then select **Edit**.
3. On the menu on the left side of the page, select **Setup host infrastructure**, and then select **Download agent installer**.

   You must now verify that the zip file that is downloaded and unblocked.
4. Go to the zip file, right-click it, and then select **Properties**.
5. In the **Properties** dialog box, select **Unblock**, and then select **Apply**.
6. On the **Configure agent** tab, select **Download configurations** to download the localagent-config.json configuration file.

Update the local agent

1. Copy the zip file and the localagent-config.json file into one of the **Orchestrator** nodes, such as **c:\DynamicsAgent** in the **Orch1** virtual machine (VM).
2. Unzip the agent installer to C:\DynamicsAgent\LocalAgent.
3. Copy the localagent-config.json file to C:\DynamicsAgent\LocalAgent.
4. In a **Command Prompt** window, go to C:\DynamicsAgent\LocalAgent, and run the following command.

```
LocalAgentCLI.exe Cleanup <path of localagent-config.json>
```

> **NOTE**
>
> You must use the current agent's binaries to clean up the agent. If you don't have the current agent's binaries, you can delete the local agent application from Service Fabric Explorer.

5. Press any key to exit the cleanup operation.
6. Verify that the local agent has been successfully cleaned up by looking in Service Fabric Explorer and making sure that there are no apps in the **Deployed Applications** section in the **Orchestrator** nodes.
7. After the local agent is successfully cleaned up, run the following command.

```
LocalAgentCLI.exe Install <path of localagent-config.json>
```

8. After the local agent is successfully installed, go back to your on-premises connector in LCS.
9. On the **Validate setup** tab, select **Message agent** to test LCS connectivity to your new local agent.

Troubleshoot on-premises deployments

2/18/2021 • 51 minutes to read • Edit Online

This topic provides troubleshooting information for deployments of Microsoft Dynamics 365 Finance + Operations (on-premises).

Access Service Fabric Explorer

You can access Service Fabric Explorer in a web browser by using the default address, `https://sf.d365ffo.onprem.contoso.com:19080`.

To verify the address, note the value that was used in the "Create DNS zones and add A records" section of the appropriate setup and deployment topic for your environment:

- Platform update 12
- Platform update 8 and Platform update 11

You can access the site only if the client certificate is in cert:\CurrentUser\My on the machine that you're accessing the site on. (In Certificate Manger, go to **Certificates - Current User** > **Personal** > **Certificates**.) When you access the site, select the client certificate when you're prompted.

Monitor the deployment

Identify the primary orchestrator

To determine the machine that is the primary instance for stateful services such as a local agent, in Service Fabric Explorer, expand **Cluster** > **Applications** > **<*intended application example*> LocalAgentType** > **fabric:/LocalAgent/OrchestrationService** > **(GUID)**.

The primary node is shown. For stateless services or the remaining applications, you must check all the nodes.

Note the following points:

- OrchestrationService orchestrates the deployment and servicing actions for Finance + Operations.
- ArtifactsManager downloads files from Microsoft Azure cloud storage to the local agent file share. It also unzips the files into the required format.

Review the orchestrator event logs

From the primary OrchestrationService orchestrator machine, in Event Viewer, go to **Applications and Services Logs** > **Microsoft** > **Dynamics** > **AX-LocalAgent**.

> **NOTE**
>
> To view the full error message, you must select the **Details** tab.

The following modules must be installed:

- Common
- ReportingServices
- AOS
- FinancialReporting

The following commands must be run:

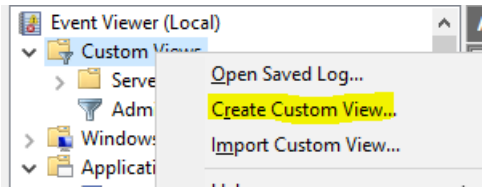
- **Setup**
- **Dvt** – This command runs a deployment verification test.
- **Cleanup** – This command is used to service and delete an environment.

The following folders contain additional information:

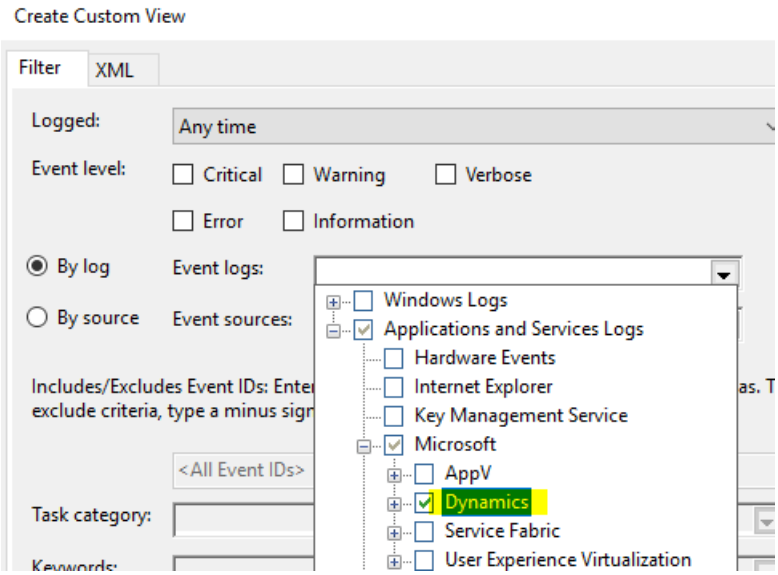
- AX-SetupModuleEvents
- AX-SetupInfrastructureEvents
- AX-BridgeService

To review Microsoft Dynamics entries in Event Viewer, follow these steps.

1. In Event Viewer, right-click **Custom Views**, and then select **Create Custom View**.



2. In the **Event logs** field, select **Dynamics**.



> **NOTE**
> Also look at **Administrative Events** in **Custom Views**.

Service Fabric Explorer

Note the state of the cluster, application, and nodes. For information about how to access Service Fabric Explorer, see Access Service Fabric Explorer.

Error: "Partition is below target replica or instance count"

You might receive the following error:

```
Partition is below target replica or instance count
```

This error isn't a root error. It indicates that the status of each node isn't ready. For AXSFType (AOS), the status might still be **InBuild**.

On the machines that are related to the error message, use Event Viewer to view the latest activity.

AXSFType

If a status of **InBuild** is shown for AXSFType (AOS), review the DB Sync status and other events from Application Object Server (AOS) machines.

To diagnose errors, use Event Viewer to review the following event logs:

- Applications and Services Logs > Microsoft > Dynamics > AX-DatabaseSynchronize
- Custom Views > Administrative Events

Error: "'ExtractInstallerService failed to extract' C:\Users\dynuser.CONTOSO\AppData\Local\Temp\1blssblh.w0n\FabricInstallerService.Code\FabricClient.dll"

You might receive the following error:

> "ExtractInstallerService failed to extract" C:\Users\dynuser.CONTOSO\AppData\Local\Temp\1blssblh.w0n\FabricInstallerService.Code\FabricClient.dll.

If you receive this error, download the latest version of Azure Service Fabric. Note that the user name and path in the error message vary, depending on your environment.

Service Fabric logs

You can find more details about Service Fabric applications in the log files at C:\ProgramData\SF\<OrchestratorMachineName>\Fabric\work\Applications\LocalAgentType_App<N>\log.

Lifecycle Services

Note the current deployment status for the environment in Microsoft Dynamics Lifecycle Services (LCS).

A time-out error occurs when a Service Fabric cluster is created

Run Test-D365FOConfiguration.ps1 as noted in the "Set up a standalone Service Fabric cluster" section of the appropriate setup and deployment topic for your environment. Note any errors.

- Platform update 12
- Platform update 8 and Platform update 11

Be sure to complete these steps:

- Verify that the Service Fabric Server client certificate exists in the LocalMachine store on all Service Fabric nodes.
- Verify that the Service Fabric Server certificate has the access control list (ACL) for Network Service on all Service Fabric nodes.
- Review the antivirus exclusions that are noted in Environment setup.

A time-out error occurs while you're waiting for Installer Service to be completed for machine x.x.x.x

Only one node type is supported for each Internet Protocol (IP) address (that is, for each machine). Check whether the nodes are being reused on the same machine. For example, AOS and ORCH must not be on the same machine, and ConfigTemplate.xml must be correctly defined.

Remove a specific application

We recommend that you use LCS to remove or clean up deployments. However, you can also use Service Fabric Explorer to remove an application as you require.

In Service Fabric Explorer, go to **Application node** > **Applications** > **MonitoringAgentAppType-Agent**. Select the ellipsis button (...) next to **fabric:/Agent-Monitoring**, and delete the application. Enter the full name of the application to confirm the deletion of the application.

You can also remove MonitoringAgentAppType-Agent by selecting the ellipsis button and then selecting **Unprovision Type**. Enter the full name to confirm the removal of the application.

Remove all applications from Service Fabric

The following script removes and unprovisions all Service Fabric applications except LocalAgent and the monitoring agent for LocalAgent. You must run this script on an orchestrator virtual machine (VM).

```
$applicationNamesToIgnore = @('fabric:/LocalAgent', 'fabric:/Agent-Monitoring', 'fabric:/Agent-
LBDTelemetry')
$applicationTypeNamesToIgnore = @('MonitoringAgentAppType-Agent', 'LocalAgentType', 'LBDTelemetryType-
Agent')

Get-ServiceFabricApplication | `
    Where-Object { $_.ApplicationName -notin $applicationNamesToIgnore } | `
    Remove-ServiceFabricApplication -Force

Get-ServiceFabricApplicationType | `
    Where-Object { $_.ApplicationTypeName -notin $applicationTypeNamesToIgnore } | `
    Unregister-ServiceFabricApplicationType -Force
```

Remove Service Fabric

To completely remove the Service Fabric cluster, follow these steps.

1. Run the following command.

   ```
   .\RemoveServiceFabricCluster.ps1 -ClusterConfigFilePath .\ClusterConfig.json
   ```

2. If an error occurs, remove a specific node on the cluster by using the **CleanFabric.ps1** command. You can find this command in C:\Program Files\Microsoft Service Fabric\bin\fabric\fabric.code.

3. Remove the **C:\ProgramData\SF** folder, if you're using the default location. Otherwise, remove the specified folder.

   If you receive an "Access denied" error, restart Microsoft Windows PowerShell or the machine.

Clean up an existing environment and redeploy

To clean up an existing environment and redeploy, follow these steps.

1. In LCS, open the project, and then, in the **Environments** section, delete the deployment.

   The applications should start to disappear from Service Fabric Explorer in the environment. This process will take one to two minutes.

2. Access the orchestrator machine that contains LocalAgentCLI.exe, and follow these steps:

   a. Run the local agent cleanup.

      ```
      .\LocalAgentCLI.exe Cleanup '<path of localagent-config.json>'
      ```

   b. Remove Service Fabric.

```
.\RemoveServiceFabricCluster.ps1 -ClusterConfigFilePath '<path of ClusterConfig.json>'
```

c. If any nodes fail, run the **CleanFabric.ps1** command. You can find this command in C:\Program Files\Microsoft Service Fabric\bin\fabric\fabric.code.

d. Remove the **C:\ProgramData\SF** folder on all Service Fabric nodes.

If you receive an "Access denied" error, restart the machine, and try again.

3. Remove or update certificates as required.

Remove old certificates from all AOS, BI, ORCH, and DC nodes.

- The certificates exist in the following certificate stores: Cert:\CurrentUser\My\, Cert:\LocalMachine\My, and Cert:\LocalMachine\Root.
- If the setup of Microsoft SQL Server will be modified, remove the SQL Server certificates.
- If the settings for Active Directory Federation Services (AD FS) will be modified, remove the AD FS certificate.

4. Update the following configuration files as required:

- ConfigTemplate.xml
- ClusterConfig.json

For information about how to correctly fill in the fields in the templates, see the appropriate setup and deployment topic for your environment:

- Platform update 12
- Platform update 8 and Platform update 11

5. In LCS, open the project, and update the LCS on-premises connector as required.

a. Re-create the LCS on-premises connector for the environment, or edit the settings of an existing connector.

To obtain easy-to-copy values for LCS, use the .\Get-AgentConfiguration.ps1 script.

b. Download the latest local agent configuration, localagent-config.json.

6. Deploy again by following the instructions in the appropriate setup and deployment topic for the environment:

- Platform update 12
- Platform update 8 and Platform update 11.

Find the local agent values that are used

You can find local agent values in Service Fabric Explorer. Go to **Cluster** > **Applications** > **LocalAgentType** > **fabric:/LocalAgent**, and then select **Details**.

Alternatively, run the following Windows PowerShell command.

```
.\Get-AgentConfiguration.ps1 -ConfigurationFilePath .\ConfigTemplate.xml
```

Install, upgrade, or uninstall a local agent

For information about how to update the local agent, see Update the local agent.

You can also use the following upgrade and uninstallation commands:

```
LocalAgentCLI.exe Install <path of localagent-config.json>
LocalAgentCLI.exe Cleanup <path of localagent-config.json>
```

> **NOTE**
>
> The **Cleanup** command doesn't remove any files that were put in the file share. The file share can be reused.

An error occurs when local agent services are started

When local agent services are started, you might receive the following error:

> Could not load file or assembly 'Lcs.DeploymentAgent.Proxy.Contract, Version=1.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35' or one of its dependencies.

This error means that strong name verification is turned on. You can turn off this verification by using Configure-PreReqs.ps1. To validate that strong name verification is no longer turned on, run Test-D365FOConfiguration.ps1.

A "Validation in progress" message is shown for several minutes in LCS

Follow these steps to troubleshoot general issues with local agent validation.

1. Run **Configure-PreReqs.ps1** on all orchestrator machines to configure the machines correctly.
2. Verify that the Test-D365FOConfiguration.ps1 script passes on all the orchestrator machines.
3. Verify that the installation of LocalAgentCLI.exe is successfully completed.
4. In Service Fabric Explorer, verify that all the applications are healthy.
5. If the applications aren't healthy, find the primary node for the service that is failing. In Event Viewer, look for events in the following locations:
   - Custom Views > Administrative Events
   - Applications and Services Log > Microsoft > Dynamics > AX-LocalAgent

Local agent errors

Issue

Error: When you install the local agent, you receive the following error.

```
LocalAgentCLI.exe Error: 0 : Exception System.InvalidOperationException: unable to get settings for
telemetry setup component
    at LBDTelemetryCommon.LBDTelemetrySetupManager.GetComponentSettings()
    at LBDTelemetryCommon.LBDTelemetrySetupManager.ApplyParameters()
    at LocalAgentCLI.Program.Main(String[] args)
Press any key to exit
```

Reason: You're trying to install local agent version 2.3.0 or later, but the localagent-config.json file that you're using isn't up to date.

Steps: Get the new version of the localagent-config.json file from LCS by following the instructions in the "Configure a connector and install an on-premises local agent" section of Set up and deploy on-premises

environments.

You can also manually add the following values in the **components** section of the localagent-config.json file.

```
{
    "name": "LBDTelemetry",
    "placementCriteria": "(IsOrchestratorEnabled == True)",
    "parameters": {
        "applicationPackagePath": {
            "value": "Applications\\LBDTelemetry"
        }
    }
},
```

Issue

Error: You might receive the following errors:

> Unable to process commands

> Unable to get the channel information

> RunAsync failed due to an unhandled exception causing the host process to crash:
> System.ArgumentNullException: Value cannot be null. Parameter name: certificate

Reason: These errors can occur because the certificate that is specified for the OnPremLocalAgent certificate either isn't valid or isn't correctly configured for the tenant.

Steps: Follow these steps to resolve the error.

1. Run **Test-D365FOConfiguration.ps1** on all orchestrator nodes to make sure that all checks pass.
2. Verify that the certificate that is specified in the local agent configuration is correct.
   - Make sure that the thumbprint that you specify in LCS and in the ConfigTemplate.xml file has no special characters.
   - The certificate should be the same certificate that is specified in the following section in infrastructure\ConfigTemplate.xml.

     ```
     <Certificate type="Orchestrator" exportable="true" generateSelfSignedCert="true">
         <Name>OnPremLocalAgent</Name>
         <Thumbprint></Thumbprint>
         <ProtectTo></ProtectTo>
     </Certificate>
     ```
3. Make sure that the same certificate that is specified in the local agent configuration in LCS was used to complete the steps in the "Configure LCS connectivity for the tenant" section of the appropriate setup and deployment topic for your environment:
   - Platform update 12
   - Platform update 8 and Platform update 11
4. Uninstall the local agent.
5. Specify the correct certificate in the local agent configuration, and download the configuration file again.
6. Install the local agent again by using the new configuration file.

Error

Error: During servicing, you receive an "Unable to download asset" error, and the details state, "The credentials supplied to the package were not recognized."

Reason: The ACL wasn't correctly defined on certificates.

Steps:

Check whether ACL was removed from client certificate on orchestrator machines. Run the .\Test-D365FOConfiguration.ps1 script on orchestrator machines, and verify the ACL.

To resolve the error, run the .\Set-CertificateAcls.ps1 script to reset the ACLs.

Error

Error:

> Access to the path '\...\agent\assets\StandAloneSetup-76308-1.zip' is denied.

Reason: The file share that is specified in the local agent configuration isn't valid.

Steps: Follow these steps to resolve the error.

1. Verify that the specified share exists.
2. Verify that the local agent user has full permission on the share. The local agent user is the Domain Name System (DNS) name that is specified in the following section in ConfigTemplate.xml.

   ```
   <ADServiceAccount type="gMSA" name="svc-LocalAgent$" refName="gmsaLocalAgent">
       <DNSHostName>svc-LocalAgent.d365ffo.onprem.contoso.com</DNSHostName>
   </ADServiceAccount>
   ```

3. Make sure that the "Set up file storage" section of the appropriate setup and deployment topic for your environment is completed:
   - Platform update 12
   - Platform update 8 and Platform update 11
4. Uninstall the local agent.
5. Specify the correct file share in the local agent configuration, and download the configuration file again.
6. Install the local agent again by using the new configuration file.

Error

Error: When you do a servicing operation, you receive the following error:

> Unable to get extract setup folder for command

Reason: The file share has been removed or changed.

Steps: To see what the file share is set to, open Microsoft SQL Server Management Studio, and run the following query on the orchestrator database:

```
select * from OrchestratorCommandArtifact where CommandId = 'xxx'
```

Error

Error:

> Login failed for user 'D365\svc-LocalAgent$'. Reason: Could not find a login matching the name provided. [CLIENT: 10.0.2.23]

Reason: The local agent user can't connect to the orchestrator database. This issue can occur because users have been deleted and then re-created in Active Directory Domain Services (AD DS). Therefore, the security identifier (SID) of the user has changed, and any access that was given to the user for the SQL Server instance or the database no longer works.

Steps: Follow these steps to resolve the error.

1. Run the following script on the SQL Server instance.

   ```
   .\Initialize-Database.ps1 -ConfigurationFilePath .\ConfigTemplate.xml -ComponentName Orchestrator
   ```

   This script creates an empty orchestrator database, if an empty database doesn't already exist. It then adds the local agent user to the database and gives it db_owner permission.

   After the correct permissions are provided, the application should automatically go to a healthy state.

2. If any settings, such as the fully qualified domain name (FQDN) of the SQL Server instance, the database name, or the local agent user, were provided incorrectly in LCS, change the settings, and then reinstall the local agent.

If the preceding steps don't resolve the error, manually remove the local agent user from the SQL Server instance and the database, and then rerun the Initialize-Database script.

If you re-create a user in AD DS, remember that the SID will change. In this case, remove the previous SID for the user, and add a new SID.

Error

Error:

> Unable to migrate database

Steps:

- Verify that you have access to the SQL Server listener.
- If you're doing testing, you can start over and use an empty orchestrator database.

Issue

When you performing the Configure the databases procedure, if the SQL Server instance is a named instance, use the **-DatabaseServer [FQDN/Instancename]** parameter.

Issue

The local agent user can't connect to the SQL Server instance or the database.

Steps: Follow these steps to resolve the error.

1. Delete the svc-LocalAgent user from the SQL Server primary node databases, and then remove the login from both servers.

2. Run the following scripts.

   ```
   .\Initialize-Database.ps1 -ConfigurationFilePath .\ConfigTemplate.xml -ComponentName Orchestrator
   .\Configure-Database.ps1 -ConfigurationFilePath .\ConfigTemplate.xml -ComponentName Orchestrator
   ```

> **IMPORTANT**
>
> These scripts don't work when an **always-on** setup is used. The database must first be created in the primary node and then replicated.

Error

Error:

> RunAsync failed due to an unhandled exception causing the host process to crash:
> System.Net.Http.HttpRequestException: An error occurred while sending the request. --->
> System.Net.WebException: The remote name could not be resolved: 'lcsapi.lcs.dynamics.com'

Reason: The local agent machines can't connect to lcsapi.lcs.dynamics.com. Review the AX-BridgeService event log for "The remote name could not be resolved: 'lcsapi.lcs.dynamics.com'."

Steps: Follow these steps to resolve the error.

1. Run **psping lcsapi.lcs.dynamics.com:80**.
2. If you don't receive a response from the preceding command, contact the IT department at your organization. Either the firewall is blocking access to lcsapi, or proxy issues are occurring.

```
lcsapi.lcs.dynamics.com:443
login.windows.net:443
uswelcs1lcm.queue.core.windows.net:443
www.office.com:443
login.microsoftonline.com:443
dc.services.visualstudio.com:443
uswelcs1lcm.blob.core.windows.net:443
uswedpl1catalog.blob.core.windows.net:443
```

Infrastructure scripts errors

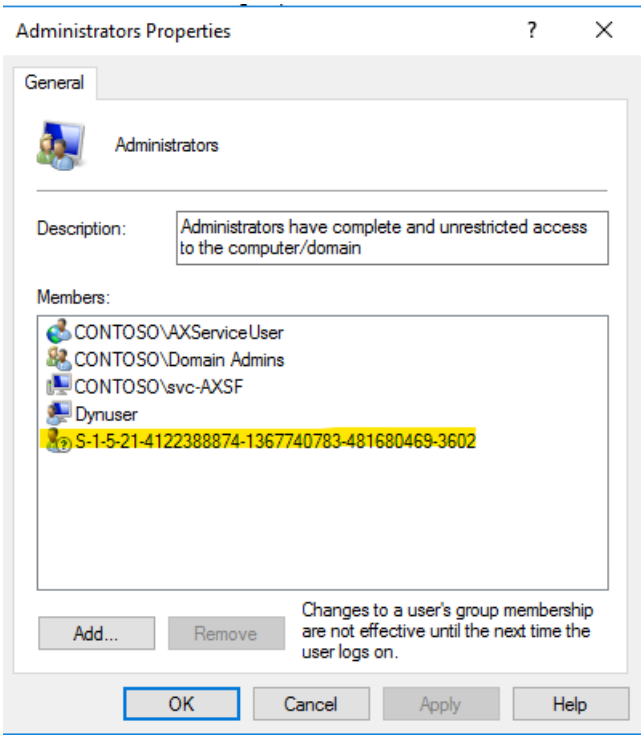
Issue

Error: When you run Test-D365FOConfiguration.ps1 or Test-D365FOConfiguration-AllVMs.ps1, you receive the message:

```
"Get-LocalGroupMember : Failed to compare two elements in the array.
At C:\Infrastructure\Scripts\Test-D365FOConfiguration.ps1:79 char:9
+         Get-LocalGroupMember -Group 'Administrators' | `
+         ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : NotSpecified: (:) [Get-LocalGroupMember], InvalidOperationException
    + FullyQualifiedErrorId : An unspecified error
occurred.,Microsoft.PowerShell.Commands.GetLocalGroupMemberCommand"
```

Reason: There is a bug in the PowerShell commandlet, Get-LocalGroupMember, which causes it to fail when there are entries that not valid.

Steps: On the machine where the script is failing, open **local users and groups**. Go to the administrators group and remove any entries that have an entry like the one highlighted in the following image.



Do this on all of the machines that receive this error. After the changes are complete, try running the script again.

Restart applications (such as AOS)

In Service Fabric, expand **Nodes** > **AOSx** > **fabric:/AXSF** > **AXSF** > **Code Packages** > **Code**. Select the ellipsis button (...), and then select **Restart**. When you're prompted, enter the code.

Upgrade Service Fabric

Service Fabric Explorer will show a message that resembles the following message:

> Unhealthy event: SourceId='System.UpgradeOrchestrationService', Property='ClusterVersionSupport', HealthState='Warning', ConsiderWarningAsError=false. The current cluster version 6.1.467.9494 support ends 5/30/2018 12:00:00 AM. Please view available upgrades using Get-ServiceFabricRegisteredClusterCodeVersion and upgrade using Start-ServiceFabricClusterUpgrade.

Because the minimum requirement is one Microsoft SQL Server Reporting Services (SSRS) node and one Management Reporter node, you must pass in a parameter to skip PreUpgradeSafetyCheck.

Follow these steps to upgrade Service Fabric in Windows PowerShell.

1. Connect to the Service Fabric cluster. In the following command, replace **123** with the server/star thumbprint, and use the appropriate IP address.

   ```
   Connect-ServiceFabricCluster -connectionEndpoint 10.0.0.12:19000 -X509Credential -FindType
   FindByThumbprint -FindValue 123 -ServerCertThumbprint 123
   ```

2. Get the latest version that was downloaded.

   ```
   Get-ServiceFabricRegisteredClusterCodeVersion
   ```

3. Start the upgrade. For **-CodePackageVersion**, enter the latest version.

> **NOTE**
>
> **-UpgradeReplicaSetCheckTimeout** is used to skip PreUpgradeSafetyCheck for SSRS and Management Reporter. For more information, see Service Fabric service upgrade not working. You might also want to use **-UpgradeDomainTimeoutSec 600 -UpgradeTimeoutSec 1800**. For more information, see Application upgrade parameters.

```
Start-ServiceFabricClusterUpgrade -Code -CodePackageVersion 6.1.472.9494 -Monitored -FailureAction
Rollback -UpgradeReplicaSetCheckTimeout 30
```

4. Get the upgrade status.

```
Get-ServiceFabricClusterUpgrade
```

For more information, see Troubleshoot application upgrades.

To learn when a new Service Fabric release comes out, see the Azure Service Fabric team blog.

If you receive a warning in Service Fabric Explorer after you upgrade, make a note of the node, and then restart by expanding **Nodes** > **AOSx** > **fabric:/AXSF** > **AXSF** > **Code Packages** > **Code**. Select the ellipsis button (...), and then select **Restart**.

Error: "Unable to load DLL 'FabricClient.dll'"

If you receive an error that states, "Unable to load DLL 'FabricClient.dll'," close and restart Windows PowerShell. If the error persists, restart the machine.

What cluster ID should be used in the agent configuration?

The cluster ID can be any globally unique identifier (GUID). This GUID is used for tracking purposes.

Encryption errors

Some examples of encryption errors include "AXBootstrapperAppType," "Bootstrapper," "AXDiagnostics," "RTGatewayAppType," "Gateway potential failure related," and "Microsoft.D365.Gateways.ClusterGateway.exe."

You might receive one of these errors if the data encipherment certificate that was used to encrypt the AOS account password wasn't installed on the machine. This certificate might be in the certificates (local computer), or the provider type might be incorrect.

To resolve the error, validate the credentials.json file. Verify that the text is correctly decrypted by entering the following command (on AOS1).

```
Invoke-ServiceFabricDecryptText -CipherText 'longstring' -StoreLocation LocalMachine | Set-Clipboard
```

This error can also occur if the '' parameter isn't defined in the ApplicationManifest file. To determine whether this parameter is defined, in Event Viewer, go to **Custom Views** > **Administrative Events**, and verify the following information:

- The encrypt credentials for the credentials.json file have the correct layout/structure. For more information, see the "Encrypt credentials" section of the appropriate setup and deployment topic for your environment:

    - Platform update 12
    - Platform update 8 and Platform update 11
- A closing quotation mark appears at the end of the line or on the next line.

In Event Viewer, under **Custom Views** > **Administrative Events**, note any errors in the **Microsoft-Service Fabric** source category.

Properties to create a DataEncryption certificate

Use the following properties to create the DataEncryption certificate:

- **Is self-signed certificate** – Enable this parameter only when you're using self-signed certificates.
- **Certificate purposes** – Enable all purposes for this certificate.
- **Signature algorithm** – Specify **sha256RSA**.
- **Signature hash algorithm** – Specify **sha256**.
- **Issuer** – Specify **CN = DataEncryptionCertificate**.
- **Public Key** – Specify **RSA (2048 bits)**.
- **Thumbprint algorithm** – Specify **sha1**.

> **WARNING**
> Don't use self-signed certificates in production environments. Instead, use certificates that are issued by certificate authorities.

The certificate and private key that should be used for decryption can't be found (0x8009200C)

If you're missing a certificate and ACL, or if you have the wrong thumbprint entry, check for special characters, and look for thumbprints in C:\ProgramData\SF\<AOSMachineName>\Fabric\work\Applications\AXBootstrapperAppType_App<N>\log\ConfigureCertificates-<timestamp>.txt.

You can also validate the encrypted text by using the following command.

```
Invoke-ServiceFabricDecryptText -CipherText 'longstring' -StoreLocation LocalMachine | Set-Clipboard
```

If you receive the message, "Cannot find the certificate and private key to use for decryption," verify the axdataenciphermentcert and svc-AXSF$ AXServiceUser ACLs.

If the credentials.json file has changed, the action you should take depends on the status of the environment in LCS.

- If your environment appears to be deployed in LCS, do the following:
    1. Go to your environment page and select **Maintain**.
    2. Select **Update settings**.
    3. Do not change any settings. Select **Prepare**.
    4. After a few minutes your environment will be prepared and you can select **Deploy**.
- If your environment is in a failed state in LCS, do the following:
    1. Select **Retry**. The new Credentials.json file will be used during the retry operation.

If none of the preceding solutions work, follow these steps.

1. Verify that the domain name and Active Directory account names that are specified in the ConfigTemplate.xml file are correct.
2. Verify that the thumbprints that are specified in the ConfigTemplate.xml file are correct if the certificate wasn't generated by using the scripts that are provided.
3. Verify that the certificate thumbprints that are specified in LCS are correct, and that they match the thumbprints that are specified in ConfigTemplate.xml. Make sure that there are no special characters. You can run **.\Get-DeploymentSettings.ps1** to obtain the thumbprints in an easy-to-copy manner.
4. If the certificates aren't self-generated, make sure that the provider names match for the following certificate types:
   - **ServiceFabricEncryption type:** Microsoft Enhanced Cryptographic Provider v1.0
   - **All other certificate types:** Microsoft Enhanced RSA and AES Cryptographic Provider
5. Verify that the Set-CertificateAcls.ps1 and Test-D365FOConfiguration.ps1 scripts were successfully run on all Service Fabric machines.
6. Verify that the credentials.json file exists, and that the entries are decrypted to correct values.

   On one of the AOS machines, run the following command to verify that the data encryption certificate is correct.

   ```
   Invoke-ServiceFabricDecryptText '<encrypted string>' -StoreLocation LocalMachine
   ```
7. If any of the certificates must be changed, or if the configuration was incorrect, follow these steps:
   a. Edit the **ConfigTemplate.xml** file so that it has the correct values.
   b. Run all the setup scripts and the **Test-D365FOConfiguration** script.
8. In LCS, reconfigure the environment.

Gateway fails to deploy

Issue: You receive the following error in the event viewer logs.

```
Message Module aos failed Detail System.InvalidOperationException: Gateway app and Bootstrapper app are not
healthy at AOSSetupHybridCloud.Program.Main(String[] args)
at System.AppDomain._nExecuteAssembly(RuntimeAssembly assembly, String[] args)
at System.AppDomain.ExecuteAssembly(String assemblyFile, String[] args)
at System.AppDomain.ExecuteAssembly(String assemblyFile, String[] args)
at SetupCore.SetupManager.LaunchProcessInAppDomain(String startupExe, String workingDir, String
currentFolder, String[] moduleArgs)
at SetupCore.SetupManager.<>c__DisplayClass12_1.<InvokeModules>b__6()
```

You also receive the following error message in the SFExplorer for the Gateway application.

```
'System.RA' reported Warning for property 'ReplicaOpenStatus'.
Replica had multiple failures during open on AOS_13. API call: IStatelessServiceInstance.Open(); Error =
System.InvalidOperationException (-2146233079)
Category does not exist.
   at System.Diagnostics.PerformanceCounterLib.CounterExists(String machine, String category, String
counter)
   at System.Diagnostics.PerformanceCounter.InitializeImpl()
   at System.Diagnostics.PerformanceCounter..ctor(String categoryName, String counterName, String
instanceName, Boolean readOnly)
   at System.Diagnostics.PerformanceCounter..ctor(String categoryName, String counterName, String
instanceName)
   at Microsoft.Dynamics.LBD.Gateways.ClusterGateway.Helpers.CpuPerfCounter..ctor()
   at Microsoft.Dynamics.LBD.Gateways.ClusterGateway.GzipContentDelegatingHandler..ctor()
   at Microsoft.Dynamics.LBD.Gateways.ClusterGateway.ClusterGateway.ConfigureApp(IAppBuilder appBuilder)
   at Microsoft.Owin.Hosting.Engine.HostingEngine.Start(StartContext context)
   at Microsoft.Dynamics.LBD.Gateways.Common.OwinCommunicationListener.b__9_0()
   at Microsoft.D365.ServicePlatform.Context.ServiceContext.Activity.d__10`2.MoveNext()
```

Reason: The pointers to the performance counter that the gateway needs may be corrupt.

Resolution: Run **lodctr /R** in a Command Prompt window that you open as administrator in all AOS nodes where the gateway is unhealthy. If you receive an error message that states that the performance counters can't be rebuilt, try to run the command again.

Management Reporter

Additional logging can be done by registering providers. Download ETWManifest.zip to the **primary** orchestrator machine, and then run the following commands. To determine which machine is the primary instance, in Service Fabric Explorer, expand **Cluster** > **Applications** > **LocalAgentType** > **fabric:/LocalAgent/OrchestrationService** > **(GUID)**.

> **NOTE**
>
> If results in Event Viewer don't appear correct (for example, if words are truncated), get the latest manifest and .dll files. To get the latest manifest and .dll files, go to the WP folder in the agent file share. This share was created in the "Set up file storage" section of the appropriate setup and deployment topic for your environment:
>
> - Platform update 12
> - Platform update 8 and Platform update 11
>
> **Example:** [*Agent Share*]\wp\[*Deployment name*]\StandaloneSetup-...\Apps\ETWManifests

```
.\RegisterETW.ps1 -ManifestsAndDll @{"C:\Files\ETWManifest\Microsoft.Dynamics.Reporting.Instrumentation.man"
= "C:\Files\ETWManifest\Microsoft.Dynamics.Reporting.Instrumentation.dll"}
```

If you must unregister providers, use the following command.

```
.\RegisterETW.ps1 -ManifestsAndDll @{"C:\Files\ETWManifest\Microsoft.Dynamics.Reporting.Instrumentation.man"
= "C:\Files\ETWManifest\Microsoft.Dynamics.Reporting.Instrumentation.dll"} -Unregister
```

After providers are registered, additional details about the new deployment are logged in Event Viewer, at **Applications and Services Logs** > **Microsoft** > **Dynamics**. The following folders will be shown:

- MR-Logger
- MR-Sql

To see the new folders, you must close and reopen Event Viewer. To see additional details, you must deploy an

environment again.

Could not load file or assembly EntityFramework

Issue: You are running Local Agent version 2.3.1 or later and you received the following stacktrace in the event logs while deploying a package that contains Platform update 29 or earlier:

```
System.Reflection.TargetInvocationException: Exception has been thrown by the target of an invocation. --->
System.Reflection.TargetInvocationException: Exception has been thrown by the target of an invocation. --->
System.IO.FileNotFoundException: Could not load file or assembly 'EntityFramework, Version=6.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089' or one of its dependencies. The system cannot find the
file
specified. at Microsoft.Dynamics.Integration.Service.Utility.AdapterProvider.RefreshAdapters()
--- End of inner exception stack trace ---
```

Resolution: Use TSG_UpdateFRDeployerConfig.ps1. For more information, see TSG_UpdateFRDeployerConfig.ps1.

Unable to deploy Financial Reporting Service

Issue: You are unable to finish deployment of Platform update 26 and later for Financial Reporting because the following error is in the application log for Service Fabric.

```
Application: FinancialReportingDeployer.exe Framework Version: v4.0.30319
Description: The process was terminated due to an unhandled exception.
Exception  Info: System.DllNotFoundException at
Microsoft.Cloud.InstrumentationFramework.NativeIfxInterop.InitializeIfxFromCloudAgentConfigureSamplingAndTra
cing_x64(System.String,  System.String, UInt32, UInt32, Boolean) at
Microsoft.Cloud.InstrumentationFramework.IfxInitializer.IfxInitialize(System.String,
Microsoft.Cloud.InstrumentationFramework.InstrumentationSpecification,
Microsoft.Cloud.InstrumentationFramework.AuditSpecification) at
Microsoft.Dynamics.Performance.Logger.IfxLogger..cctor() Exception Info:  System.TypeInitializationException
at
Microsoft.Dynamics.Performance.Logger.IfxLogger..ctor(System.String,
Microsoft.Dynamics.Performance.Logger.IfxLoggerOptions) at
Microsoft.Dynamics.Performance.Logger.IfxLoggerProvider.CreateLogger(System.String)  at
Microsoft.Extensions.Logging.Logger..ctor(Microsoft.Extensions.Logging.LoggerFactory,  System.String) at
```

Reason: The Microsoft Visual C++ Redistributable Package for Visual Studio 2013 was not correctly installed or is corrupt in some or all of the MR nodes.

Steps: Re-run the installation of the Microsoft Visual C++ Redistributable Package for Visual Studio 2013.

An error occurs while AddAXDatabaseChangeTracking is running

If you receive an error while you run AddAXDatabaseChangeTracking at Microsoft.Dynamics.Performance.Deployment.FinancialReportingDeployer.Utility.InvokeCmdletAndValidateSuccess(DeploymentCmdlet cmdlet), verify that the full path is correct. An example of a full path is **ax.d365ffo.onprem.contoso.com**.

The error might also occur because of an issue with the star certificate. For example, the remote certificate CN=*.d365ffo.onprem.contoso.com has a name that isn't valid or that doesn't match the host, ax.d365ffo.onprem.contoso.com.

Run the initialize database script, and validate that databases have correct users

If you receive only the AddAXDatabaseChangeTracking event, try to reach the MetadataService service for Finance + Operations by going to `https://ax.d365ffo.contoso.com/namespaces/AXSF/services/MetadataService`.

Next, check the certificates of the service in the wif.config file. To find the file, sign in to one of the AOS machines, and then, in Task Manager, find **AxService.exe**. Right-click, and select **Open file location**. In the wif.config file, you should see three thumbprints. Note the following requirements for these thumbprints:

- They must be different.
- They must be in this order:
    1. Financialreporting thumbprint
    2. ReportingService thumbprint
    3. SessionAuthentication thumbprint

If the thumbprints don't meet both these requirements, you must redeploy from LCS by using correct thumbprints.

The remote name can't be resolved

Error:

> The remote name could not be resolved: 'x.d365fo.onprem.contoso.com' / There was no endpoint listening at `https://x.d365fo.onprem.contoso.com/namespaces/AXSF/services/MetadataService` that could accept the message.

Reason: This issue is often caused by an incorrect address or SOAP action.

Steps: Verify that the address can be reached, by manually opening the URL. For more details, see the "InnerException" text in the Event Viewer, if it's present.

Error on ImportDefaultReports

If Management Reporter reports are checked out during deployment, the deployment will fail. To see whether reports are checked out, run the following **select** statements on the FinancialReporting database.

```
select checkedoutto, * from Reporting.ControlReport where checkedoutto is not null
select checkedoutto, * from Reporting.ControlRowMaster where checkedoutto is not null
select checkedoutto, * from Reporting.ControlColumnMaster where checkedoutto is not null
```

To learn which user has objects checked out, you can run the following **select** statement.

```
select * from Reporting.SecurityUser where UserID = ''
```

To resolve this issue manually, update the following tables, and set **checkedoutto** to **null** by using the following commands.

```
update Reporting.ControlReport set checkedoutto = null where checkedoutto is not null
update Reporting.ControlRowMaster set checkedoutto = null where checkedoutto is not null
update Reporting.ControlColumnMaster set checkedoutto = null where checkedoutto is not null
```

axdbadmin can't connect to the database server SQL-LS.contoso.com

Reason: The user doesn't have permission to connect to the AXDB database.

Steps:

1. Remove the axdbadmin user from the database, if it already exists.
2. In the **ConfigTemplate.xml** file, specify the user name that must be added to the AXDB database.

```
<Security>
    <User refName="axdbadmin" type="SqlUser" userName="axdbadmin" />
</Security>
```

3. Run the initialize database script again to add the axdbadmin user.

Unable to resolve the xPath value

In the expected behavior, the following **xPath** value can't be resolved:

[TopologyInstance/CustomizationGroup[@name='ServiceConfiguration']/Group[@name='AOSServicePrincipalUser']/Customizations/Customization[@fieldName='PrincipalUserAccountPassword']/@selectedValue

Therefore, the fact that the **xPath** value can't be resolved isn't an issue. The **xPath** value looks for AOS runtime user information. However, because of integrated security, that information isn't required. The fact that the **xPath** value can't be resolved is communicated in case the failure must be investigated for another reason.

AD FS

The sign-in page doesn't redirect you

The sign-in page might not redirect you but continues to prompt for credentials. Alternatively, you might be redirected but receive the following message:

> An error occurred. Contact your administrator for more information.

In these cases, you can follow these steps to resolve the issue:

- Add the AD FS link to the list of trusted sites.
- Add the Dynamics 365 link to the list of trusted sites.
- Add a trailing slash (/), and see whether the behavior changes.

Verify the AD FS Manager by going to **ADFS** > **Application groups**. Double-click **Microsoft Dynamics 365 for Operations on-premises**. Then, under **Native application**, double-click **Microsoft Dynamics 365 for Operations on-premises - Native application**.

Note the **Redirect URI** value. It should match the DNS forward lookup zone for Finance + Operations.

Error: "Could not establish trust relationship for the SSL/TLS secure channel"

If you receive an error that states, "Could not establish trust relationship for the SSL/TLS secure channel," follow these steps.

1. In Service Fabric, go to **Cluster** > **Applications** > **AXSFType** > **fabric:/AXSF**, and then, on the **Details** tab, scroll down and note the URLs for **Aad_AADMetadataLocationFormat** and **Aad_FederationMetadataLocation**.
2. Browse to those URLs from AOS.
3. On the AOS machine, in Event Viewer, go to **Applications and Services Logs** > **Microsoft** > **Dynamics** > **AX-SystemRuntime** for details.
4. Verify that the AD FS certificate is trusted:
    a. Verify the AD FS certificate. On the AD FS machine, in Server Manager, go to **Tools** > **AD FS Management**.
    b. Expand **AD FS** > **Service** > **Certificates**, and make a note of the certificates. For example, one certificate might be dc1.contoso.com.

c. On the AOS machine, in the Microsoft Management Console Certificates snap-in, go to **Certificates (Local Computer)** > **Trusted Root Certification Authorities** > **Certificates**, and verify that the AD FS certificate is listed.

If you receive a message that states that the site isn't secure, you haven't added your Secure Sockets Layer (SSL) certificate for AD FS to the Trusted Root Certification Authorities store.

You can't connect to the remote server in some locations

You might not be able to connect to the remote server at the following places:

- System.Net.HttpWebRequest.GetResponse()
- System.Xml.XmlDownloadManager.GetNonFileStream(Uri uri, ICredentials credentials, IWebProxy proxy, RequestCachePolicy cachePolicy)
- System.Xml.XmlUrlResolver.GetEntity(Uri absoluteUri, String role, Type ofObjectToReturn)

In this case, go to the C:\ProgramData\SF\AOS_1\Fabric\work\Applications\AXSFType_App35\log folder where you receive the error, and note the out file. The out file contains the following information:

> System.Net.WebException: Unable to connect to the remote server --->
>
> System.Net.Sockets.SocketException: A connection attempt failed because the connected party did not properly respond after a period of time, or established connection failed because connected host has failed to respond x.x.x.x:443

You can also use Psping to try to reach the remote server. For information about Psping, see Psping.

Redirect sign-in questions and issues

If you're having issues when you sign-in, in Service Fabric Explorer, verify that the **Provisioning_AdminPrincipalName** and **Provisioning_AdminIdentityProvider** values are valid. Here is an example:

- **Provisioning_AdminPrincipalName**: `AXServiceUser@contoso.com`
- **Provisioning_AdminIdentityProvider**: `https://DC1.contoso.com/adfs`

If the values aren't valid, you won't be able to proceed, and you must redeploy from LCS.

If you used Reset-DatabaseUsers.ps1, you must restart the Dynamics Service before your changes take effect. If you still have sign-in issues, in the USERINFO table, note the **NETWORKDOMAIN** and **NETWORKALIAS** values. Here is an example:

- **NETWORKDOMAIN:** `https://DC1.contoso.com/adfs`
- **NETWORKALIAS:** `AXServiceUser@contoso.com`
- **IDENTITYPROVIDER:** This should match the **NETWORKDOMAIN** value.

On the AD FS machine, in Server Manager, go to **Tools** > **AD FS Management** > **Service**. Right-click **Service and Edit Federation Service Properties**. The **Federation Service identifier** value should match the **USERINFO.NETWORKDOMAIN** value, and it should have **https** in the URL (for example, `https://DC1.contoso.com/adfs`).

On the AD FS machine, in Event Viewer, go to **Applications and Services Logs** > **AD FS** > **Admin**, and make a note of any errors.

Fiddler

Fiddler can be used for additional debugging. For in-depth information about Fiddler, see AD FS 2.0: How to Use Fiddler Web Debugger to Analyze a WS-Federation Passive Sign-In and Cracking the AD FS Token from another AD FS Claims Provider.

The following sections provide focused debugging steps for claims that are returned to Microsoft Dynamics.

Repo/capture

1. Open Fiddler, go to **Tools** > **Options** > **HTTPS**, and select **Decrypt HTTPS traffic**.
2. Start to capture traffic (the shortcut key is F12). You can verify that that traffic is being captured by looking at the lower left of the tool.
3. Open an InPrivate instance of Internet Explorer or an Incognito instance of Chrome.
4. Open Finance + Operations (for example, `https://ax.d365ffo.onprem.contoso.com/namespaces/AXSF/`).
5. Sign in by using the USERINFO.NETWORKALIAS account and password.
6. After you're signed in, stop Fiddler from capturing traffic.

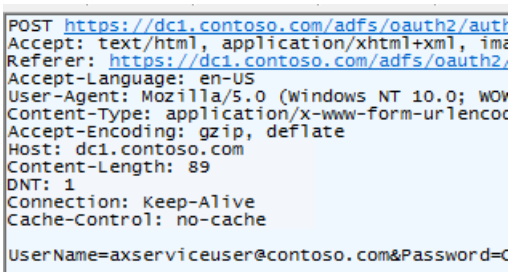
Analyze

In the right pane of Fiddler, notice that a horizontal divider separates the request from the response. Unlike a network trace, where you typically get one frame for a request and another frame for a response, Fiddler provides one frame that contains both the request and the response.

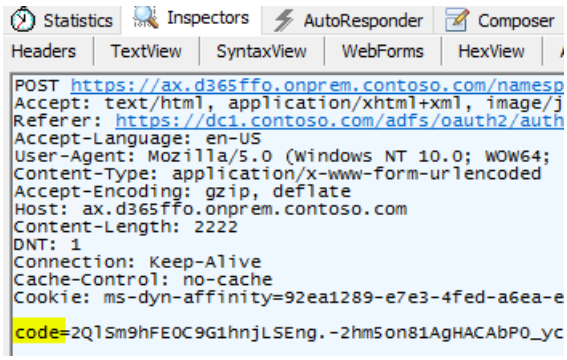
1. In Fiddler, in the upper-right corner, select **Inspectors** > **Raw**.
2. In the lower-right corner, select **Cookies**.
3. Do a search for **MSISAuth**.
4. Select the row that has a result of **200** for the AD FS host.
5. Look above the row that you just selected to find a row that has a result of **302**. Select the row.

   You should see the AD FS URL, host, user name, and password.

   > **IMPORTANT**
   >
   > For privacy, you might have to scrub personally identifiable information.



6. Select the next row that has a result of **302**. The URL should be **.../namespaces/AXSF/**.
7. Find the code line that is shown on that row.



8. Copy the value of code line after the equal sign (=).
9. Go to https://www.base64decode.org/, and paste the code that you just copied.

10. In the **Source charset** field, select **ASCII**.
11. Select **Decode**.
12. Copy the results, and follow these steps:
    - Make sure that the **upn** value matches the user name.
    - Make sure that the **unique_name** value is the Active Directory user that is being tested.
    - Go to **Active Directory Users and Computers** > **domain** > **Users**, and make sure that this user is being tested.

Sign-in issues

If you or other users experience sign-in issues, in Service Fabric Explorer, verify that the **Provisioning_AdminPrincipalName** and **Provisioning_AdminIdentityProvider** values are valid. If the values are valid, run the following command on the primary SQL Server machine.

```
.\Reset-DatabaseUsers.ps1
```

On each AOS machine, in Task Manager, select **AXService.exe**, and then select **End task**.

To verify that a user has been reset, run the following **select** query in the AXDB SQL database.

```
select SID, NETWORKDOMAIN, NETWORKALIAS, * from AXDB.dbo.USERINFO where id = 'admin'
```

> **NOTE**
> In an Azure Active Directory (Azure AD) environment (that is, an online environment), the SID is a hash of a network alias and a network domain. In an AD DS environment (that is, an on-premises environment), the SID is a hash of a network alias and an identify provider.

In some cases, you still might not be able to sign in, and you might receive the following error:

> You are not authorized to login with your current credentials. You will be redirected to the login page in a few seconds.

If this error occurs, follow these steps.

1. On the AD FS machine, go to **Server Manager** > **Tools** > **AD FS Management**.
2. Right-click **AD FS**, and then select **Edit Federation Service Properties**.
3. Make sure that the **Federation Service Identifier** value matches the **Userinfo.NetworkDomain** and **UserInfo.IdentityProvider** values.
4. On the AD FS machine, open Windows PowerShell, and run **Get-AdfsProperties**.
5. Make sure that the **IdTokenIssuer** value matches the **Federation Service Identifier** value from step 3, and also the **Provisioning_AdminIdentityProvider** value on the **fabric:/AXSF Details** tab at **Service Fabric Explorer** > **Cluster** > **Applications** > **AXSFType**.
6. In Service Fabric Explorer, verify that the **Provisioning_AdminPrincipalName** and **Provisioning_AdminIdentityProvider** values are valid.

If the preceding steps don't resolve the issue, see the AD FS section of this topic.

System.Data.SqlClient.SqlException (0x80131904) and

System.ComponentModel.Win32Exception (0x80004005)

You might receive one of the following errors:

> System.Data.SqlClient.SqlException (0x80131904): A connection was successfully established with the server, but then an error occurred during the sign-in process. (provider: SSL Provider, error: 0 - The certificate chain was issued by an authority that is not trusted.)

> System.ComponentModel.Win32Exception (0x80004005): The certificate chain was issued by an authority that is not trusted

In this case, either the certificates haven't been installed, or they haven't given access to the correct users. To resolve this error, add the public key SQL Server certificate to all the Service Fabric nodes.

Keyset doesn't exist

If you find that the keyset doesn't exist, scripts weren't run on all machines. Review and complete the "Set up VMs" section of the appropriate setup and deployment topic for your environment:

- Platform update 12
- Platform update 8 and Platform update 11

Copy the scripts in each folder to the VMs that correspond to the folder name.

Additionally, check the .csv file to verify that the correct domain is used.

Error: "RunAsync failed due to an unhandled FabricException causing replica to fault"

You might receive the following error:

> RunAsync failed due to an unhandled FabricException causing replica to fault: System.Fabric.FabricException: The first Fabric upgrade must specify both the code and config versions. Requested value: 0.0.0.0:

In this case, in the ClusterConfig.json file, change **diagnosticsStore** from a network share to a local path. For example, change **\\server\path** to a default value of **C:\ProgramData\SF\DiagnosticsStore**.

Service Fabric AOS node error during build: The execution time-out expired

Error:

> The timeout period elapsed prior to completion of the operation or the server is not responding.
> The statement has been terminated.

Only one AOS machine can run DB Sync at a time. You can safely ignore this error, because it means that one of the AOS VMs is running DB Sync. Therefore, the other VMs produce a warning that they can't run it. To verify that DB Sync is running, on the AOS VM that isn't producing warnings, in Event Viewer, go to **Applications and Services Log** > **Microsoft** > **Dynamics** > **AX-DatabaseSynchronize/Operational**.

Error: "RequireNonce is 'true' (default) but validationContext.Nonce is null"

You might receive the following error:

> RequireNonce is 'true' (default) but validationContext.Nonce is null

This error also appears as an HTTP error 500 in Internet Explorer after you sign in to the client. The nonce that is issued can't be validated if Internet Explorer is in Enhanced Security Configuration.

To sign in to the client, disable Enhanced Security Configuration for Internet Explorer via Server Manager.

Error: "Invalid algorithm specified / Cryptography"

If you receive an "Invalid algorithm specified / Cryptography" error, you must use the Microsoft Enhanced RSA and AES Cryptographic Provider. For more information, see the certificate requirements. Additionally, verify that the structure of the credentials.json file is correct.

If you must re-create the certificate by using the correct provider, follow these steps.

1. Create the certificate again by using the correct provider.
2. Change the **ConfigTemplate.xml** file.
3. Run the infrastructure scripts on all machines in the cluster, and make sure that the **Test-D365FOConfiguration.ps1** script passes.
4. Reconfigure the environment from LCS.

An "Unable to find certificate" error occurs when you run Test-D365FOConfiguration.ps1

If you receive an "Unable to find certificate" error when you run Test-D365FOConfiguration.ps1, check whether certificates or thumbprints are being combined for multiple purposes. For example, you will receive this error if the client certificate and the SessionAuthentication certificate are combined. We recommend that you not combine certificates. For more information, see the certificate requirements, and check the acl.csv file for **domain.com\user** versus **domain\user** (for example, NETBIOS structure).

The client and server can't communicate because they don't have a common algorithm

If the client and server can't communicate because they don't have a common algorithm, verify that the certificates that are created use the specified provider, as explained in the "Plan and acquire your certificates" section of the appropriate setup and deployment topic for your environment:

- Platform update 12
- Platform update 8 and Platform update 11

Find a list of group managed service accounts

To find a list of all groups and hosts, run the following command.

```
Get-ADServiceAccount -Identity svc-LocalAgent$ -Properties PrincipalsAllowedToRetrieveManagedPassword
```

AddCertToServicePrincipal script fails on Import-Module

Error:

> AddCertToServicePrincipal script failing on Import-Module : Could not load file or assembly

> 'Commands.Common.Graph.RBAC, Version=1.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35' or one of its dependencies. Strong name validation failed. (Exception from HRESULT: 0x8013141A) may have multiple versions of the same module installed.

Steps: To resolve this issue, follow these steps.

1. Run the following command in Windows PowerShell.

```
Uninstall-Module -Name AzureRM
Install-Module AzureRM
```

2. Close the Windows PowerShell window, and try to run the script again.

ReportingServicesSetup.exe error

Error:

> ReportingServicesSetup.exe Error: 0 : Application fabric:/ReportingService is not OK after 10 minutes
> Application: ReportingServicesBootstrapper.exe
> Framework Version: v4.0.30319
> Description: The process was terminated due to an unhandled exception.

Reason: If you receive this error, strong name validation is enabled in the Reporting server, but it should not be enabled.

Steps: To resolve this issue, run the **config-PreReq** script on the Reporting server machine.

The requested operation requires elevation

This issue occurs because AOS users aren't in the local administrator group, and User Account Control (UAC) hasn't been disabled correctly. To resolve the issue, follow these steps.

1. Add AOS users as local admins, as described in the "Join VMs to the domain" section of the appropriate setup and deployment topic for your environment:
   - Platform update 12
   - Platform update 8 and Platform update 11
2. Run the **Config-PreReq** script on all the AOS machines.
3. Make sure that the **Test-Configuration** script passes.
4. If UAC was changed, you must restart the machine before the changes take effect.

Files in use errors

If these "Files in use" errors occur, set up the exclusion rules that Service Fabric advises. For information, see Environment setup.

Apply deployable packages during deployment

Package deployment fails because of a "path too long" exception

Because of a 260-character limit in Microsoft Windows, deployment will fail if a package has a longer name, or if the on-premises share has the full FQDN path. If the character limit is exceeded, you receive the following error:

> System.IO.PathTooLongException: The specified path, file name, or both are too long. The fully qualified file

> name must be less than 260 characters, and the directory name must be less than 248 characters. at System.IO.PathHelper.GetFullPathName

To work around this issue, shorten the package name, and then apply the package again. Alternatively, shorten the overall length of the share path for the on-premises assets.

Package deployment fails because of a serialization error

During package deployment, you might receive the following error:

> Serialization version mismatch detect, make sure the runtime DLLs are in sync with the deployed metadata. Version of file 'XXX'. Version of DLL 'XXX'

In this case, the version of the environment where the package was developed might differ from the version of the environment that the package is being deployed in.

To work around this issue, keep the development or build environments on the same version as the deployed on-premises environment. You can confirm the package version by looking in the **Additional details** section in the Asset library where the package is uploaded. To fix the error, generate the package on a version that is the same as or earlier than the version that is deployed in the on-premises environment.

Package deployment fails because of dependencies on missing modules

If you try to apply a package that is missing dependent modules, package application will fail, and you will receive a message that resembles the following message:

> Package [dynamicsax-My_commonextension.7.0.4679.35176.nupkg has missing dependencies: [dynamicsax-demodatasuite;dynamicsax-financialreportingadaptors;dynamicsax-fleetmanagement;dynamicsax-fleetmanagementextension;dynamicsax-publicsectorformadaptor]]
>
> Package [dynamicsax-My_coreextension.7.0.4679.35176.nupkg has missing dependencies: [dynamicsax-demodatasuite;dynamicsax-financialreportingadaptors;dynamicsax-fiscalbooksformadaptor;dynamicsax-fleetmanagement;dynamicsax-fleetmanagementextension;dynamicsax-fleetmanagementunittests;dynamicsax-generalledgerformadaptor;dynamicsax-publicsectorformadaptor;dynamicsax-retailformadaptor]]
>
> Package [dynamicsax-My_uiextension.7.0.4679.35176.nupkg has missing dependencies: [dynamicsax-demodatasuite;dynamicsax-financialreportingadaptors;dynamicsax-fiscalbooksformadaptor;dynamicsax-fleetmanagement;dynamicsax-fleetmanagementextension;dynamicsax-fleetmanagementunittests]]

To confirm the issue and find the missing dependencies, in Event Viewer, open **Application and Services**, and then go to **Microsoft** > **Dynamics** > **AX-SetupModuleEvents** to view events that have missing modules. For example, one of the modules that is typically missing is ApplicationFoundationFormAdaptor.

To fix this issue and successfully apply the package, either add dependent modules, or remove modules that require dependent modules. To add dependent modules, you must include the dependencies when you build the package. To remove modules, you can use ModelUtil.exe to delete a module. For more information, see Export and import models.

Package deployment works in a one-box environment but not in the sandbox environment

A one-box environment might have all the modules installed, whereas the sandbox environment might have only the modules that are required in order to run your production environment. If the package that was built in the dev environment has a dependency on modules that are present in the one-box environment but not in the sandbox environment, the package won't work in the sandbox environment.

To resolve this issue, look at all the modules that you're dependent on, and make sure that you don't pull any farm adapter or any other module that isn't required in the production environment. The best practice is to take

the package from the build box.

An error occurs when you sign in to on-premises environments

- **Platform update 12:** Turn off the Skype integration by going to **System administration** > **Setup** > **Client performance options**. When you go to the app, append **?debug=true** to the URL, as shown in the following example: `https://ax.d365ffo.onprem.contoso.com/namespaces/AXSF/?debug=true`
- **Platform update 8 and Platform update 11:** A known issue for the Skype application programming interface (API) affects the ability to sign in to on-premises environments. Microsoft is investigating a resolution for this issue. To work around this issue, you can add **?debug=true** to the end of the URL, as shown in the following example: `https://ax.d365ffo.onprem.contoso.com/namespaces/AXSF/?debug=true`

The local agent stops working after the tenant for the project from LCS is changed

Follow these steps to configure the local agent with the updated tenant.

1. Uninstall the local agent.

   ```
   .\LocalAgentCLI.exe Cleanup <path of localagent-config.json>
   ```

2. Follow the steps in the "Configure LCS connectivity for the tenant" section of the appropriate setup and deployment topic for your environment:

   - Platform update 12
   - Platform update 8 and Platform update 11

3. Create a new LCS connector in the new tenant.

4. Download the **local-agent.config** file.

5. Install the local agent.

   ```
   .\LocalAgentCLI.exe Install <path of localagent-config.json>
   ```

Additional deployments (for example, two sandbox deployments, or a sandbox and production deployment)

You will receive the following error when you deploy an additional environment:

> .\Publish-ADFSApplicationGroup.ps1 -HostUrl `https://ax.d365ffo.onprem.contoso.com` New-AdfsApplicationGroup : MSIS9908: The application group identifier must be unique in AD FS configuration.

You can skip or modify the following sections in the deployment instructions.

Plan and acquire your certificates (as documented for Platform update 12 or Platform update 8 and Platform update 11)

- You must use the same on-premises local agent certificate.
- You can use same star certificates (AOS SSL and Service Fabric).
- The remaining certificates should probably differ from the certificates for the existing environment.

Download setup scripts from LCS (as documented for Platform update 12 or Platform update 8 and Platform update 11)

- The scripts that are downloaded should be copied into a new folder.

Set up a standalone Service Fabric cluster (as documented for Platform update 12 or Platform update 8 and Platform update 11)

- The scripts that are downloaded should be copied into a new folder.

Configure LCS connectivity for the tenant (as documented for Platform update 12 or Platform update 8 and Platform update 11)

- You must complete this task only one time for the tenant.

Configure AD FS (as documented for Platform update 12 or Platform update 8 and Platform update 11)

- Configure AD FS according to the Reuse the same AD FS instance for multiple environments guide.

Redeploy SSRS reports

Version 10.0.13 or later

Run the following command against your business data database (AXDB):

```
 UPDATE SF.synclog SET STATE=5, SyncStepName = 'ReportSyncstarted' WHERE CODEPACKAGEVERSION in (SELECT
TOP(1) CODEPACKAGEVERSION from SF.SYNCLOG ORDER BY CREATIONDATE DESC)
```

Version 10.0.12 or earlier

Run the following command against your business data database (AXDB):

```
    DELETE FROM SF.synclog WHERE CODEPACKAGEVERSION in (SELECT TOP(1) CODEPACKAGEVERSION from SF.SYNCLOG
ORDER BY CODEPACKAGEVERSION DESC)
```

> **NOTE**
>
> If you are using version 10.0.12 or earlier, a full database synchronization will be executed.

After running the command, restart one of your AOS nodes through Service Fabric Explorer or restart the VM that the node is running on.

Add axdbadmin to tempdb after a SQL Server restart via a stored procedure

When SQL Server is restarted, the tempdb database is re-created. Therefore, there will be missing permissions. Run the following script to create a stored procedure on the master database.

```
\-----
USE [master]
GO
CREATE procedure [dbo].[CREATETEMPDBPERMISSIONS] as begin exec ('USE tempdb; declare @dbaccesscount int;
select @dbaccesscount = COUNT(*) from master..syslogins where name = ''axdbadmin''; if (@dbaccesscount <> 0)
exec sp_grantdbaccess ''axdbadmin''; ALTER USER [axdbadmin] WITH DEFAULT_SCHEMA=dbo; EXEC sp_addrolemember
N''db_datareader'', N''axdbadmin''; EXEC sp_addrolemember N''db_datawriter'', N''axdbadmin''; EXEC
sp_addrolemember N''db_ddladmin'', N''axdbadmin''; exec sp_grantdbaccess ''contoso\svc-AXSF$''; ALTER USER
[contoso\svc-AXSF$] WITH DEFAULT_SCHEMA=dbo; EXEC sp_addrolemember N''db_datareader'', N''contoso\svc-
AXSF$''; EXEC sp_addrolemember N''db_datawriter'', N''contoso\svc-AXSF$''; EXEC sp_addrolemember
N''db_ddladmin'', N''contoso\svc-AXSF$'';') end
GO
EXEC sp_procoption N'[dbo].[CREATETEMPDBPERMISSIONS]', 'startup', '1'
\-----
```

Error: "Updates to existing credential with KeyId '<key>' is not allowed"

You might receive the following error:

> Updates to existing credential with KeyId '<key>' is not allowed

The steps to resolve this issue depend on whether you have only an on-premises project, or whether you have both an online project and an on-premises project.

If have only an on-premises project

If have only an on-premises project, you can't update the existing credential with KeyId '<key>'.

```
New-AzureRmADSpCredential : Update to existing credential with KeyId '<key>' is not allowed.
At C:\InfrastructureScripts\Add-CertToServicePrincipal.ps1:62 char:1
New-AzureRmADSpCredential -ObjectId $servicePrincipal.Id -CertValue $ ...
CategoryInfo : InvalidOperation: (:) [New-AzureRmADSpCredential], Exception
FullyQualifiedErrorId : Microsoft.Azure.Commands.ActiveDirectory.NewAzureADSpCredentialCommand
```

Run the following PowerShell command to resolve the issue.

```
Remove-AzureRmADSpCredential -ServicePrincipalName "00000015-0000-0000-c000-000000000000" -KeyId <key>
```

If you have both an online project and an on-premises project

If you have both an online project and an on-premises project, follow these steps.

1. Verify that the Microsoft .NET Framework version 4.7.2 is installed.
2. Run the following Windows PowerShell script to install the Azure PowerShell module.

   ```
   Install-Module -Name Az
   ```

3. Run the following Windows PowerShell script to upload the new certificate.

```
Import-Module -Name Az.Accounts
Import-Module -Name Az.Resources

Connect-AzAccount

$servicePrincipalName = "00000015-0000-0000-c000-000000000000";
$CertificateThumbprint = <Thumbprint of Agent Certificate>
$cert = Get-ChildItem -path Cert:\CurrentUser\my | Where-Object { $_.Thumbprint -eq
$CertificateThumbprint }
if (!$cert)
{
    $cert = Get-ChildItem -path Cert:\LocalMachine\my | Where-Object { $_.Thumbprint -eq
$CertificateThumbprint }
    if (!$cert)
    {
        throw "Unable to find the certificate in the Local machine or Current User store"
    }
}

$keyValue = [System.Convert]::ToBase64String($cert.GetRawCertData())
$servicePrincipal = Get-AzADServicePrincipal -ServicePrincipalName $servicePrincipalName
if (!$servicePrincipal)
{
    throw "Unable to find the service principal"
}
New-AzADSpCredential -ObjectId $servicePrincipal.Id -CertValue $keyValue -EndDate $cert.NotAfter -
StartDate $cert.NotBefore
Get-AzADSpCredential -ObjectId $servicePrincipal.Id
```

4. Run the following command to remove the duplicate certificate, if more than one certificate exists.

```
Remove-AzADSpCredential -ServicePrincipalName "00000015-0000-0000-c000-000000000000" -KeyId <key>
```

ODBC driver 17 is required for platform updates

The latest platform binary update uses Open Database Connectivity (ODBC) driver 17. This upgrade resolves stability issues that are linked to older ODBC drivers. The Setup perquisites documentation has been updated to reflect the change in which ODBC driver 17 must be installed on each AOS server. If you don't install ODBC driver 17, you will receive DB Sync errors during servicing of the environment.

Here are some examples of errors:

- In Service Fabric:

> Unhealthy event: SourceId='System.RA', Property='ReplicaOpenStatus', HealthState='Warning', ConsiderWarningAsError=false.
> Replica had multiple failures during open on AOS3. API call: IStatelessServiceInstance.Open(); Error = System.Exception (-2146233088)
> **DB sync failed.**

- On AOS machines:
  - Event Viewer > Custom Views > Administrative Events:

> Application: Microsoft.Dynamics.AX.Deployment.Setup.exe Framework Version: v4.0.30319
> Description: The process was terminated due to an unhandled exception. Exception Info: System.IO.FileNotFoundException at Microsoft.Dynamics.AX.Deployment.Setup.Program.Main(System.String[])

- C:\ProgramData\SF\AOSx\Fabric\work\Applications\AXSFType_Appxx\log:

> Microsoft.Dynamics.AX.Deployment.Setup.exe -bindir "C:\ProgramData\SF\AOS1\Fabric\work\Applications\AXSFType_App18\AXSF.Code.1.0.20180831174152\Packages" -metadatadir "C:\ProgramData\SF\AOS1\Fabric\work\Applications\AXSFType_App18\AXSF.Code.1.0.20180831174152\Packages" -sqluser "axdbadmin" -sqlserver "SQL-LS.contoso.com" -sqldatabase "AXDB" -setupmode servicesync -syncmode fullall -onprem
>
> Unhandled Exception: System.IO.FileNotFoundException: **Could not load file or assembly 'aoskernel.dll' or one of its dependencies. The specified module could not be found.** at Microsoft.Dynamics.AX.Deployment.Setup.Program.Main(String[] args)
>
> **DB sync failed.**

A "No subscription found in the context" error occurs when you run Add-CertToServicePrincipal

Recent versions of Windows PowerShell might cause a "No subscription found in the context" error. To resolve this issue, install and load an older version of Windows PowerShell, such as version 5.7.0.

```
# Install version 5.7.0 of Azure PowerShell
Install-Module -Name AzureRM -RequiredVersion 5.7.0

# Load version 5.7.0 of Azure PowerShell
Import-Module -Name AzureRM -RequiredVersion 5.7.0
```

Service Fabric Explorer warnings occur after you restart a machine

Error:

> Error event: SourceId='MonitoringAgentService', Property='ServiceState'. System.Management.Automation.RuntimeException: Error: **The GUID passed was not recognized as valid by a WMI data provider.** (Exception from HRESULT: 0x80071068). Stack trace:

Steps: To resolve this issue, restart the application package that generated the warning message. For more information, see Restart applications (such as AOS).

The internal time zone version number that is stored in the database is higher than the version that is supported by the kernel (13/12)

This database synchronization error can cause an old platform build (Platform update 12) to be deployed on top of a database that had a newer build (Platform update 15).

To resolve this issue, note the **SYSTIMEZONESVERSION** value.

```
select * from SQLSYSTEMVARIABLES where parm = 'SYSTIMEZONESVERSION'
```

Update the value to the version that was returned in the error message.

```
update SQLSYSTEMVARIABLES set VALUE = 12 where parm = 'SYSTIMEZONESVERSION'
```

Printing randomly stops

Make sure that all network printers that have been installed on the AOS server are running as the Windows service account that the AXService.EXE process is running as.

For more information about how to configure network printers in on-premises environments, see Install network printer devices in on-premises environments.

Ax-DatabaseSynchronize isn't populated with events

In Platform update 20 and later, there is database synchronization log issue where the synchronization logs aren't written under **Ax-DatabaseSynchronize** in Event Viewer.

To resolve this issue, go to <SF-dir>\AOS_<x>\Fabric\work\Applications\AXSFType_App<X>\log. For example, go to C:\ProgramData\SF\AOS_11\Fabric\work\Applications\AXSFType_App183\log. Here, you can see the output from DatabaseSynchronize in the Code_AXSF_M_<X>.out files. Troubleshoot any issues that pertain to this component.

You can't access Finance + Operations: "AADSTS50058: A silent sign-in request was sent but no user is signed in"

After a user enters credentials to sign in to Finance + Operations, the browser briefly shows the application layout. However, it then tries to redirect outside Finance + Operations, but fails with the following error:

> AADSTS50058: A silent sign-in request was sent but no user is signed in.

The cookies that represent the user's session weren't sent in the request to Azure AD. This issue can occur if the user is using Internet Explorer or Microsoft Edge, and if the web app that sends the silent sign-in request is in a different IE security zone than the Azure AD endpoint (login.microsoftonline.com).

This issue occurs because there was a change in the Skype Presence API, and on-premises environments connect to this API by default.

To resolve the issue, run the following SQL Server query.

```
update [AXDB].[dbo].[SYSCLIENTPERF] set SkypeEnabled = 0
```

Alternatively, turn off the **Skype presence enabled** option on the **Client performance options** page (**System administration** > **Setup** > **Client performance options**). To use this approach, you must be able to sign in to Finance + Operations. Therefore, you must first block redirection in the browser. After you disable the Skype presence, you can unblock redirection again.

The Google Chrome browser blocks redirection by default.

Error: "There was an error during CodePackage activation. Service host failed to activate. Error:0x8007052e"

You might receive the following error during a new installation:

> Error event: SourceId='System.Hosting', Property='CodePackageActivation:Code:EntryPoint'. There was an error during CodePackage activation.Service host failed to activate. Error:0x8007052e

This error will cause the AXSF service to fail with the same error.

To resolve this issue, follow these steps.

1. In the agent share path, find the **netstandard.dll** file. For example, this file might be at \wp\<name>\StandaloneSetup-<ver>\Apps\AOS\AXServiceApp\AXSF\Code\bin\netstandard.dll.
2. On each AOS server, open a Command Prompt window as an administrator, and run the following command.

   ```
   "C:\Program Files (x86)\Microsoft SDKs\Windows\v8.1A\bin\NETFX 4.5.1 Tools\gacutil.exe" -i <path from step 1.>\netstandard.dll /f
   ```

3. Delete **AXBootstrapperApp** from Service Fabric.
   a. Delete the **fabric:/Bootstrapper/AXBootstrapper** service.
   b. Delete the **fabric:/Bootstrapper** application.
   c. Unprovision the **AXBootstrapperAppType** type.
4. Redeploy the environment from LCS.

SQL Server 2016 service pack 2 is recommended for Reporting Services instances

When you go through LCS servicing operations, you might receive the following error:

> The process cannot access the file 'C:\Program Files\Microsoft SQL Server\MSRS13.MSSQLSERVER\Reporting Services\ReportServer\bin\Microsoft.Dynamics.AX.Framework.Services.Platform.Client.dll' because it is being used by another process.

This issue occurs because Reporting Services has a lock on a Microsoft Dynamics .dll file. We currently recommend that you have SQL Server 2016 service pack 2 installed on Reporting Services instances.

> **NOTE**
> You must have service pack 2 installed, and no additional cumulative updates or hotfixes must be installed.

SysClassRunner doesn't run successfully

Issue: When you try to run SysClassRunner on Platform update 29 through Platform update 31, you get the following exception:

```
Microsoft.Dynamics.Ax.Xpp.ClrErrorException: TypeInitializationExeption --->
System.TypeInitializationException: The type initializer for
'Microsoft.Dynamics.Ax.Metadata.XppCompiler.CompilerTracer' threw an exception. --->
System.TypeInitializationException: The type initializer for
'Microsoft.Dynamics.Ax.DesignTime.Telemetry.OneDS' threw an exception. --->
System.IO.FileLoedAxception: Could not load file or assembly 'Microsoft.Diagnostics.Tracing.TraceEvent,
Version=2.0.43.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a' or one of its dependencies.
The located assembly's manifest definition does not match the assembly reference. (Exception from HRESULT:
0x80131040) at Microsoft.Dynamics.Ax.DesignTime.Telemetry.OneDS.cctor()
--- End of inner exception stack trace ---
```

Reason: There is a .dll mismatch between the runtime and the application.

Resolution: Use TSG_SysClassRunner.ps1. For more information, see TSG_SysClassRunner.ps1.

DBSync fails with PEAP APP version 10.0.9 Platform update 33

Issue: During deployment of the APP 10.0.9 PU33 PEAP-package, the deployment fails with the AXSF applications staying in "InBuild" status in Service Fabric explorer. When reviewing the logs on the AXSF nodes's work directories, the following DBSync error can be found.

Error message from DBSync:

```
Microsoft.Dynamics.AX.Deployment.Setup.exe -bindir
"C:\ProgramData\SF\LBDEN08FS1AOS03\Fabric\work\Applications\AXSFType_App398\AXSF.Code.1.0.20200123151456\Pac
kages" -metadatadir
"C:\ProgramData\SF\LBDEN08FS1AOS03\Fabric\work\Applications\AXSFType_App398\AXSF.Code.1.0.20200123151456\Pac
kages" -sqluser "" -sqlserver "" -sqldatabase "" -setupmode servicesync -syncmode fullall -onprem
Stack trace: Invalid attempt to call  running in CIL on the client.
  at Microsoft.Dynamics.Ax.MSIL.Interop.throwException(Int32 ExceptionValue, interpret* ip)
  at Microsoft.Dynamics.Ax.MSIL.Interop.ThrowCQLError(IL_CQL_ERR cqlErr, String p1)
  at Microsoft.Dynamics.AX.Kernel.ApplicationId.LogOrRethrow(Exception exception)
  at Microsoft.Dynamics.AX.Kernel.ApplicationId.LogOrRethrowFormattedMessage(Exception exception, String
typeName, String elementName)
  at Microsoft.Dynamics.AX.Kernel.ApplicationId.LogOrRethrowFormattedMessage(Exception exception, String
typeName, Int32 typeId)
  at Microsoft.Dynamics.AX.Kernel.ApplicationId.ApplicationIdBridge.LoadTableById(ApplicationIdBridge* ,
Int32 id, ObjectIdDelegate* cb)
  at cqlClass.callEx(cqlClass* , Char* , interpret* )
  at Microsoft.Dynamics.Ax.MSIL.cqlClassIL.Call(IntPtr c, String methodName, Object[] parameters, Type[]
types, Object[] varargs, Type[] varargsTypes)
  at Microsoft.Dynamics.Ax.Xpp.XppObjectBase.Call(String methodName, Object[] parameters, Type[] types,
Object[] varargs)
  at Microsoft.Dynamics.Ax.Xpp.DictTable.Supportinheritance()
  at Dynamics.AX.Application.SysDictTable.`getRootTable(Int32 _tabid) in
xppSource://Source/ApplicationPlatform\AxClass_SysDictTable.xpp:line 1498
  at Dynamics.AX.Application.SysDictTable.getRootTable(Int32 _tabid)
  at Dynamics.AX.Application.SysDataBaseLog.`ConfigureSqlLogging() in
xppSource://Source/ApplicationPlatform\AxTable_SysDataBaseLog.xpp:line 60
  at Dynamics.AX.Application.SysDataBaseLog.ConfigureSqlLogging()
  at SysDataBaseLog::ConfigureSqlLogging(Object[] , Boolean& )
  at Microsoft.Dynamics.Ax.Xpp.ReflectionCallHelper.MakeStaticCall(Type type, String MethodName, Object[]
parameters)

DB sync failed.
```

Reason: This issue occurs because there is data in the SQL DatabaseLog table that conflicts with the metadata in the package.

Resolution: Run the following query on AXDB to clean the DatabaseLog table and retry the deployment.

```
select * into databaselog_bak from databaselog
truncate table databaselog
```

DBSync fails to start

Issue: During deployment, the deployment fails with the AXSF applications staying in "InBuild" status in Service Fabric explorer. When reviewing the logs on the AXSF nodes's work directories, the following DBSync error can be found.

```
Microsoft.Dynamics.AX.InitializationException: Database login failed. Please check SQL credentials and try
again.
   at Microsoft.Dynamics.AX.AOS.StartupInternal(String[] Arguments)
   at Microsoft.Dynamics.AX.AOS.Startup()
   at Microsoft.Dynamics.AX.AosConfig.?A0xb5100bbf.GetAosConfig()
   at Microsoft.Dynamics.AX.AosConfig.Config.InitInternal()
   at Microsoft.Dynamics.AX.AosConfig.Config.InitOnce(Boolean isOfflineMode)
   at Microsoft.Dynamics.AX.Framework.Database.Tools.LegacyCodepath.StartAosCode(SyncOptions syncOptions,
String sqlConnectionString)
   at Microsoft.Dynamics.AX.Framework.Database.Tools.LegacyCodepath.ExecuteWithinAOS(SyncOptions
syncOptions, String sqlConnectionString, IMetadataProvider metadataProvider, Func`1 func, Action`1
errorHandler)
   at
Microsoft.Dynamics.AX.Framework.Database.Tools.LegacyCodepath.NOTE_LeavingSynchronizer_CallStackAboveThisLin
eIsCustomCode(SyncOptions syncOptions, String sqlConnectionString, IMetadataProvider metadataProvider,
Action`1 a)
   at Microsoft.Dynamics.AX.Framework.Database.Tools.LegacyCodepath.RunCustomAction(SyncOptions syncOptions,
String sqlConnectionString, IMetadataProvider metadataProvider, Action`1 a)
   at Microsoft.Dynamics.AX.Framework.Database.Tools.SyncEngine.PreTableSync()
   at Microsoft.Dynamics.AX.Framework.Database.Tools.SyncEngine.FullSync()
   at Microsoft.Dynamics.AX.Framework.Database.Tools.SyncEngine.RunSync()
   at Microsoft.Dynamics.AX.Framework.Database.Tools.SyncEngine.Run(String metadataDirectory, String
sqlConnectionString, SyncOptions options)
```

Reason: This issue may occur because the SQL password contains special characters.

Resolution: Update the password of the SQL user and remove the special characters. Then, update the Credentials.json file with the new password and retry the deployment from LCS.

DBSync fails with PEAP and first release APP version 10.0.14 Platform update 38

Issue: During deployment, the deployment fails with the AXSF applications staying in "InBuild" status in Service Fabric explorer. When reviewing the logs on the AXSF nodes's work directories, the following DBSync error is present multiple times.

```
10/01/2020 14:49:25: Failed when creating deadlock capture session event System.Data.SqlClient.SqlException
(0x80131904): User does not have permission to perform this action.
   at System.Data.SqlClient.SqlConnection.OnError(SqlException exception, Boolean breakConnection, Action`1
wrapCloseInAction)
   at System.Data.SqlClient.TdsParser.ThrowExceptionAndWarning(TdsParserStateObject stateObj, Boolean
callerHasConnectionLock, Boolean asyncClose)
   at System.Data.SqlClient.TdsParser.TryRun(RunBehavior runBehavior, SqlCommand cmdHandler, SqlDataReader
dataStream, BulkCopySimpleResultSet bulkCopyHandler, TdsParserStateObject stateObj, Boolean& dataReady)
   at System.Data.SqlClient.SqlCommand.RunExecuteNonQueryTds(String methodName, Boolean async, Int32
timeout, Boolean asyncWrite)
   at System.Data.SqlClient.SqlCommand.InternalExecuteNonQuery(TaskCompletionSource`1 completion, String
methodName, Boolean sendToPipe, Int32 timeout, Boolean& usedCache, Boolean asyncWrite, Boolean inRetry)
   at System.Data.SqlClient.SqlCommand.ExecuteNonQuery()
   at Microsoft.Dynamics.AX.Framework.Database.Monitor.DeadlockMonitor.CreateDeadlockTrackingSystemEvent()
```

Reason: This issue occurs because the SQL Server account used by Finance + Operations does not have sufficient permissions to execute the operation.

Resolution: Execute the following command in your SQL Server:

```
use master
GRANT ALTER ANY EVENT SESSION to axdbadmin;
```

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Scripts for resolving issues in on-premises environments

2/18/2021 • 4 minutes to read • Edit Online

This topic will serve as a central repository for scripts that you can use to fix issues in on-premises environments. These scripts must usually be run as pre-deployment or post-deployment scripts.

For more information about how to resolve issues in on-premises environments, see Troubleshoot on-premises deployments.

Prepare your environment for script execution

1. Configure the execution of pre-deployment and post-deployment scripts. For more information, see Local agent pre-deployment and post-deployment scripts.
2. Add the following code to your Predeployment.ps1 script.

```
# This has to be filled out
# $agentShare = '<Agent-share path>' # E.g '\\LBDContosoShare\agent''

$agentShare = '\\servername\D365FFOAgent'
Write-Output "AgentShare is set to $agentShare"

# The scripts make the assumption that the wp folder only contains one folder for the environment
name.
# If you have multiple folders in there from older deployments, then please remove those.
# It is not recommended to use the same agent share for multiple environments.

#& $agentShare\scripts\TSG_UpdateFRDeployerConfig.ps1 -agentShare $agentShare

#& $agentShare\scripts\TSG_WindowsAzureStorage.ps1 -agentShare $agentShare

#& $agentShare\scripts\TSG_SysClassRunner.ps1 -agentShare $agentShare

#& $agentShare\scripts\TSG_RemoveFilesFromZip.ps1 -agentShare $agentShare -filesToRemove
'Packages\TaxEngine\bin\Microsoft.Dynamics365.ElectronicReportingMapping.dll','Packages\TaxEngine\bin
\Microsoft.Dynamics365.ElectronicReportingMapping.pdb','Packages\TaxEngine\bin\Microsoft.Dynamics365.
ElectronicReportingServiceContracts.dll','Packages\TaxEngine\bin\Microsoft.Dynamics365.ElectronicRepo
rtingServiceContracts.pdb','Packages\TaxEngine\bin\Microsoft.Dynamics.ElectronicReporting.Instrumenta
tion.dll','Packages\TaxEngine\bin\Microsoft.Dynamics.ElectronicReporting.Instrumentation.pdb','Packag
es\TaxEngine\bin\Microsoft.Dynamics365.LocalizationFrameworkCore.dll','Packages\TaxEngine\bin\Microso
ft.Dynamics365.LocalizationFrameworkCore.pdb','Packages\TaxEngine\bin\Microsoft.Dynamics365.Localizat
ionFrameworkForAx.dll','Packages\TaxEngine\bin\Microsoft.Dynamics365.LocalizationFrameworkForAx.pdb'
```

3. From the relevant section of this topic, copy the code that you require to fix your issue, and paste it into a new file. Save this file in the same folder where your Predeployment.ps1 script is stored. The file name must match the title of the section that you copied the code from. Repeat this step for other issues that you must fix.
4. In the Predeployment.ps1 script, in the code that you added earlier, uncomment the lines that invoke the scripts that you want to use.

TSG_SysClassRunner.ps1

The following script is used to fix an issue that occurs when SysClassRunner is run in some versions of the

platform. For more information about this issue, see SysClassRunner doesn't run successfully.

```
param (
    [Parameter(Mandatory=$true)]
    [string]
    $agentShare = ''
)

$delete = @("Microsoft.Diagnostics.Tracing.TraceEvent.dll", "Microsoft.AI.Agent.Intercept.dll",
"Microsoft.AI.DependencyCollector.dll", "Microsoft.AI.DependencyCollector.xml",
"Microsoft.AI.PerfCounterCollector.dll", "Microsoft.AI.ServerTelemetryChannel.dll",
"Microsoft.AI.ServerTelemetryChannel.xml", "Microsoft.AI.Web.dll", "Microsoft.AI.Web.xml",
"Microsoft.AI.WindowsServer.dll","Microsoft.AI.WindowsServer.xml", "Microsoft.ApplicationInsights.dll",
"Microsoft.ApplicationInsights.xml")

$ErrorActionPreference = "Stop"

$basePath = Get-ChildItem $agentShare\wp\*\StandaloneSetup-*\Apps |
    Select-Object -First 1 -Expand FullName

#Some customers experience an unexpected behavior with the previous command.
if($basePath -notmatch "\AOS")
{
    $basePath = Join-Path $basePath -ChildPath "\AOS"
}

$basePath = Join-Path $basePath -ChildPath "AXServiceApp\AXSF\Code"

if(!(Test-Path $basePath))
{
    Write-Error "Basepath: $basePath , not found" -Exception InvalidOperation
}

foreach( $file in $delete)
{
    if(Test-Path -Path "$basePath\$file")
    {
        Remove-Item -Path "$basePath\$file"
    }
}

$axConfig = Join-Path $basePath -ChildPath "AXService.exe.config"

if(!(Test-Path $axConfig))
{
    Write-Error "Unable to find AxService.exe.config in path: $axConfig" -Exception InvalidOperation
}

Write-Output "Found config: $axConfig"

[xml]$xml = get-content $axConfig
$xml.SelectNodes("//*[@name = 'Microsoft.AI.Agent.Intercept']/..") | ForEach-Object {
    $_.bindingRedirect.oldVersion = "0.0.0.0-2.4.0.0"
    $_.bindingRedirect.newVersion = "2.4.0.0"
}

if($xml.SelectNodes("//*[@name = 'Microsoft.ApplicationInsights']").Count -eq 0)
{
    [xml]$newNode = @"
    <dependentAssembly xmlns="urn:schemas-microsoft-com:asm.v1">
        <assemblyIdentity name="Microsoft.ApplicationInsights" publicKeyToken="31bf3856ad364e35"
culture="neutral" />
        <bindingRedirect oldVersion="0.0.0.0-2.9.0.0" newVersion="2.9.0.0" />
    </dependentAssembly>
"@
    $xml.configuration.runtime.assemblyBinding.AppendChild($xml.ImportNode($newNode.dependentAssembly,
$true))
```

```
}

$xml.save($axConfig)

Write-Output "TSG SysClassRunner script succeeded"
```

TSG_UpdateFRDeployerConfig.ps1

The following script is used to fix an issue that occurs when Financial Reporting is deployed in some versions of the platform. For more information about this issue, see Could not load file or assembly EntityFramework.

```
param (
    [Parameter(Mandatory=$true)]
    [string]
    $agentShare = ''
)

$frConfig = Get-ChildItem $agentShare\wp\*\StandaloneSetup-
*\Apps\FR\Deployment\FinancialReportingDeployer.exe.config |
    Select-Object -First 1 -Expand FullName

if( -not $frConfig)
{
    Write-Output "Unable to find FinancialReportingDeployer.exe.Config"
    return
}

Write-Output "Found config: $frConfig"

[xml]$xml = get-content $frConfig

$nodeList = $xml.GetElementsByTagName("loadFromRemoteSources")

if($nodeList.Count -eq 0)
{
    # Create the node
    $newNode = $xml.CreateNode("element","loadFromRemoteSources","")
    $newNode.SetAttribute("enabled","true")
    # Find the parent
    $nodeList = $xml.GetElementsByTagName("runtime")
    $runtimeNode = $nodeList[0]
    $runtimeNode.AppendChild($newNode)
    # Save doc
    $xml.save($frConfig)
    Write-Output "Inserted new node: "$newNode.Name
}
else
{
    $node = $nodeList[0]

    $attribute = $node.Attributes.GetNamedItem("enabled")

    if($attribute.Value -eq "true")
    {
        Write-Output "Node already exists: "$node.Name
    }

    else
    {
        Write-Output "Node already exists but attribute is incorrect: " $attribute.Name "is"
$attribute.Value
    }
}
```

TSG_WindowsAzureStorage.ps1

The following script is used to fix an issue where files can't be downloaded or exported in some versions of the platform.

```
param (
    [Parameter(Mandatory=$true)]
    [string]
    $agentShare = ''
)
$ErrorActionPreference = "Stop"

$delete = @("Microsoft.WindowsAzure.Storage.dll", "Microsoft.WindowsAzure.Storage.xml")

$basePath = Get-ChildItem $agentShare\wp\*\StandaloneSetup-*\Apps |
    Select-Object -First 1 -Expand FullName

#Some customers experience an unexpected behavior with the previous command.
if($basePath -notmatch "\AOS")
{
    $basePath = Join-Path $basePath -ChildPath "\AOS"
}

$basePath = Join-Path $basePath -ChildPath "AXServiceApp\AXSF\Code"

if(!(Test-Path $basePath))
{
    Write-Error "Basepath: $basePath , not found" -Exception InvalidOperation
}

foreach( $file in $delete)
{
    if(Test-Path -Path "$basePath\$file")
    {
        Remove-Item -Path "$basePath\$file"
    }
}

Write-Output "TSG WindowsAzureStorage script succeeded"
```

TSG_RemoveFilesFromZip.ps1

The following script is used to fix an issue that occurs for some customers who were previously on versions 10.0.5 through 10.0.9. Due to how the prepare process works, there are some older versions of DLLs that remain in the TaxEngine folder, which in newer releases have been moved to different module folders. This script ensures that these DLLs are removed from the downloaded asset, before being deployed to the AOS nodes.

```
[CmdletBinding()]
param
(
    [Parameter(Mandatory)]
    [ValidateNotNullOrEmpty()]
    [ValidateScript({ Test-Path -Path $_ })]
    [string] $agentShare,

    [ValidateNotNullOrEmpty()]
    [string[]]$filesToRemove
)

#requires -Version 5
$ErrorActionPreference = 'Stop'
$InformationPreference = 'Continue'
$files = @()

[Reflection.Assembly]::LoadWithPartialName('System.IO.Compression')

if(! (Test-Path $AgentShare))
{
    throw "The path provided for the agentshare is not valid/accessible."
}

ForEach($file in $FilesToRemove)
{
    if(!$file.StartsWith('Packages'))
    {
        throw "The path of $file does not start with Packages."
    }
    $files += $file.Replace('\', '/')
}

$basePath = Get-ChildItem $AgentShare\wp\*\StandaloneSetup-*\Apps | Select-Object -First 1 -Expand FullName

#Some customers experience an unexpected behavior with the previous command.
if($basePath -notmatch "\AOS")
{
    $basePath = Join-Path $basePath -ChildPath "\AOS"
}

$basePath = Join-Path $basePath -ChildPath "AXServiceApp\AXSF\Code"
$zipfile = Join-Path $basePath -ChildPath "Packages.zip"

try
{
    $stream = New-Object IO.FileStream($zipfile, [IO.FileMode]::Open)
    $mode   = [IO.Compression.ZipArchiveMode]::Update
    $zip    = New-Object IO.Compression.ZipArchive($stream, $mode)

    ($zip.Entries | ? { $files -contains $_.FullName }) | % { $_.Delete() } | Write-Host
}
catch
{
    throw 'An Error Occurred'
}
finally
{
    $zip.Dispose()
    $stream.Close()
    $stream.Dispose()
}
```

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Create deployable packages of models

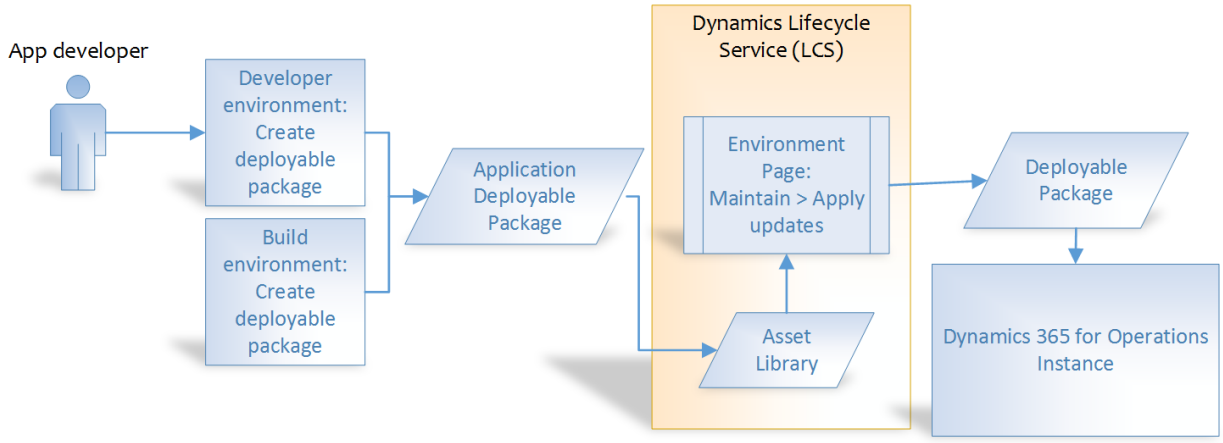
2/18/2021 • 2 minutes to read • Edit Online

An AOT package is a deployment and compilation unit of one or more models that can be applied to an environment. It includes model metadata, binaries, reports and other associated resources. One or more AOT packages can be packaged into a deployable package, which is the vehicle used for deployment of code (and customizations) on demo, sandbox, and production environments. This topic guides you through the process of creating and applying a deployable package.

Overview of the process

In order to deploy your code and customizations to a runtime environment (demo, sandbox, or production), you must create deployable packages of your solution or implementation. Deployable packages can be created by using **Visual Studio dev tools** or by using the **build automation process** that is available on build environments. These deployable packages are referred to as Application Deployable Packages or AOT Deployable Packages. The following image shows an overview of the process. After a deployable package is created, it must be uploaded to the Lifecycle Services (LCS) project's asset library. An administrator can then go to the LCS environment page and apply the package to a runtime environment using the **Maintain > Apply updates** tool.

> **NOTE**
>
> Custom payment connector for Commerce needs to be packaged using a combined AOT deployable package. For more information, see Create payment packaging for Application Explorer in Service Fabric deployments.



> **NOTE**
>
> Application Deployable Packages do not contain source code.

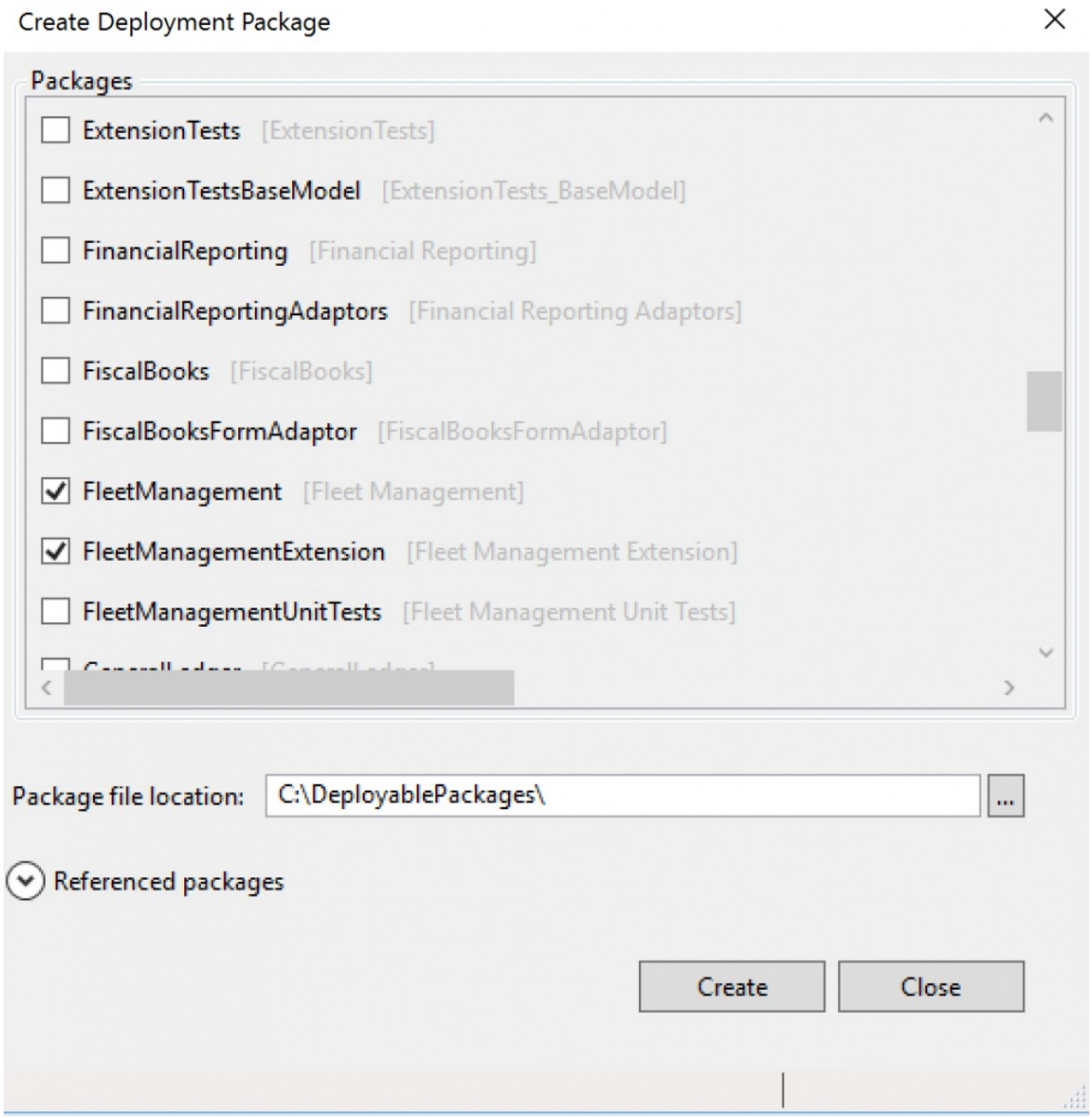
> It is always recommended to use a build environment to create deployable packages that are intended to go to production.

Create a deployable package

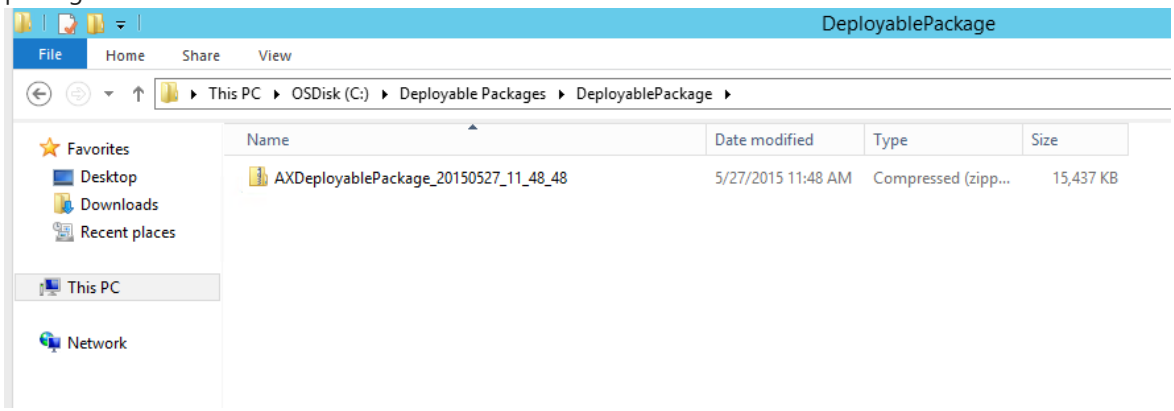
We recommend using a build environment to create deployable packages. You can also create a deployable package on a development environment.

On a development environment, after you have completed development and testing, follow these steps to create a deployable package in Visual Studio.

1. In Microsoft Visual Studio, select **Dynamics 365** > **Deploy** > **Create Deployment Package**.



2. Select the packages that contain your models, and then select a location in which to create the deployable package.



3. After a deployable package is created, sign in to Lifecycle Services, and then, in your LCS project, click the **Asset Library** tile.

4. Upload the deployable package that you created earlier.

Apply a deployable package

To apply a deployable package to an environment, see Apply updates to cloud environments.

Remove a deployable package

To uninstall or remove a deployable package from an environment, see Uninstall a package.

Install deployable packages from the command line

2/18/2021 • 6 minutes to read • Edit Online

This topic walks you through the steps for using the command line to apply either a binary update or an application (AOT) deployable package that was created in your development or build environment.

> **IMPORTANT**
>
> For most types of environments, you can apply a deployable package to an environment directly from Microsoft Dynamics Lifecycle Services (LCS). For more information, see Apply updates to cloud environments. Therefore, this topic applies primarily to environment types that don't support the application of updates via LCS. Examples include local development environments (downloadable virtual hard disks [VHDs]), multi-box development/test environments in Microsoft Azure (LCS Partner and trial projects), and build environments. However, you can also use this topic any time that you want to install deployable packages by using the command line instead of LCS.

Key concepts

- **Deployable package** – A deployable package is a unit of deployment that can be applied to any environment. It can consist of a binary hotfix to the runtime components of Application Object Server (AOS), an updated application package, or a new application package.
- **AXUpdateInstaller** – AXUpdateInstaller is an executable program that is bundled in the deployable package. When the package is downloaded to a computer, the installer is available.
- **Runbook** – The deployment runbook is a series of steps that is generated and used to apply the deployable package to the target environment. Some of the steps are automated, and some are manual. AXUpdateInstaller enables these steps to be run one at a time and in the correct order.

Install an application (AOT) deployable package on a development environment

> **NOTE**
>
> The steps listed below are for customization packages only. Do not use the **devinstall** parameter when running the Data Upgrade deployable package as part of an upgrade from Microsoft Dynamics AX 2012 to a Finance and Operations app.

An AOT deployable package is a package that contains customizations and extensions to your application. If you want to use the command line just to install an AOT deployable package on a development or demo environment, follow the instructions in this section. You can then skip the rest of this topic.

1. On the virtual machine (VM), download the zip file for the deployable package. Make sure that the zip file is stored in a non-user folder.

   > **NOTE**
   >
   > After you download the zip file, right-click it, and then select **Properties**. Then, in the **Properties** dialog box, on the **General** tab, select **Unblock** to unlock the files.

2. Extract the files.

3. Open a Command Prompt window, and go to the folder where you extracted the deployable package.
4. Run the following command.

```
AXUpdateInstaller.exe devinstall
```

The **devinstall** option installs the AOT deployable package on the VM.

> **NOTE**
> This command doesn't run database synchronization. You must run database synchronization from Microsoft Visual Studio after you install the deployable package.

Collect topology configuration data

1. In LCS, on the **Environment** page, select the name of a VM. Establish a Remote Desktop connection to the VM by using the user name and password that are provided on the **Environment** page.
2. On the VM, download the zip file for the deployable package from LCS. Make sure that the zip file is stored in a non-user folder.

> **NOTE**
> After you download the zip file, right-click it, and then select **Properties**. Then, in the **Properties** dialog box, on the **General** tab, select **Unblock** to unlock the files.

3. Extract the files.
4. In the folder where you extracted the deployable package, find and open the file that is named **DefaultTopologyData.xml**. You must specify the VM name and the installed components in this file.
   - To specify the VM name, follow these steps:
     a. In File Explorer, right-click **This PC**, and then select **Properties**.
     b. In the system properties, find and make a note of the computer name (for example, **AOS-950ed2c3e7b**).
     c. In the DefaultTopologyData.xml file, replace the machine name with the computer name that you found in the previous step.
   - To specify the installed components, follow these steps:
     a. Open a Command Prompt window as an administrator.
     b. Go to the extracted folder, and run the following command to see a list of all the components that are installed on the computer.

```
AXUpdateInstaller.exe list
```

     c. Update the DefaultTopologyData.xml file with the list of components.

   When you've finished specifying the VM name and the installed components, the DefaultTopologyData.xml file should resemble the following illustration.

```
<?xml version="1.0" encoding="utf-8"?>
<TopologyData xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Name>VAL200AA2BMEDIU</Name>
  <MachineList>
    <Machine>
      <Name>AOS-950ed2c3e7b</Name>
      <ServiceModelList>
  <string>AOSService</string>
        <string>DIXFService</string>
        <string>RetailCloudPos</string>
        <string>RetailSelfService</string>
        <string>RetailServer</string>
  <string>SCMSelfService</string>
      </ServiceModelList>
    </Machine>
    <Machine>
      <Name>AOS-499ee94c842</Name>
      <ServiceModelList>
  <string>AOSService</string>
        <string>DIXFService</string>
        <string>RetailCloudPos</string>
        <string>RetailSelfService</string>
        <string>RetailServer</string>
  <string>SCMSelfService</string>
      </ServiceModelList>
    </Machine>
    <Machine>
      <Name>BI-1a62b3b31d78</Name>
      <ServiceModelList>
  <string>ReportingService</string>
      </ServiceModelList>
    </Machine>
    <Machine>
      <Name>BI-7ee6d8fbed7a</Name>
      <ServiceModelList>
  <string>ReportingService</string>
      </ServiceModelList>
    </Machine>
  </MachineList>
  <BackupScript>
    <FileName />
    <Automated>false</Automated>
    <Description>Please backup your enviorment now, set the step to complete once you finsihed backup</Description>
  </BackupScript>
</TopologyData>
```

5. Repeat steps 1 through 4 for every other VM that is listed on the **Environment** page.

Generate a runbook from the topology

Based on the topology information in the DefaultTopologyData.xml file, you must generate the runbook file that will provide step-by-step instructions for updating each VM.

- On any VM, run the following command to generate the runbook.

  ```
  AXUpdateInstaller.exe generate -runbookid=[runbookID] -topologyfile=[topologyFile] -servicemodelfile=
  [serviceModelFile] -runbookfile=[runbookFile]
  ```

  Here is an explanation of the parameters that are used in this command:

  - **[runbookID]**– A parameter that is specified by the developer who applies the deployable package.
  - **[topologyFile]**– The path of the DefaultTopologyData.xml file.
  - **[serviceModelFile]**– The path of the DefaultServiceModelData.xml file.
  - **[runbookFile]**– The name of the runbook file to generate (for example, **AOSRunbook.xml**).

    Example

    ```
    AXUpdateInstaller.exe generate -runbookid="VAL200AA2BMEDIU-runbook" -
    topologyfile="DefaultTopologyData.xml" -servicemodelfile="DefaultServiceModelData.xml" -
    runbookfile="VAL200AA2BMEDIU-runbook.xml"
    ```

The runbook provides the sequence of steps that must be run to update the environment. The following illustration shows an example of a runbook file. Each step in a runbook is associated with an ID, a machine name, and step execution details.

```
<?xml version="1.0" encoding="UTF-8"?>
- <RunbookData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/200
    <RunbookID>AXDeployablePackage_20160212_22_57_44</RunbookID>
  - <RunbookTopologyData>
      <Name>AX topology</Name>
    - <MachineList>
      - <Machine>
          <Name>AOS-77edc66f7a1</Name>
        - <ServiceModelList>
            <string>AOSService</string>
            <string>DIXFService</string>
            <string>RetailCloudPos</string>
            <string>RetailSelfService</string>
            <string>RetailServer</string>
            <string>SCMSelfService</string>
          </ServiceModelList>
        </Machine>
      + <Machine>
      + <Machine>
      + <Machine>
      </MachineList>
    + <BackupScript>
    </RunbookTopologyData>
  + <RunbookServiceModelData>
  - <RunbookStepList>
    - <Step>
        <ID>1</ID>
        <Description>Stop script for service model: AOSService on machine: AOS-77edc66f7a1</Description>
        <MachineName>AOS-77edc66f7a1</MachineName>
        <ServiceModelName>AOSService</ServiceModelName>
      - <ScriptToExecute>
          <FileName>AutoStopAOS.ps1</FileName>
```

Install a deployable package

1. On the first machine (VM) that is listed in the runbook file, follow these steps:

   a. Import the runbook by running the following command.

   ```
   AXUpdateInstaller.exe import -runbookfile=[runbookFile]
   ```

   Example

   ```
   AXUpdateInstaller.exe import -runbookfile="VAL200AA2BMEDIU-runbook.xml"
   ```

   b. Verify the runbook.

   ```
   AXUpdateInstaller.exe list
   ```

   c. Run the runbook.

   ```
   AXUpdateInstaller.exe execute -runbookid=[runbookID]
   ```

   Example

   ```
   AXUpdateInstaller.exe execute -runbookid="VAL200AA2BMEDIU-runbook"
   ```

   AXUpdateInstaller updates the runbook file after each step is run on a VM. The runbook also logs information about each step.

   For manual steps, follow the instructions, and then run the following command to mark the step as completed in the runbook.

```
AXUpdateInstaller.exe execute -runbookID=[runbookID] -setstepcomplete=[stepID]
```

Example

```
AXUpdateInstaller.exe execute -runbookid="VAL200AA2BMEDIU-runbook" -setstepcomplete=2
```

If errors occur during any step, debug the script or the instructions in the step, and update accordingly.

d. Export the runbook.

```
AXUpdateInstaller.exe export -runbookid=[runbookID] -runbookfile=[runbookFile]
```

Example

```
AXUpdateInstaller.exe export -runbookid="VAL200AA2BMEDIU-runbook" -
runbookfile="VAL200AA2BMEDIU-runbook.xml"
```

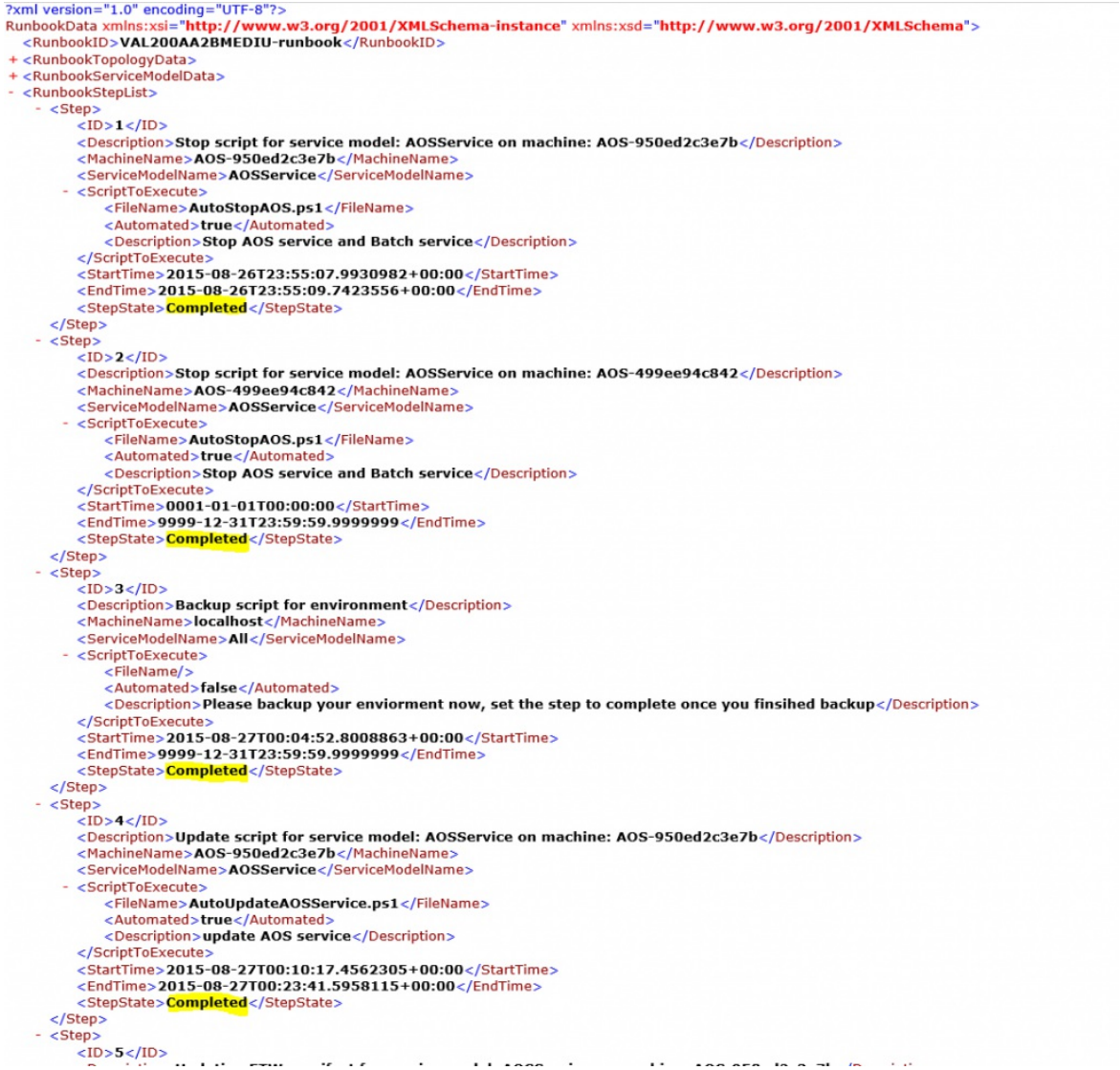
2. Repeat step 1 on every other VM that is listed in the runbook file. For one-box environments, such as development, build, and demo environments, there is only one VM.

Verify installation

1. Run the following command to verify that the new updates are installed.

```
AXUpdateInstaller.exe list
```

2. View the runbook to see the completed steps. Here is an example of a runbook file where the steps have been completed.



Backup the runbook file

- After all the steps in the runbook are completed and you've exported the runbook, save the file outside the computer for future reference. For example, you might have to use the runbook file in these situations:
  - You must analyze the downtime requirements for production, and so on.
  - You must send the file to Microsoft because a deployable package can't be installed.

Troubleshooting

- If any step in the runbook fails, you can rerun it by running the following command.

```
AXUpdateInstaller.exe execute -runbookid=[runbookID] -rerunstep=[stepID]
```

- To prevent version mismatch or downgrade, or installation of the same deployable package, run the following command.

```
AXUpdateInstaller.exe execute -runbookid=[runbook ID] -versioncheck=true
```

- To verify database synchronization, in the **aosservice\scripts** folder, find and open the **dbsync.error.txt** file, and look for any errors.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Uninstall a package

2/18/2021 • 2 minutes to read • Edit Online

Occasionally, you might have to uninstall a deployable package. For example, you might be reorganizing your source code. Alternatively, you no longer require an independent software vendor (ISV) product and haven't renewed the license. Therefore, you must remove the package.

Remove a model

A model is a design-time concept that is part of a package. When a model isn't the only model in a module, you can just remove it from the source code. No other steps are required, because when you deploy the updated module, the old module is overwritten. All overlayer models fall into this category. For more information, see Deleting a model.

Prerequisites

- If any models reference the module that will be removed, the references must be removed from them. For information about how to find the references that must be removed, see Viewing model dependencies.
- Build and deploy any modules that references were removed from.
- All references to and from the modules must be removed before you begin to uninstall the module. To remove all a module's references, add a single class to the model. This class should contain no code. It should contain only a reference to the application platform.
- A Microsoft module cannot be removed. If this is attempted, a validation error will be shown on the package in Lifecycle Services.
- A module cannot be removed if it is part of the AOT deployable package being installed. If you want to remove a module, be sure that it is not part of the package before adding the name to the ModuleToRemove.txt file.

Uninstall a package

1. Create a file that is named **ModuleToRemove.txt**.
2. In the file, put the name of each module that you want to remove on a separate line. Make sure that you've completed the prerequisites for each module that you're removing.
3. Create a valid deployable package, and put the ModuleToRemove.txt file in the **package\AOSService\Scripts** folder.
4. Upload the package to the Lifecycle Services asset library. Wait for the asset to finish validation, and review any warnings that are shown on the Asset Details panel on the right side of the page.
5. Install the deployable package. For more information about how to install deployable packages, see Apply updates to cloud environments.
6. Verify that the package was uninstalled before you complete this procedure in a production environment.

Troubleshoot package application issues

2/18/2021 • 9 minutes to read • Edit Online

This topic provides detailed information that will help you troubleshoot issues that might occur when you apply packages on your Tier 1 or Tier 2 through Tier 5 environments. For information about how to apply a package, see Apply updates to cloud environments.

General troubleshooting and diagnostics

If a package isn't successfully applied, you have two options:

- Retry the operation that failed.
- Use the logs.

Retry the failed operation

If package application fails, and you want to retry the operation, select **Resume**.

Use the logs

If package application fails, and you want to use the logs, follow these steps.

1. Download and then unzip the log files.
2. Select the role that a step failed for, such as **AOS** or **BI**.
3. Select the virtual machine (VM) where the step failed. You can find this information in the **Machine name** column in the **Environment updates** section.
4. In the VM logs, select the folder that corresponds to the step where the issue occurred. The folder name identifies the step that each folder corresponds to.

   For example, if the issue occurred during the execution of a step, select the **ExecuteRunbook** folder. The step number is highlighted and is the number after the globally unique identifier (GUID).

Package application issues

Issue: The package that was applied isn't valid

Description

Because the package that was applied wasn't valid, the servicing status is **Failed**, and no updates are listed in the **Environment updates** section. To verify whether the package is valid, follow these steps.

1. Download and unzip the logs.
2. Navigate to the logs for the Application Object Server (AOS) machine.
3. Verify that the **DownloadFilesAndSlipstreamTools-xxx** and **GenerateRunbook-xxx** folders exist.
4. Open the **GenerateRunbook-xxx folder**, and then open the file for the output type.

   If you find an error message or an exception that states that a file is missing or failed to generate any steps for the runbook, there was an attempt to upload a package that wasn't valid.

Action

Select **Abort** to abort the current package, upload a new package, and then restart the servicing flow.

Issue: Package deployment fails even though no steps failed

Description

A time-out occurs when you download the package to the machine. During pre-servicing, a few steps must be completed before steps are completed in the runbook. As part of the pre-servicing, the package must be downloaded to all the machines. The time to download might vary slightly, depending on the datacenter where the environment resides. Any download doesn't occur within 30 minutes is considered a failure in the servicing status and will be stopped.

Action

1. Select **Resume** to see whether you can resolve the issue. If this step doesn't work, move on to step 2.
2. Download the logs from the **Environment** page.
3. Verify that the package download on all the machines has logs that include the DownloadFilesAndSlipstreamTools folder.
4. Review the log files. If you don't see the following text at the end of the log file, the issue occurred because the package download wasn't completed: "Completed file download and slipstream successfully"

Issue: Package deployment fails even though no steps failed

Description

There isn't enough disk space to download the package. Inspect all the machines. This issue occurs if the servicing drive of any machines is full.

Note that if you select **Resume** in this situation, you won't resolve the issue.

To verify that the deployment failed because space issues, follow these steps.

1. Navigate to the **DownloadFilesAndSlipstreamTools-xxx** folder.
2. Open the output file, and view the error message to see whether the step failed because space issues.

Action

As part of the automated cleanup on topologies, any deployable packages in %ServiceVolume%\DeployablePackages that are more than 30 days old are deleted. The same timeline is also used to delete servicing-related logs. These logs are usually located in C:\Dynamics.

However, on dev/one-box machines, there is more flexibility. The number of retention days and the minimum disk space of the ServiceVolume and Logs drives can be customized.

- Under the **HKLM:\SOFTWARE\Microsoft\Dynamics\Deployment** registry key, you can create the following keys to customize when cleanup should occur. The automated cleanup task will consider these values.
    - **CutoffDaysForCleanup** – The number of days that old packages and logs should be retained. The default value is **30**.
    - **CutoffDiskSpaceLimitForPackages** – The minimum free disk space (in gigabytes [GB]) on the service volume drive where the package folder is located. For example, if the disk space is 200 GB, the cleanup task will remove the packages, based on the number of days.
    - **CutoffDiskSpaceLimitForLogs** – The minimum free disk space (in GB) of the system drive where the log folder is located. For example, if the disk space is 100 GB, the cleanup task will remove the servicing-related logs, based on the number of days.

Issue: A step failed with errors

Description

A step might fail with errors for one of the following reasons:

- The package has customization issues or is missing dependencies.
- There is an issue with the servicing scripts.
- A random failure occurred when the step was executed.

To verify what the issue is, follow these steps.

1. Download and navigate to the step logs.
2. Open the output file for the step, and see whether there are any errors.
3. If any additional log files are available, inspect them for errors.

Action

1. Download the logs.
2. Select **Rerun step** to retry the failed step.

If the step fails again, and the same error occurs, go back to the logs to look for more information.

- If you notice that there is an issue with the customizations, abort this package, and retry by using the new package.
- See whether a fix for the issue is available in Issue search.
- If you see the following step failure, either database synchronization or report deployment might have failed: "GlobalUpdate script for service model: AOSService"
- Look for the DBSync.err file, and see what the errors are. Inspect the DBSync.log file. For specific failures during the DB Sync step, look in the **Common DB Sync Failures** section.

Issue: The deployment status is Servicing but the servicing status is Failed

Description

A built-in mechanism enables the system to retry multiple times before it gives up. The following preparation steps can also be retried several times:

- Download the package to the machines.
- Slipstream the servicing package.
- Generate the runbook.

Action

You can download the logs to view the error and take action before all retries are exhausted. If the retry mechanism fails to go beyond the preparation stage, and you see a status of **Failed**, open a ticket so that the Microsoft team can investigate the issue further.

Issue: The dashboard doesn't open after package deployment is completed

Description

If the dashboard doesn't open after package deployment is completed, a run-time error might be occurring when the AOS machine is started.

Action

1. Start Event Viewer.
2. Navigate to **Applications and Services Logs** > **Microsoft** > **Dynamics** > **Ax-SystemRuntime** > **Operational Filter by errors**, see whether there are any errors, and investigate the errors as required.

Issue: A package application failed with error code : DSU#####

Description

You may receive an error stating **A critical component has encountered an error processing your**

request. Error code: DSU#####, or similar. The error code of **DSU#####** indicates that there's a temporary outage happening in the underlying Microsoft API. This type of outage may also impact database movement functionality.

Action

Microsoft is proactively monitoring the service status and this type of outage is expected to be mitigated shortly.

There's no impact on the status and the health of your environment.

If you experience this error when scheduling a service request or performing a database movement task, please try again at a later time.

Typical database synchronization issues

If you see the following step failure, a database synchronization issue might be occurring: "GlobalUpdate script for service model: AOSService"

Follow these steps to look for the DBSync.err file, find the errors, and inspect the DBSync.log file.

1. Download and navigate to the step logs.
2. Open the output file for the step, and see whether there are any errors.
3. If additional log files are available, inspect the logs for any errors.
4. The following table shows the failures that are seen most often and the action that you should take. The **Issue** column shows the error message that you will see in the dbsync.err file. The percentage sign (%) can be replaced with the metadata name of the table, field, index, and so on.

ISSUE	ACTION
%Table Sync Failed for Table%Create Unique Index%	This issue typically occurs when a unique index is created, but the data isn't unique. Fix the data before you run the step again.
%Application configuration sync failed.%Custom action sync failed with error:%	View the information in the error message and the call stack to determine the application code that is causing the issue.
%cannot be found from underlying query''s table%	This issue was fixed. For more information, refer to KB 4018815.
%Table Sync Failed for Table%Converting Field%	Follow the error message, fix the issue, and run the step again.
%failed because one or more objects access this column%	See whether the index is in the metadata. If the index is in the metadata, this issue is a SyncEngine product issue. If the index isn't in the metadata, remove the index from the SQL database before you run the step again.
%cannot be found from underlying data source''s table%	This issue was fixed. For more information, refer to KB 4018815.
'%Table Sync Failed for Table%' and errorMessage like '%There is already an object named%'	The Internal SqlDictionary table and SQL schema are out of sync. There isn't enough information in the logs to understand how this state was reached.

ISSUE	ACTION
%Table Sync Failed for Table%Column names in each table must be unique%	SqlDictionary entries for the table are corrupted, and the field is missing. There isn't enough information in the logs to understand how this state was reached.
%Column name 'LOAD_' does not exist in the target table or view. CREATE INDEX%	This issue appears to be a SyncEngine issue. Create a ticket on Microsoft Support.
Cannot drop the index 'VENDREQUESTPROFILEQUESTIONNAIRE.I_1301PROFILEQUESTIONNAIRE', because it does not exist or you do not have permission	This issue appears to be a SyncEngine issue. Create a ticket on Microsoft Support.
%The index entry of length 2046 bytes for the index 'I_65750INDEX1' exceeds the maximum length of 1700 bytes for nonclustered indexes%	Modify the index before you run the step again.
%Incorrect syntax near%	This issue is a SyncEngine issue. Create a ticket on Microsoft Support.
Reference to database and/or server name in 'TEMPDB.DBO.T_TRVREQUISITIONLINE_C4C3569DD5A14CDABAE71A341743FB61' is not supported in this version of SQL Server	This issue is a SyncEngine issue. Create a ticket on Microsoft Support.
%Error: Timeout expired. The timeout period elapsed prior to obtaining a connection from the pool.%	Retry the step.
Database execution failed: Invalid column name 'DEFAULTDIMENSION'. CREATE VIEW	This issue appears to be a SyncEngine issue. Create a ticket on Microsoft Support.
Database execution failed: Invalid object name 'PMBI_DEPROJECTTIMESHEET'. CREATE VIEW	This issue appears to be a SyncEngine issue. Create a ticket on Microsoft Support.
%provider: Named Pipes Provider, error: 40 - Could not open a connection to SQL Server%	This issue should have been fixed in Platform Update 3. Retry the step.

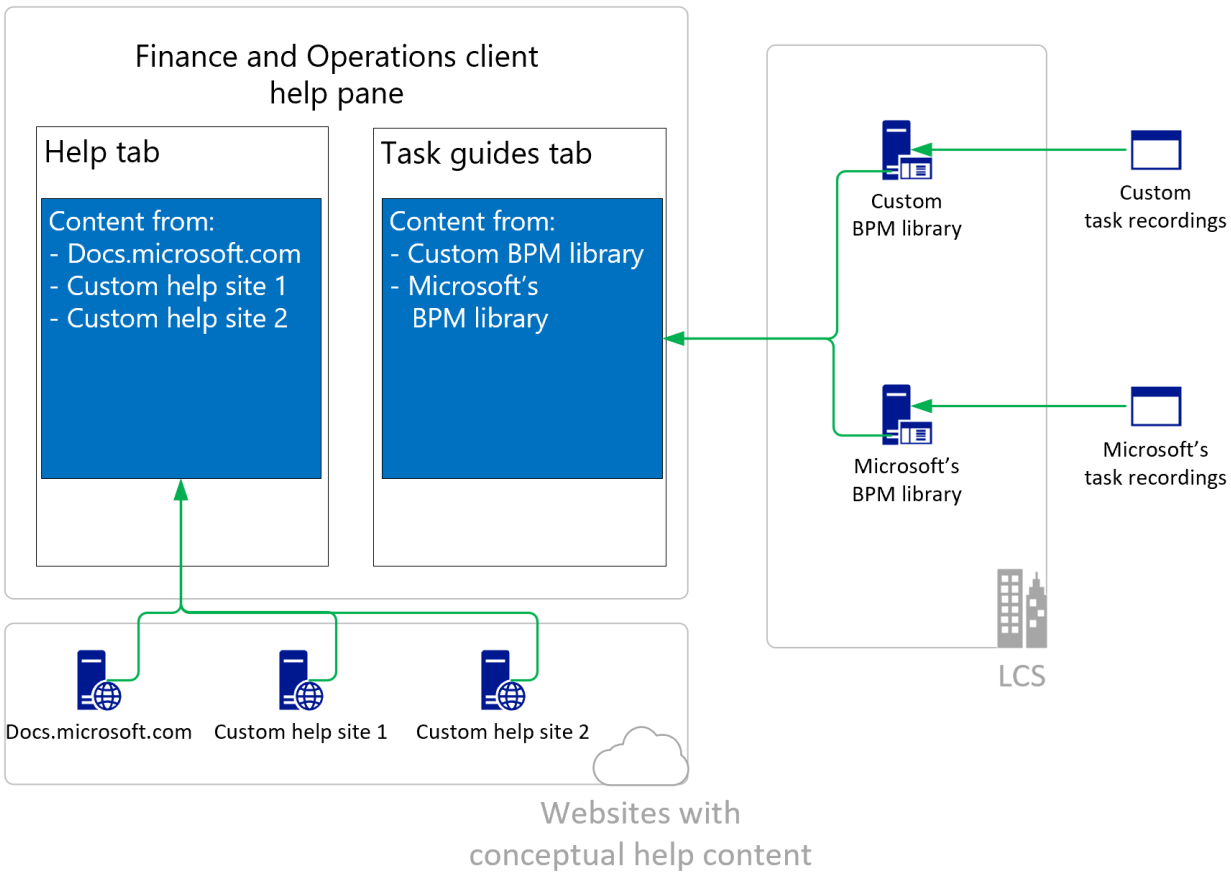
Custom Help overview

2/18/2021 • 4 minutes to read • Edit Online

Finance and Operations apps are often customized and extended to fit an organization's needs. If your solution is based on Microsoft Dynamics 365 Finance, Dynamics 365 Supply Chain Management, or Dynamics 365 Commerce, you can connect solution-specific and customer-specific Help content to the Help pane in the Finance and Operations client. This topic describes the main steps and decision points.

> **NOTE**
>
> Users of Finance and Operations apps can create custom task guides to supplement conceptual content that describes the functionality of their solution. These conceptual descriptions are also referred to as Help and can be provided by Microsoft, partners, and an organization itself. For more information, see Help system.

The following illustration, and this topic in general, use the term *Help* for conceptual descriptions that either include or exclude how-to guides. The term *task guides* refers to in-product task guides.



Custom Help content

Custom Help content typically originates from one of three sources:

- Microsoft documentation repositories (repos)

  You can use the HTMLFromRepoGenerator tool from the Custom Help Toolkit to clone content from any of the Finance and Operations repositories and generate corresponding HTML files. Those files can then be updated with content that is specific to your solution.

- Existing customized Dynamics AX content

  You can convert Dynamics AX custom Help content so that it can be used in Dynamics 365.

- HTML files that are created specifically for your solution

  Learn more about the metadata that must be added to your HTML files for context-sensitive Help and search to work correctly.

Process

The end-to-end process depends on the actual customer solution and the users' expectations. A typical process involves the following steps:

1. Create the custom Help content.
2. Publish the content on a website.
3. Index the content by using a search service.
4. Connect the custom **Help** pane to the website and the search service.

Microsoft provides a toolkit that can help you generate HTML files from the Microsoft Help repositories, generate JavaScript Object Notation (JSON) files for search services, and change the locale of HTML files so that it matches the locale of your solution.

You're welcome to share your knowledge by contributing to this documentation through the link at the bottom of the page or by joining the Dynamics 365 community.

The following table outlines the main objectives that admins typically have for configuring the Help experience.

OBJECTIVE	LEARN MORE
I want to give my users a customized in-product Help experience that reflects their actual solution.	See the Custom Help websites section of this topic and Create documentation or training with Task Recorder.
I want to use the Microsoft Help content as a baseline for Help content that is specific to my solution.	See Custom Help Toolkit: The HtmlFromRepoGenerator tool.
I want to contribute to the Microsoft Help content.	See Extend, customize, and collaborate on the Help.
I want to reuse my existing Dynamics AX content.	See Convert Dynamics AX custom Help for use in Dynamics 365.
I want to set up a website for my Help content.	See the Custom Help websites section of this topic.
I want to add my content to the **Help** pane.	See Connect a custom Help website to the Help pane.
Our technical writers want guidance that will help them convert our earlier content into Markdown so that it becomes easier for them to customize the Microsoft content.	See Moving to Markdown.

Custom Help websites

Before the product can connect to your Help content, you must customize the in-product **Help** pane so that it shows your content. The following conditions must be met:

- Your content must be available on a website.

  You can deploy your content to an existing website, or you can set up a dedicated website to host your

content. The website can be private or public, but we recommend that users **not** be required to sign in to access your content.

- Your content must be indexed by a search service.

  If you use the AzureSearchCustomHelp solution that is part of the Custom Help Toolkit for context-sensitive Help, the **Help** pane will generate a query that must be run against the search service's index. The query depends on specific metadata in the Help topics. For more information, see Metadata requirements for custom Help topics.

The Deploy custom Help to Azure topic describes an approach for hosting content on Azure. It includes information about how to set up a search service that indexes your content so that it can be found by the in-product **Help** pane. If you don't have an Azure subscription, create an account before you begin. You can start with a free account for 12 months. For more information, see Create your Azure free account today.

See also

Connect a custom Help website to the Help pane
Deploy custom Help to Azure
Custom Help Toolkit
Language and locale descriptors in the product and in Help
Configure the Help experience for Finance and Operations apps
Help system

Prepare content for use with the Help pane

2/18/2021 • 4 minutes to read • Edit Online

Content that can be used with the **Help** pane in Microsoft Dynamics 365 Finance, Dynamics 365 Supply Chain Management, and Dynamics 365 Commerce can be derived from existing Microsoft content, migrated from existing Dynamics AX 2012 Help content, or created as new files.

Microsoft creates Help in multiple languages for the locales that are supported by Finance, Supply Chain Management, and Commerce. However, locale support isn't restricted to those locales.

Creating custom Help content that is derived from existing Microsoft content

You can use Microsoft Help content as a baseline for content that describes your solution. The HtmlFromRepoGenerator tool can retrieve the content from Markdown files in Microsoft repositories (repos) and convert it to HTML files.

For more information about how to use existing Microsoft content as a baseline for content that describes your solution, see Extend, customize, collaborate on the Help.

Migrating content from existing AX 2012 Help content

If you have existing content from AX 2012, you can reuse it for Finance, Supply Chain Management, and Commerce. However, you must transform the HTML files so that they can be used in the custom Help environment. The Custom Help Toolkit includes a Windows PowerShell script, **run_ax2012.ps1**, that transforms AX 2012 HTML files so that they can be used in the custom Help environment.

Creating new Help content

You use the AzureSearchCustomHelp solution that is provided as part of the Custom Help Toolkit to connect your content to the **Help** pane. The **Help** pane will generate a query that is run against the search service's index. Context-sensitive Help and full-text search in the **AzureSearchCustomHelp** solution require that each topic contain specific metadata.

Metadata requirements for custom Help topics

The following metadata must be present in your topics for context-sensitive Help and full-text search to return results in the AzureSearchCustomHelp solution.

PROPERTY	DESCRIPTION
title	The value is used for full-text search from the **Help** pane.
description	The value is used for full-text search from the **Help** pane.
ms.search.form	The value contains the Application Object Tree (AOT) name of a page and is used for context-sensitive search from the **Help** pane.

PROPERTY	DESCRIPTION
ms.locale	The value indicates the language of the topic. It's mapped against the current browser locale when the **Help** pane searches the content. Language fallback can be configured for the target custom Help website. For more information, see Language and locale descriptors in the product and in Help.
ms.search.scope	The value determines which client the Help topic is shown in. You can specify one or more values. Values include **Core**, **Operations**, **Retail**, and **Human Resources**.

The following table describes the values that can be specified for the **ms.search.scope** property. You can specify one or more values. The values determine which client a Help topic is shown in.

VALUE	DESCRIPTION
Core	If this value is present, the topic appears in the **Help** pane. Otherwise, the topic doesn't appear in the **Help** pane. This value is set for the part of the Microsoft content that must always be available in the **Help** pane, for all users across all supported Dynamics 365 solutions.
Operations	This value applies to solutions that are based on Finance or Supply Chain Management.
Retail	This value applies to solutions that are based on Commerce.
Human Resources	This value applies to solutions that are based on Dynamics 365 Human Resources.
Talent	This value applies to solutions based on Dynamics 365 Talent. (Note that the Dynamics 365 Talent: Attract and Dynamics 365 Talent: Onboard apps are being retired. For more information, see Retiring Dynamics 365 Talent: Attract and Onboard apps.)

Non-required metadata

The following properties are reserved for future use:

- **ms.search.region** – Eventually, this property might be used to limit the content that is shown to content that is tagged either as global or for the region of the implementation.
- **ms.search.validFrom** – Eventually, this property might be used to limit the content that is shown to content for a product that was released on a given date or earlier.
- **ms.dyn365.ops.version** – Eventually, this property might be used to limit the content that is shown to content for a specific version of a product or earlier.
- **ms.search.industry** – Eventually, this property might be used to limit the content that is shown to content for a specific industry.

> **TIP**
> Microsoft content in the public GitHub repos contains additional metadata that Microsoft uses in internal processes that aren't related to the mechanics of the Help system. You can ignore these metadata properties if you extend or customize the Microsoft content.

Changing the locale of topics to match the locale of solutions

If your solution is intended to support multiple markets, you will want to provide Help content for each market. For example, your solution might support German (Germany) and German (Austria), but you have HTML files only for German (Germany). To make the same content available in German (Austria), you can make a copy of the HTML files and then use the HtmlLocaleChanger tool to update the **ms.locale** metadata. You can also add content that is specific to Austria to these new HTML files, as required.

Converting HTML files to JSON files for use with an Azure search service

If you use the AzureSearchCustomHelp solution that is provided as part of the Custom Help Toolkit, the search service requires that all your content be in JavaScript Object Notation (JSON) format. The ConvertHtmlToJson tool transforms HTML files into JSON files.

See also

Custom Help overview
Custom Help Toolkit
Language and locale descriptors in the product and in Help
Configure the Help experience for Finance and Operations apps
Help system

Deploy custom Help to Azure

2/18/2021 • 12 minutes to read • Edit Online

This topic describes the steps for setting up a web app to host your content and for setting up a search service to make the content discoverable by the in-product **Help** pane. You will set up a Microsoft Azure web app and then use that web app to host content that is connected to the in-product **Help** pane.

If you don't have an Azure subscription, create an account before you follow the steps in this topic. You can start with a free account for 12 months. For more information, see Create your Azure free account today.

Get started

The Prepare content for use with the Help pane topic describes the steps for preparing Help content so that it can be used with the in-product **Help** pane. After you have a set of HTML files and their equivalent JavaScript Object Notation (JSON) files, you can set up the web app and the search service.

Process overview

The general process for creating your Azure resources consists of the following steps:

1. In the Azure portal, create a resource group.
2. In the Azure portal, create a web app, a storage account, and a search service.
   - The web app stores and serves HTML files. The HTML files contain your Help content.
   - The storage account uses a blob container to store JSON files. The JSON files are your Help files after they have been converted to JSON format so that they can be used to generate an index of your content for search purposes. For more information, see Custom Help Toolkit: The ConvertHtmlToJson tool.
   - The search service indexes the Help content. An index makes your content discoverable by the in-product **Help** pane. For more information, see Create a basic index in Azure Cognitive Search.
3. Upload HTML files to the web app by using File Transfer Protocol (FTP).

   When you complete this step, you put the HTML files in the relevant language subfolders, based on the locales that the content was written for. For information about the names to use for these subfolders, see Language and locale descriptors in the product and in Help.
4. Upload JSON files into Azure Blob storage in the storage container, in subfolders that correspond to the language subfolders for the HTML files.
5. Configure the search service so that it has a data source, index, and indexer on the search service, by using the REST application programming interface (API).

   In this example, an API tool that is named Postman is used to make the REST API calls. To use a language-specific index analyzer, you must create language-specific indexes.

In the remaining sections of this topic, the assumption is that you have an Azure account and a valid subscription. If you don't have an Azure subscription, create an account before you begin. You can start with a free account for 12 months. For more information, see Create your Azure free account today.

Create a resource group

To host your web app, search service, and storage account, you must first create one or more resource groups. We recommend that you create all resources in a single resource group to make management easier. For more

information, see Create resource groups.

1. In the Azure portal, select **Resource groups**, select **Add**, and then specify a name for the resource group, such as **MyCustomHelp**.
2. Select **Review + Create** to finish creating the resource group.

Create a web app

To host your content, you must create a web app in Azure. For more information, see Create a static HTML web app in Azure.

Create the web app

1. In the Azure portal, go to your resource group, select **Add**, select **Web App**, and then specify the runtime stack and a name for the web app, such as **MyCustomHelpWebApp**.

   You can use any .NET Core stack as the runtime stack. For more information, see Create an ASP.NET Core web app in Azure.

2. After the deployment is completed, select **Go to resource**.

3. On the left side of the page, select **Deployment Center**. Under **Manual Deployment (push/sync)**, select **FTP**, and then select **Dashboard**.

   You can use either *app credentials* or *user credentials* to upload your content to the web app. We recommend that you use app credentials. To upload your content, you must have the FTP/FTPS endpoint, the user name, and the password.

   You might want to copy the user name and password to a temporary location before you continue. Additionally, we recommend that you reset the credentials after you've completed the deployment. For more information, see Configure deployment credentials for Azure App Service.

Next, you will add your HTML files to the web app. You can use an FTP client such as FileZilla, Visual Studio, Cyberduck, or WinSCP. For more information, see Deploy your app to Azure App Service using FTP/S.

Upload HTML files

1. Open your preferred FTP client. For information about best practices that are related to uploading files to a web app, see Deploy your app to Azure App Service using FTP/S.

2. Enter the host (the FTPS endpoint value from the Deployment Center for the web app), user name, and password, and then connect.

3. Under **/site/wwwroot** on the host, create a language folder for each language that your custom Help website must support. Upload the HTML files and other associated files to each language folder.

   > **IMPORTANT**
   > Remember to use folder names that correspond to the languages that the client expects. For more information, see Language and locale descriptors in the product and in Help.

Your custom Help website has now been deployed to Azure and should be visible in a browser.

Create a storage account

Next, create a storage account that uses a blob container to store the JSON files that the search service will use. For more information about Azure Storage, see the Azure Storage documentation.

You can generate the JSON files from your HTML Help files by using the ConvertHtmlToJson tool that is part of

the Custom Help Toolkit.

1. In the Azure portal, go to your resource group, select **Add**, select **Storage account**, and specify a name for the storage account, such as **mycustomhelpstorage**. Then select **Review + Create**. For more information, see Create a storage account.

2. Validate and create the storage account.

   After the deployment is completed, the new storage account is listed under the resource group.

3. Select your storage account, and then, on the left side of the page, under **Blob Service**, select **Containers**. Add a container, and specify a name, such as **mycustomhelpcontainer**. For more information, see Quickstart: Upload, download, and list blobs with the Azure portal.

You can now upload your JSON files. The folder structure that you use must match the folder structure that you created for the HTML files to match the languages of your solution. For example, if your web app has HTML files in an **en-US** folder, create an **en-US** folder in the container, and upload the en-US JSON files to this folder.

There are several ways to upload JSON files to the blob container. If you prefer to use a user interface (UI), Azure Storage Explorer is a convenient tool that lets you use Azure Storage to manage file operations. If you prefer a command-line option, you can use AzCopy. For more information, see Transfer data with the AzCopy on Windows.

Create a search service

Next, create a search service, so that the content can be indexed and is discoverable by the in-product **Help** pane. For more information, see What is Azure Cognitive Search?

- In the Azure portal, go to your resource group, select **Add**, and select **Azure Cognitive Search**, and then, under **URL**, specify a name for the service, such as **mycustomhelpsearch**. Then select **Review + Create**.

  The service is added to the resource group.

Configure the search service

In the previous section, you created a search service. In this section, you will configure it by creating a data source, an index, and an indexer for each locale. The JSON files that you uploaded to the blob container will then be indexed and searchable. In the examples that follow, the Postman tool is used to make several API calls. However, you can use your own method to call those APIs.

Create a data source

1. Open Postman, and create a new POST request. If you're unfamiliar with this tool, see Explore Azure Cognitive Search REST APIs using Postman.

2. In the **Enter request URL** field, enter `https://[AzureSearchServicename].search.windows.net/datasources?api-version=2017-11-11`. Replace **[AzureSearchServicename]** with the name of the search service that you created in the Create a search service section of this topic (for example, **mycustomhelpsearch**).

3. On the **Headers** tab, set **"Content-type"** to **application/json**, and set **api-key** to the key from your Azure Cognitive Search service. You can find the key in **Access keys** under **Settings** on the left side of the search service.

4. In the **Authorization** tab, set **Type** to **No Auth**.

5. On the **Body** tab, paste the following text.

```
{
    "name" : "[datasourcename]",
    "type" : "azureblob",
    "credentials" :
    {
        "connectionString" : "DefaultEndpointsProtocol=https;AccountName=
[StorageAccountName];AccountKey=[key];EndpointSuffix=core.windows.net"
    },
    "container" :
    {
        "name" : "[JSONStorageContainerName]"
    }
}
```

Replace the following parameters with the relevant values:

- **[datasourcename]** – Specify a name for the data source, such as **mycustomhelpdatasource**.
- **[StorageAccountName]** – Specify the name of the storage account that you created in the Create a storage account section (for example, **mycustomhelpstorage**).
- **[key]** – Specify the access key for your storage account. You can find the key in **Keys** under **Settings** on the left side of the storage account.
- **[JSONStorageContainerName]** – Specify the name of the blob container that you created in the Create a storage account section (for example, **mycustomhelpcontainer**).

6. Select **Send**, and make sure that the value in the **Status** field is **201 Created**.

Next, you will configure the search service so that it has an index of the content for each locale that you want to support.

Create an index

1. In Postman, create a new POST request that has the following parameters:
   - **URL**: `https://[AzureSearchServicename].search.windows.net/indexes?api-version=2017-11-11` (Replace **[AzureSearchServicename]** with the name of your search service.)
   - **Type** (on the **Authorization** tab): **No Auth**
   - **Content-Type** (on the **Headers** tab): **application/json**
   - **api-key** (on the **Headers** tab): The key from your Azure Cognitive Search service
2. On the **Body** tab, paste the following text.

```
{
    "name" : "[IndexName]",
    "fields": [
        { "name": "id", "type": "Edm.String", "key": true,"searchable": true, "filterable": true,
"sortable": true, "facetable": true },
        { "name": "title", "type": "Edm.String", "searchable": true, "filterable": true, "sortable":
true, "facetable": true, "analyzer":"[AnalyzerName]" },
        { "name": "description", "type": "Edm.String", "searchable": true, "filterable": true,
"sortable": true, "facetable": true, "analyzer":"[AnalyzerName]" },
        { "name": "ms_search_form", "type": "Edm.String", "searchable": true, "filterable": true,
"sortable": true, "facetable": true },
        { "name": "ms_search_region", "type": "Edm.String", "searchable": true, "filterable": true,
"sortable": true, "facetable": true },
        { "name": "ms_locale", "type": "Edm.String", "searchable": true, "filterable": true,
"sortable": true, "facetable": true },
        { "name": "metadata_storage_path", "type": "Edm.String", "searchable": true, "filterable":
true, "sortable": true, "facetable": true },
        { "name": "metadata_storage_name", "type": "Edm.String", "searchable": true, "filterable":
true, "sortable": true, "facetable": true },
        { "name": "metadata_storage_content_type", "type": "Edm.String", "searchable": true,
"filterable": false, "sortable": false, "facetable": false }
    ]
}
```

Replace the following parameters with the relevant values:

- **[IndexName]** – Specify the name of the index that should be created, such as **indexenus**.
- **[AnalyzerName]** – Specify the name of the language analyzer that should be used, such as **en.microsoft**.

> **NOTE**
>
> The index is language-specific. The **title** and **description** fields contain translations, and it's important that you set the corresponding language analyzer value. Use an appropriate value, based on the language of the index that you're creating. For a list of language analyzers, see Analyzer List.

3. Select **Send**, and make sure that the **Status** field is set to **201 Created**.
4. If you prepared custom Help content for multiple languages, repeat these steps to create a unique index for each language.

The index isn't an index until it has been processed by an indexer. Think of the table of contents in a reference book. It would not really be useful unless it also lists the page numbers for where to find the various sections in the book. Similarly, the indexer for your search service fills in the index, based on a search. For more information, see Indexers in Azure Cognitive Search.

Create an indexer

1. In Postman, create a new POST request that has the following parameters:
   - **URL**: `https://[AzureSearchServicename].search.windows.net/indexers?api-version=2017-11-11` (Replace **[AzureSearchServicename]** with the name of your search service.)
   - **Type** (on the **Authorization** tab): **No Auth**
   - **Content-Type** (on the **Headers** tab): **application/json**
   - **api-key** (on the **Headers** tab): The key from your Azure Cognitive Search service
2. On the **Body** tab, paste the following text.

```
{
    "name" : "[IndexerName]",
    "dataSourceName" : "[DatasourceName]",
    "targetIndexName" : "[IndexName]",
    "schedule" : { "interval" : "PT10H" },
    "parameters" : { "configuration" : { "parsingMode" : "json" } },
    "fieldMappings" : [
        { "sourceFieldName" : "/title", "targetFieldName" : "title" },
        { "sourceFieldName" : "/ms.search.form", "targetFieldName" : "ms_search_form" },
        { "sourceFieldName" : "/ms.search.region", "targetFieldName" : "ms_search_region" },
        { "sourceFieldName" : "/ms.locale", "targetFieldName" : "ms_locale" },
        { "sourceFieldName" : "/description", "targetFieldName" : "description" }
    ]
}
```

Replace the following parameters with the relevant values:

- **[IndexerName]** – Specify the name of the indexer that should be created, such as **indexerenus**.
- **[DatasourceName]** – Specify the name of the data source that you created, such as **mycustomhelpdatasource**.
- **[IndexName]** – Specify the name of the index that you created, such as **indexenus**.

> **NOTE**
>
> This configuration will set up an indexer that is scheduled to run every 10 hours (**"schedule" : { "interval" : "PT10H" }**). However, you can adjust the interval as appropriate.

3. Select **Send**, and make sure that the **Status** field is set to **201 Created**.
4. If you prepared custom Help content for multiple languages, repeat these steps to create a unique indexer for each index.

> **NOTE**
>
> An index will contain data only after its indexer has run. You might want to manually run the indexer if you're planning to test the index immediately.

> **IMPORTANT**
>
> If you update your content, remember to regenerate the JSON files and upload them to the storage service. If you add content, you can either manually run the indexers or wait for the next scheduled run. Your updated content won't be available in search results until the next time that the indexers are run.

You can optionally use Postman to test the search. For an example that shows how to use Postman for testing, see Search your JSON files.

Next steps

If you completed all the steps in this topic, your Help content has now been uploaded to an Azure web app, and it has been indexed.

The next step is to extend the **Help** pane so that it can detect your content. In that way, when users open the **Help** pane in your Dynamics 365 solution, the in-product **Help** pane will be able to generate context-sensitive links to your Help. For more information, see Connect a custom Help website to the Help pane.

See also

Custom Help overview
Custom Help Toolkit
Language and locale descriptors in the product and in Help

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Connect a custom Help website to the Help pane

2/18/2021 • 4 minutes to read • Edit Online

If you deliver custom Help content for a Finance and Operations solution, you can extend the **Help** pane so that it consumes that content. You complete this one-time configuration by using the Finance and Operations development environment in Microsoft Visual Studio. After you've finished, users can select among tabs for task guides, Microsoft Help content, and your Help content.

The process for connecting your custom Help website to the in-product **Help** pane involves the following steps:

1. Extend the **Help** pane in Visual Studio.
2. Assign an index to a language.
3. Customize language fallback.

> **IMPORTANT**
> The procedures that follow require the development tools for Finance and Operations apps in Visual Studio. For more information, see Development tools in Visual Studio.

Extend the Help pane and assign the custom Help indexes to languages

The **Help Pane extension** folder of the Custom Help Toolkit contains the **AzureSearchCustomHelp** solution that you can open in the Finance and Operations development environment. That folder also contains the **HelppaneOption.axpp** project that you can then import into the solution in Visual Studio.

Extend the Help pane

1. In the Finance and Operations development environment, open the **AzureSearchCustomHelp.sln** solution.
2. On the **Dynamics 365** menu, select **Import project**.
3. In the **File name** field, specify the path of the **HelppaneOption.axpp** project, and then select **OK** to complete the import process. Update the references so that no references are missing.
4. In the **HelppaneMacro** file, update the values of the following parameters:
   - **[WebAppName]** – Specify the name of the web app that you created in Create a web app. For example, specify **MyCustomHelpWebApp**.
   - **Admin key value** – Specify the admin key for the Azure Cognitive Search service. You can find the key in **Access keys** under **Settings** on the left of the search service in the Azure portal.
   - **[SearchServiceName]** – Specify the name of the search service that you created in Create a search service. For example, specify **mycustomhelpsearch**.

   The following example shows the content of the HelppaneMacro file.

```
#define.webApp('http://[WebAppName].azurewebsites.net/')
#define.queryApiKey('Admin key value')
#define.defaultstring('dashboard')
#define.searchservicename('[SearchServiceName]')
#define.CustomResultError('error')
#define.CustomTabPage('CustomTabPage')
#define.CustomHelp('Custom Help')
#define.CustomTitle('CustomTitle')
#define.htm('html')
```

5. Optional: If you want to change any of the user interface (UI) strings that appear in the **Help** pane, edit the **Customhelppane.en-US.label.txt** file.

Next, you must specify the language that the search index for your custom Help is intended for.

Assign a custom index to a language

1. Open the **Language.config** file in the solution.
2. In the list, find the language of the index, and specify an index name by using the **index=""**, **parentindex=""**, or **ultimateindex=""** key.

   For example, you created search indexes for English (United States) and German (Austria), and you named them **myenusindex** and **mydeatindex**, respectively. In this case, here is what your entries will look like.

   ```
   <add language="en-US" ultimateindex="myenusindex" />
   <add language="de-AT" parentlanguage="de" index="mydeatindex" />
   ```

3. Optional: Customize language fallback for your index, as described in the next section.
4. Build the **AzureSearchCustomHelp** solution.

The result is a model that you upload to the Asset library of the customer project or the solution project in Microsoft Dynamics Lifecycle Services (LCS).

Customize language fallback

Language fallback means that the **Help** pane runs a search in additional languages if the intended language either doesn't return a result or doesn't exist.

> **NOTE**
> A custom index must be available for the additional languages.

The search and fallback order have the following order of priority:

1. The language that is set in the client (for example, **de-AT**)
2. The language that is defined in the **parentlanguage** attribute for that language (for example, **<add language="de-AT" parentlanguage="de" index="mydeindex" />**)
3. The language that the **ultimateindex** attribute is set for (for example, **<add language="en-US" ultimateindex="myenusindex" />**)

> **IMPORTANT**
> If the **parentlanguage** attribute is set, there must be a corresponding **parentindex** key.

The following scenario is valid, because **language="de"** has **parentindex="indexde"**, and both **de-DE** and **de-AT** are descendants of **de**.

```
<add language="de" parentindex="indexde"/>
<add language="de-DE" parentlanguage="de" index=""/>
<add language="de-AT" parentlanguage="de-DE" index="indexdeat"/>
```

For more information about languages, see Languages, translations, and adaptations.

The following sections provide sample configurations.

Help content for one locale

In this configuration, you have Help content only for English (United States). Regardless of the locale that clients are set to, they will show the Help content in English (United States).

```
<add language="en-US" ulitmateindex="indexenus"/>
```

Help content for multiple locales

In this configuration, you have Help content for French, German, and English (United States). Clients that are set to the **de** locale will show the Help content in German, clients that are set to the **fr** locale will show the content in French, and clients that are set to any other locale will show the content in English (United States).

```
<add language="en-US" ulitmateindex="indexenus"/>
<add language="fr" parentindex="indexfr"/>
<add language="de" parentindex="indexde"/>
```

If clients are set to the **de** or **fr** locale, but no results are found in the German or French content, respectively, results will be shown in English (United States), if content is available in that language.

Help content that uses parent locales

In this configuration, you have Help content for German, German (Austria), and English (United States). For example, you have several topics that are related specifically to features for Austria, but topics in German can be used otherwise.

```
<add language="en-US" ulitmateindex="indexenus"/>
<add language="de" parentindex="indexde"/>
<add language="de-AT" parentlanguage="de" index="indexdeat"/>
```

If the client is set to the **de-AT** locale, but no results are found in the German (Austria) content, results will be shown in German and English (United States), if content is available in those languages.

See also

Deploy custom Help to Azure
Custom Help Toolkit
Language and locale descriptors in the product and in Help
Configure the Help experience for Finance and Operations apps
Help system

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Custom Help Toolkit

2/18/2021 • 2 minutes to read • Edit Online

Microsoft has published a GitHub repository (repo) that includes scripts and tools that can help you prepare context-sensitive Help for customized Finance and Operations solutions. This context-sensitive Help can be accessed from the in-product **Help** pane.

Tools in the toolkit

The Custom Help Toolkit is available at https://github.com/microsoft/dynamics365f-o-custom-help. The repo contains the following tools and the source code for those tools:

- The **HtmlFromRepoGenerator** tool

  For more information, see Custom Help Toolkit: The HtmlFromRepoGenerator tool.

- The **ConvertHtmlToJson** tool

  For more information, see Custom Help Toolkit: The ConvertHtmlToJson tool.

- The **HtmlLocaleChanger** tool

  For more information, see Custom Help Toolkit: The HtmlLocaleChanger tool.

- The **Help Pane extension** Microsoft Visual Studio project

  For more information, see Connect a custom Help website to the Help pane.

- Dynamics AX 2012 metadata scripts

  For more information, see Convert Dynamics AX custom Help for use in Dynamics 365.

> **NOTE**
>
> The first version of the toolkit is available as a release in the GitHub repo.

See also

Custom Help overview
Deploy custom Help to Azure
Connect a custom Help website to the Help pane
Language and locale descriptors in the product and in Help
Convert Dynamics AX custom Help for use in Dynamics 365

Custom Help Toolkit: The HtmlFromRepoGenerator tool

2/18/2021 • 5 minutes to read • Edit Online

The Custom Help Toolkit includes the **HtmlFromRepoGenerator** tool, which gets Microsoft content in Markdown files and converts it to HTML files. You can then deploy the HTML files to a website.

Use the HtmlFromRepoGenerator tool to get Markdown files and generate HTML files

The **HtmlFromRepoGenerator** tool provides functionality that supports the creation of custom Help that is based on source files from Microsoft. You can use the tool to perform these tasks:

- Clone a Microsoft documentation repository (repo).
- Remove developer and admin content from your clone of the Microsoft repo.
- Update links to files that are no longer present in the clone.
- Update the value of the **ms.locale** property so that it matches the language options that are supported by the Finance and Operations client.

  The language descriptors that the client uses differ from the language descriptors that are used in the corresponding GitHub repos. Before localized custom Help can be called, the language descriptors in the source content must be changed so that they match the client's languages. For more information, see Language and locale descriptors in the product and in Help.
- Generate HTML files that can be used to publish content.

  The HTML files will be generated in the **d365F-O** subfolder. The files are generated based on style sheets and templates that are part of the tool. For more information, see the Modifying the styling of the generated HTML files section of this topic.
- Compare a localized Microsoft repo to the en-US repo to identify differences and update links accordingly.

In the first version of the Custom Help Toolkit, this tool was named ConsoleApp.

Syntax

Here is the syntax for running HtmlFromRepoGenerator.exe.

```
HtmlFromRepoGenerator.exe --Json <Articles/> --Out <path> --ExternalText <text> [--DoNotClone <true|false>]
[--Repo <URL>] [--RemoveGitFolder <true|false>] [--ReplaceUrl <URL>] [--LogsDir <.\logs>] [--EnRepo <URL>]
[--EnOut <path>] [--Lng <language code>] [--Rtl] [--?[--]]
```

Here is an explanation of the parameters.

PARAMETER	DESCRIPTION
Json	Specify a relative path for the location of the **docfx.json** file. In Microsoft documentation repos, this location is typically **articles/**.

PARAMETER	DESCRIPTION
Out	Specify the folder where your existing cloned repo exists, or the folder to clone the repo to. If you run the **HtmlFromRepoGenerator** tool to clone a repo, this folder must not already exist. Use the language name as the folder name, as described in Language and locale descriptors in the product and in Help.
ExternalText	Specify text that should be added to the updated links if the **HtmlFromRepoGenerator** tool must replace the original links.
DoNotClone	Set this parameter when you run the tool against a previously cloned repo.
Repo	Specify the repo URL. This parameter is optional if you run the tool against a previously cloned repo. Examples of URLs for Microsoft documentation repos include `https://github.com/MicrosoftDocs/Dynamics-365-Unified-Operations-public` for English (United States) and `https://github.com/MicrosoftDocs/Dynamics-365-Operations.de-de` for German (Germany).
RemoveGitFolder	Specify whether the **.git** folder should be removed.
ReplaceUrl	Specify the URL that should replace links between files when the target files aren't present. This parameter is intended to be used to turn relative links into absolute links.
LogsDir	Specify the folder to save logs files to.

The following additional parameters are used when the tool is run against the localized Microsoft documentation repos.

PARAMETER	DESCRIPTION
EnRepo	Specify the URL of the en-US repo. This parameter is optional if you run the tool against a previously cloned repo. The URL of the Microsoft documentation repo for English (United States) is `https://github.com/MicrosoftDocs/Dynamics-365-Unified-Operations-public` .
EnOut	Specify the folder where the en-US repo exists, or the folder to clone it to. If you run the tool against a previously cloned repo, this folder must not already exist.
Lng	Specify the language value that should be used for **ms.locale** metadata in the generated HTML files. The value must correspond to the value that is specified in the language settings of the Finance and Operations client. If this parameter isn't set, the tool uses **en-US**. For more information, see Language and locale descriptors in the product and in Help.

PARAMETER	DESCRIPTION
Rtl	Include this parameter if the language uses right-to-left (RTL) formatting. Examples of RTL languages include Arabic and Hebrew.

Examples

> **NOTE**
> Because the Microsoft repos contain many files, the process takes several minutes. If you run the tool against multiple localized repos, the process takes longer.

The following example clones the en-US repo and generates HTML files for en-US.

```
HtmlFromRepoGenerator.exe --json articles/ --out "D:\D365-Operations\en-US" --repo
"https://github.com/MicrosoftDocs/Dynamics-365-unified-Operations-public" --externalText "(This is an
external link)" --replaceUrl "https://docs.microsoft.com/en-us/dynamics365/supply-chain" --LogsDir D:\D365-
Operations\logs\en-US
```

The following example uses a previously cloned en-US repo and generates HTML files for en-US.

```
HtmlFromRepoGenerator.exe --json articles/ --out "D:\D365-Operations\en-US" --externalText "(This is an
external link)" --replaceUrl "https://docs.microsoft.com/en-us/dynamics365/supply-chain" --LogsDir D:\D365-
Operations\logs\en-US
```

The following example clones the de-DE and en-US repos, and generates HTML files for de.

```
HtmlFromRepoGenerator.exe --json articles/ --out "D:\D365-Operations\de" --repo
"https://github.com/MicrosoftDocs/Dynamics-365-Operations.de-de" --externalText "(This is an external link)"
--EnRepo "https://github.com/MicrosoftDocs/Dynamics-365-unified-Operations-public" --EnOut "D:\D365-
Operations\en-us" --replaceUrl "https://docs.microsoft.com/de-de/dynamics365/supply-chain" --lng "de" --
LogsDir D:\D365-Operations\logs\de
```

The following example uses the existing de-DE and en-US repos, and generates HTML files for de. If you use the existing de-DE repo, make sure that it's up to date.

```
HtmlFromRepoGenerator.exe --json articles/ --out "D:\D365-Operations\de" --DoNotClone --externalText "(This
is an external link)" --enOut "D:\D365-Operations\en-us" --replaceUrl "https://docs.microsoft.com/de-
de/dynamics365/supply-chain" --lng "de" --LogsDir D:\D365-Operations\logs\de
```

> **IMPORTANT**
> Because the **HtmlFromRepoGenerator** tool changes links during processing, don't run HtmlFromRepoGenerator.exe more than one time against a previously cloned repo. If it's run more than one time against the same content, links will be incorrect. If you want to rerun HtmlFromRepoGenerator.exe, either use the **HtmlFromRepoGenerator** tool to create a new clone of the repo, or revert all local changes that have been made to your existing clone.

Modifying the styling of the generated HTML files

The HTML files that the **HtmlFromRepoGenerator** tool generates are based on a set of predefined templates. In most cases, you can edit the style sheets in the **d365F-O\styles** folder to change the appearance of your

content.

For advanced scenarios, you can change the templates that the **HtmlFromRepoGenerator** tool uses. The source files are included in the **SourceCode** folder in the GitHub repo. The templates are in the **SourceCode\HtmlFromRepoGenerator\HtmlFromRepoGenerator\HtmlFromRepoGenerator\Resources** subfolder.

> **NOTE**
> If you change the templates, you must rebuild HtmlFromRepoGenerator.exe.

For more information, see Introduction to DocFX Template System.

See also

Custom Help Toolkit
Custom Help overview
Deploy custom Help to Azure
Language and locale descriptors in the product and in Help
Convert Dynamics AX custom Help for use in Dynamics 365

Custom Help Toolkit: The ConvertHtmlToJson tool

2/18/2021 • 2 minutes to read • Edit Online

The Custom Help Toolkit includes the **ConvertHtmlToJson** tool, which converts HTML files to JavaScript Object Notation (JSON) files. The search service uses the JSON files to index Help content.

Use the ConvertHtmlToJson tool to generate JSON files

The **ConvertHtmlToJson** tool transforms HTML files into JSON files. You can then add the JSON files to the Microsoft Azure Search service, which will generate context-sensitive links to your Help content.

The JSON files include metadata that the indexer uses to identify the form and language that the target Help page is intended for.

Here is the syntax for running ConvertHtmlToJson.exe.

```
ConvertHtmlToJson.exe --h <path> -j <path> --v <true|false>
```

Here is an explanation of the parameters.

PARAMETER	DESCRIPTION
h	Specify the path of the HTML files to process.
j	Specify the folder to save the JSON files to. The specified folder must already exist.
v	Set this parameter to **true** to turn on verbose logging. Otherwise, set it to **false**.

Example

The following example generates JSON files without verbose logging.

```
ConvertHtmlToJson.exe --h D:\D365-Operations\d365F-O\supply-chain\de -j D:\D365-Operations\json\supply-chain\de
```

See also

Custom Help overview
Deploy custom Help to Azure
Language and locale descriptors in the product and in Help
Convert Dynamics AX custom Help for use in Dynamics 365

Custom Help Toolkit: The HtmlLocaleChanger tool

2/18/2021 • 2 minutes to read • Edit Online

The Custom Help Toolkit includes the **HtmlLocaleChanger** tool, which can process HTML files that are generated by the HtmlFromRepoGenerator tool tool.

Use the HtmlLocaleChanger tool to align locales

The **HtmlLocaleChanger** tool can update your HTML files by setting a new value for the **ms.locale** property. For example, you have HTML files for German (Germany), and you want to make the same content available in German (Austria). In this case, you can run the tool to change the setting from **ms.locale: de-de** to **ms.locale:de-at**.

Here is the syntax for running HtmlLocaleChanger.exe.

```
HtmlLocaleChanger.exe --h <path> --l <locale> --v <true|false>
```

Here is an explanation of the parameters.

PARAMETER	DESCRIPTION
h	Specify the path of the HTML files to process.
l	Specify the new locale for the HTML files.
v	Set this parameter to **true** to turn on verbose logging. Otherwise, set it to **false**.

Example

The following example changes the locale to **de-at** and turns on verbose logging.

```
HtmlLocaleChanger.exe --h D:\D365-Operations\d365F-O\supply-chain\de --l de-at --v
```

See also

Custom Help overview
Deploy custom Help to Azure
Language and locale descriptors in the product and in Help
Convert Dynamics AX custom Help for use in Dynamics 365

Convert Dynamics AX custom Help for use in Dynamics 365

2/18/2021 • 2 minutes to read • Edit Online

If you have existing content from Microsoft Dynamics AX 2012, you can reuse it for Dynamics 365 Finance, Dynamics 365 Supply Chain Management, and Dynamics 365 Commerce. However, you must first transform the HTML files so that they can be used in the custom Help environment.

Converting AX 2012 content

The Microsoft Custom Help Toolkit includes a Windows PowerShell script, **run_ax2012.ps1**, that can transform AX 2012 HTML files so that they can be used in the custom Help environment. The script makes the following changes to the AX 2012 HTML files:

- Replace the **Microsoft.Help.F1** metadata name with **ms.search.form**.
- Replace the **Title** metadata name with **title**.
- Change the file name extension from **.htm** to **.html**.
- Add the following metadata.

```
<meta name="ms.search.region" content="Global" />
<meta name="ms.search.scope" content="Operations, Core" />
<meta name="ms.dyn365.ops.version" content="AX 7.0.0" />
<meta name="ms.search.validFrom" content="2016-05-31" />
<meta name="ms.search.industry" content="cross" />
```

Running the script

You can run the following command from a Command Prompt window, or you can run the script directly in Windows PowerShell.

```
PowerShell.exe -File run_ax2012.ps1 "Original file location" "New file location"
```

The following metadata is currently used, or it's reserved so that it can be used during indexing.

```
<meta name="title" content="Title of file" />
<meta name="ms.locale" content="locale" />
<meta name="ms.search.form" content="FormAOTName" />
<meta name="description" content="Description of file" />
<meta name="ms.search.region" content="Global" />
<meta name="ms.search.scope" content="Operations, Core" />
<meta name="ms.dyn365.ops.version" content="AX 7.0.0" />
<meta name="ms.search.validFrom" content="2016-05-31" />
<meta name="ms.search.industry" content="cross" />
```

For a description of the required metadata, see Metadata requirements for custom Help topics.

Moving to Markdown

To convert your existing content to Markdown, you can use various third-party tools, such as PanDoc and the

Writage plug-in for Word.

After you've converted your content to Markdown, you can use open-source tools such as DocFx to generate content for your website. In general, by working in Markdown, you gain access to a wide range of open-source tools. For more information, see Extend, customize, and collaborate on the Help.

See also

Custom Help overview
Custom Help Toolkit
Extend, customize, and collaborate on the Help

Language and locale descriptors in the product and in Help

2/18/2021 • 2 minutes to read • Edit Online

The client that Finance and Operations apps use supports multiple languages and locales. To add custom Help content for one or more locales to the in-product **Help** pane, you must make sure that both the following conditions are met:

- The value of the **ms.locale** property in each HTML file matches the locale of the content.

  For example, the German (Germany) content must have a setting of **ms.locale: de-de**.

- On the custom Help website, the content is in a folder has the same name as the locale.

  For example, the German (Germany) content must be in a folder that is named **de-de**.

For more information, see Custom Help overview and Deploy custom Help to Azure.

Languages and descriptors

The following table maps the language names between the Finance and Operations client and the GitHub repositories (repos) that contain translated Microsoft Help content.

LANGUAGE/LOCALE IN THE CLIENT	LANGUAGE/REGION NAME	NAME OF THE GITHUB REPO
ar	Arabic (Saudi Arabia)	Dynamics-365-Operations.ar-sa
ar-ae	Arabic (United Arab Emirates)	Dynamics-365-Operations.ar-ae
cs	Czech	Dynamics-365-Operations.cs-cz
da	Danish	Dynamics-365-Operations.da-dk
de	German (Germany)	Dynamics-365-Operations.de-de
de-at	German (Austria)	Dynamics-365-Operations.de-de
de-ch	German (Switzerland)	Dynamics-365-Operations.de-de
en-au	English (Australia)	Dynamics-365-Unified-Operations-Public
en-ca	English (Canada)	Dynamics-365-Unified-Operations-Public
en-gb	English (United Kingdom)	Dynamics-365-Unified-Operations-Public
en-ie	English (Ireland)	Dynamics-365-Unified-Operations-Public

LANGUAGE/LOCALE IN THE CLIENT	LANGUAGE/REGION NAME	NAME OF THE GITHUB REPO
en-in	English (India)	Dynamics-365-Unified-Operations-Public
en-my	English (Malaysia)	Dynamics-365-Unified-Operations-Public
en-nz	English (New Zealand)	Dynamics-365-Unified-Operations-Public
en-sg	English (Singapore)	Dynamics-365-Unified-Operations-Public
en-us	English (US)	Dynamics-365-Unified-Operations-Public
en-za	English (South Africa)	Dynamics-365-Unified-Operations-Public
es	Spanish (Spain)	Dynamics-365-Operations.es-es
es-mx	Spanish (Mexico)	Dynamics-365-Operations.es-es
et	Estonian	Dynamics-365-Operations.et-ee
fi	Finnish	Dynamics-365-Operations.fi-fi
fr	French (France)	Dynamics-365-Operations.fr-fr
fr-be	French (Belgium)	Dynamics-365-Operations.fr-fr
fr-ca	French (Canada)	Dynamics-365-Operations.fr-fr
fr-ch	French (Switzerland)	Dynamics-365-Operations.fr-fr
hu	Hungarian	Dynamics-365-Operations.hu-hu
is	Icelandic	Dynamics-365-Operations.is-is
it	Italian	Dynamics-365-Operations.it-it
it-ch	Italian (Switzerland)	Dynamics-365-Operations.it-it
ja	Japanese	Dynamics-365-Operations.ja-jp
lt	Lithuanian	Dynamics-365-Operations.lt-lt
lv	Latvian	Dynamics-365-Operations.lv-lv
nb-no	Norwegian Bokmål	Dynamics-365-Operations.nb-no
nl	Dutch (Netherlands)	Dynamics-365-Operations.nl-nl

LANGUAGE/LOCALE IN THE CLIENT	LANGUAGE/REGION NAME	NAME OF THE GITHUB REPO
nl-be	Dutch (Belgium)	Dynamics-365-Operations.nl-nl
pl	Polish	Dynamics-365-Operations.pl-pl
pt-br	Portuguese (Brazil)	Dynamics-365-Operations.pt-br
ru	Russian	Dynamics-365-Operations.ru-ru
sv	Swedish	Dynamics-365-Operations.sv-se
th	Thai	Dynamics-365-Operations.th-th
tr	Turkish	Dynamics-365-Operations.tr-tr
zh-hans	Chinese	Dynamics-365-Operations.zh-cn

Languages, translations, and adaptations

Microsoft teams create content in English (United States). That content is then translated into several languages, and the translated content is made available in a public GitHub repo for each language.

To maximize reuse of translations, translation services treat some languages as variants of another language. For example, German for Austria and German for Germany are considered so closely related that they are treated as variants of each other. Therefore, you can use the files in the Dynamics-365-Operations.de-de GitHub repo as a starting point for both German (Germany) content and German (Austria) content.

When you extend the in-product **Help** pane, you must assign indexes for the relevant locales. For more information, see Customize language fallback.

See also

Custom Help overview
Custom Help Toolkit

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Extend, customize, and collaborate on the Help

2/18/2021 • 5 minutes to read • Edit Online

The source files for the Microsoft Help for Finance and Operations apps are available in public GitHub repositories (repos). Any solution provider can easily extend and customize the content for specific solutions. This topic explains how to work with the GitHub repos and Markdown files.

For information about how to create Markdown files in GitHub repos, see the Docs contributor guide. For information about how to deploy custom Help, see Custom Help overview.

Contribute to the content

One benefit of GitHub is that you can contribute to the core content that the Microsoft team provides in the MicrosoftDocs/Dynamics-365-Unified-Operations-public repo. For example, you have a new topic that you think will be helpful to other users, or you have a correction to an existing topic. If you want to contribute to the Dynamics-365-Unified-Operations-public repo, you can create a *pull request* from your repo to the Dynamics-365-Unified-Operations-public repo. The Microsoft team will then review the request and include your changes as appropriate.

You can also contribute and make edits to the existing documentation. To get started, select the **Edit** button (pencil symbol) in a topic. The following video shows how you can contribute to the Microsoft documentation.

> **NOTE**
> Microsoft currently accepts pull requests only to the Dynamics-365-Unified-Operations-public repo, not to the language-specific repos. If you have feedback about translations, you can report a GitHub issue in the relevant repo.

Extend and customize Microsoft source content from GitHub repos

Microsoft uses separate repos in GitHub for the source content and for each language that Microsoft translates content into. The Dynamics-365-Unified-Operations-public repo contains the source content in English (United States). If you want to access the content in other languages, the names follow the pattern **Dynamics-365-Operations.<language>-<country>**. For example, the version for German (Germany) is named Dynamics-365-Operations.de-de.

When Microsoft publishes an update to the content, the *live* branch in the corresponding GitHub repo is updated. The source repo is updated weekly. However, the related language-specific repos are updated less often. The frequency depends on when new translations are made available. If you fork one of the Microsoft repos, you can choose to update your fork with updates from the Microsoft repo on a monthly basis or less often, depending on your preferred work processes. The GitHub platform and tooling will help you manage any potential merge conflicts if you change files that Microsoft has also changed. For more information, see Set up Git repository locally for documentation in the Docs authoring guide and Fork a repo in the Help for GitHub.

> **TIP**
> If you just want to get the Microsoft content as it is, you don't have to be familiar with GitHub. For more information, see the Get the content without a GitHub account section of this topic. However, if you want to extend or customize the Microsoft content, we recommend that you join Microsoft on GitHub.

Get started with GitHub

To join Microsoft in the world of GitHub and Markdown, you must be familiar with some new terminology and tools. The following list outlines the main steps, but you can find additional content, tools, and ideas in the GitHub documentation and other forums.

1. Sign up for GitHub.

   For more information, see GitHub account setup and Install content authoring tools in the Docs contributor guide.

2. Fork the appropriate repo.

   To extend and customize Microsoft content for a custom Help solution, you must create a fork of the repo. If you want to customize Microsoft content in Markdown format, we recommend that you manually fork the relevant repo and use your favorite Markdown editor. For more information, see Set up Git repository locally for documentation and Git and GitHub essentials for Docs in the Docs contributor guide.

   > **TIP**
   > You aren't required to make your GitHub repos public. When you fork a public repo, in the settings for the new repo, you can specify whether the repo is public, private, or available only to specific GitHub accounts.

Markdown format

The syntax that is used to format text for topics is named Markdig Flavored Markdown. This syntax complies with CommonMark. To learn more about how to work with Markdown, see Getting started with writing and formatting on GitHub.

You can convert content from Microsoft Word to Markdown by using open-source tools or other tools. In this way, you can easily recycle content.

Get updates from Microsoft

Microsoft makes frequent changes to the content, and those changes show up in the public GitHub repos. The base repo, MicrosoftDocs/Dynamics-365-Unified-Operations-public, is updated weekly. However, you can choose to get updates monthly, twice a year, or once a year, for example. The translation repos are updated less frequently, so you might want a monthly schedule or less frequent updates, as appropriate.

When you decide that it's time to get the latest version of the content from Microsoft, you can use the Git command line or GitHub Desktop. The Help for GitHub provides an example that shows how this process works in GitBash. In GitHub Desktop, you use the **Merge into current branch** command to pull changes from the origin into your fork.

If your solution is available in more than one country or region, you will probably want to make the content available in multiple languages. Although Microsoft has a GitHub repo for each supported language, the configuration files are available only in the English (United States) version of the base repo, MicrosoftDocs/Dynamics-365-Unified-Operations-public. You can use the HtmlFromRepoGenerator tool from the Custom Help Toolkit to get the files.

Because the Microsoft repos are public, you don't have to have a valid GitHub account to get the content. However, we recommend that, at a minimum, your organization have a system account that has access to GitHub.

For more information, see Custom Help Toolkit.

Get the content without a GitHub account

If you don't want to collaborate with Microsoft on the content, you can get the latest version of the content from

GitHub even if you don't have a GitHub account. For example, you can just clone the relevant GitHub repo. A GitHub account isn't required to clone a repo. Because the Microsoft repos are public, anyone can always access them.

Translate the content

You can use the Dynamics 365 Translation Service (DTS) to translate your own or the Microsoft-provided content into other languages. The service is hosted in Microsoft Dynamics Lifecycle Services (LCS), and currently supports translation of content in Word documents and HTML files. For more information, see Translate documentation files.

See also

Convert Dynamics AX custom Help for use in Dynamics 365
Docs contributor guide
Docs Authoring Pack for Visual Studio Code
Getting started with writing and formatting on GitHub
Visual Studio Code
Atom
DocFx

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Develop and customize home page

2/18/2021 • 3 minutes to read • Edit Online

This topic provides links to topics about development.

Overview

The Finance and Operations applications represent the next-generation enterprise resource planning (ERP) offering from Microsoft. The apps enable the entire ERP application suite as a cloud-based solution, for both public and private clouds, as well as on-premises. The apps leverage the speed, simplicity, and cost-effectiveness of working in the cloud, while building on the latest technology from Microsoft. The development experience includes:

- Development tools that are decoupled from any running environment. You develop against local, XML-based files, not the online database.
- Microsoft Visual Studio is the development environment. The Visual Studio environment is customized to provide you with a smooth and familiar experience.
- The X++ compiler generates Common Intermediate Language (CIL) for all features. CIL is the same intermediate language used by other .NET-based (managed) languages, such as the C# programming language.
- You can leverage the browser-based client and the design patterns for forms to provide an improved end-user experience.
- The Application Lifecycle Model (ALM) supports build automation, test automation, and deployment of models to the cloud.

Architecture

- Application stack and server architecture

Getting started

- Get evaluation copies
- Sign up for preview subscriptions
- Deploy and access development environments
- Development system requirements
- Removed or deprecated features
- Deprecated APIs
- Rename a local development (VHD) environment
- Introduction to Azure DevOps (Video)

Fleet Management

- Fleet Management sample application
- End-to-end scenario for the Fleet Management sample application

Development tools

Tutorials for development tools

- Development tools tutorial
- Create models and data model elements overview
- Build and debug projects
- Version control, metadata search, and navigation

Tools, models, and VMs

- Development tools in Visual Studio
- Application Explorer
- Finance and Operations project type in Visual Studio
- Element designers
- Commands for determining how elements are used
- Models and packages
- Build operations
- Code editor features
- Tools add-ins for Visual Studio
- Export and import models
- Metadata search in Visual Studio
- Create new users on development machines
- Update the Visual Studio development tools
- Development and build VMs that don't allow admin access FAQ

Build automation using Azure

- Build automation using Microsoft-hosted agents and Azure Pipelines
- Add license files to a deployable package in Azure Pipelines
- Create deployable packages in Azure Pipelines
- X++ model-versioning in Azure Pipelines
- Download assets by using Azure Pipelines
- Upload assets by using Azure Pipelines
- Deploy assets by using Azure Pipelines
- Create a Lifecycle Services (LCS) connection in Azure Pipelines
- Update a legacy pipeline in Azure Pipelines

X++ programming language

Overviews

- X++ and debugger features
- Write business logic by using C# and X++ source code

Language support

- Changes in X++ and the X++ compiler
- EventHandlerResult classes in request or response scenarios
- Debug X++ code by using the debugger in Visual Studio
- Language Integrated Query (LINQ) provider for C#
- Write best practice rules

Reference

- X++ language reference

Customize with extensions and overlayering

- Extensibility home page
- Customize App Suite reports by using extensions

Code migration

- How to resolve conflicts in Visual Studio (video)
- Code migration and upgrade home page

Move packages between environments

- Create deployable packages of models

Performance

- Take traces by using Trace parser
- Diagnose issues and analyze performance by using Trace parser
- Performance timer

User interface concepts

The client is an HTML web client that runs in all major browsers. For information about developing and customizing the user interface, see the User interface development home page.

Analytics

- Analytics, aggregate measurements, and KPI modeling

Reporting services

- Electronic reporting (ER) overview

Data entities and OData

- Data entities overview
- Open Data Protocol (OData)

Testing support in Visual Studio

- Testing and validations
- Test projects in Visual Studio
- Deploy and use a continuous build and test automation environment
- Task recorder resources

Office integration

- Office integration overview

Intelligence

- Business intelligence (BI) and reporting home page

Mobile platform

- Mobile platform resources

Global finance management

- Development for Dynamics 365 Finance home page

Licensing

- Independent software vendor (ISV) licensing

Supply Chain Management

- Gantt control development guide
- Create a new transportation management engine

Additional resources

Insider tips on development

Application stack and server architecture

2/18/2021 • 4 minutes to read • Edit Online

The application stack is divided into platform models and application-specific models. The platform models are Application Platform, Application Foundation, and Test Essentials. There are many application-specific models. Some examples are Application Suite, Ledger, Retail, and Case Management.

Overview

The application stack and server architecture align with three key pillars:

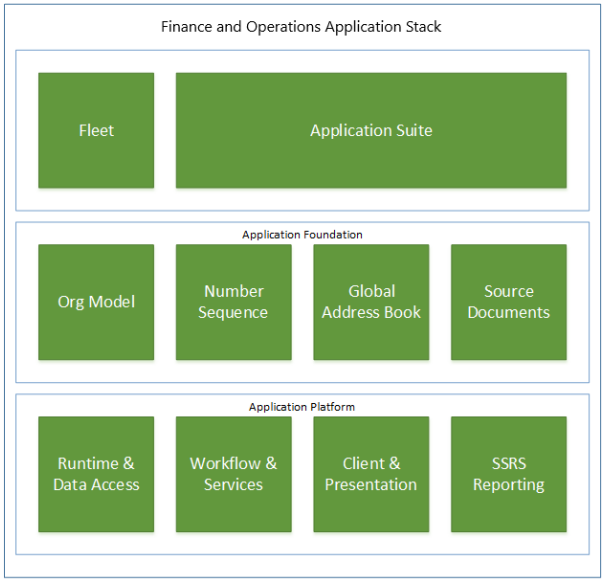
- New client
- Cloud readiness
- New development stack

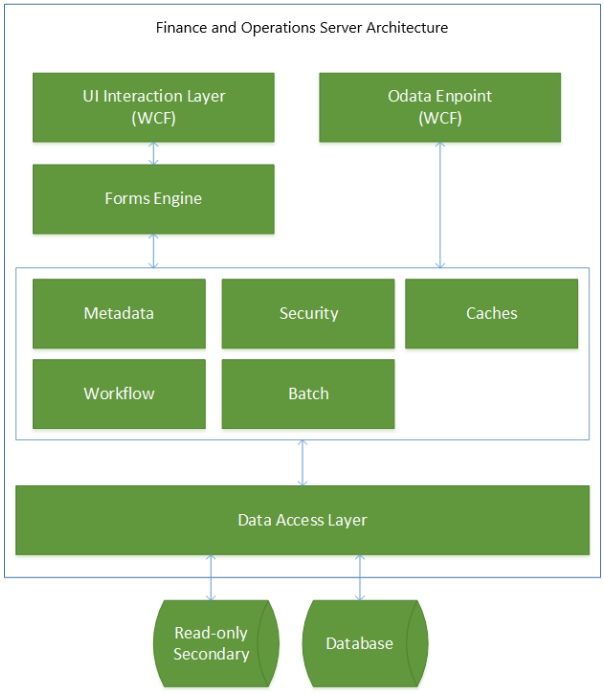
The application stack is divided into several models: Application Platform, Application Foundation, Test Essentials, and the application suites. The separation enables new application development on the base foundation models, just as the Fleet Management sample application has been developed. Note the following important points about the changes in the server architecture:

- The services endpoint on the server is now responsible for returning all form and control metadata and data to the browser-based client. There is no longer any remote procedure call (RPC)-based communication with the server. The form objects still run on the server, and rendering has been optimized for browsers and other clients through server and client-side (browser) investments.
- The server, including the application code base, is deployed to an Internet Information Services (IIS) web application. In the cloud, it's deployed to Microsoft Azure infrastructure as a service (IaaS) virtual machines (VMs).
- It is hosted on Azure and is available for access through the Internet. A user can use a combination of clients and credentials to access it. The recommended primary identity provider is OrgID, and the store for the identity is Azure Active Directory (Azure AD). The security subsystem uses the same AuthZ semantics for users and roles.
- Two types of clients must be considered for access in the cloud: active clients and passive clients.
    - Active clients can programmatically initiate actions based on responses from the server. An active client doesn't rely on HTTP redirects for authentication. A smart/rich client is an example of an active client.
    - Passive clients can't programmatically initiate actions based on responses from the server. A passive client relies on HTTP redirects for authentication. A web browser is an example of a passive client.

  Currently, Access Control Service (ACS) doesn't support a mechanism for non-interactive authentication. Therefore, even when active clients try to authenticate by using ACS, they must use passive client authentication, in which a browser dialog box prompts the user to enter their credentials.
- A completely revamped metadata subsystem incorporates the new compiler and Microsoft Visual Studio–based development model. The model store is represented as a set of folders and XML artifacts that are organized by model. The model elements, such as tables, forms, and classes, are represented by an XML file that contains both metadata and source code.

The left side of the following diagram shows how the application stack has been split into distinct models. The right side shows how the key components are stacked in the server.





The Finance and Operations applications use an entry point security model. A form allows read-only access if the menu item used for navigation to that form has only Read Permissions. However, navigation to that same form through another menu item that provides Create Permissions, Delete Permissions, or Update Permissions allows write operations on the form. This behavior simplifies the development experience, because developers can specify the behavior for a form through a given entry point.

Cloud architecture

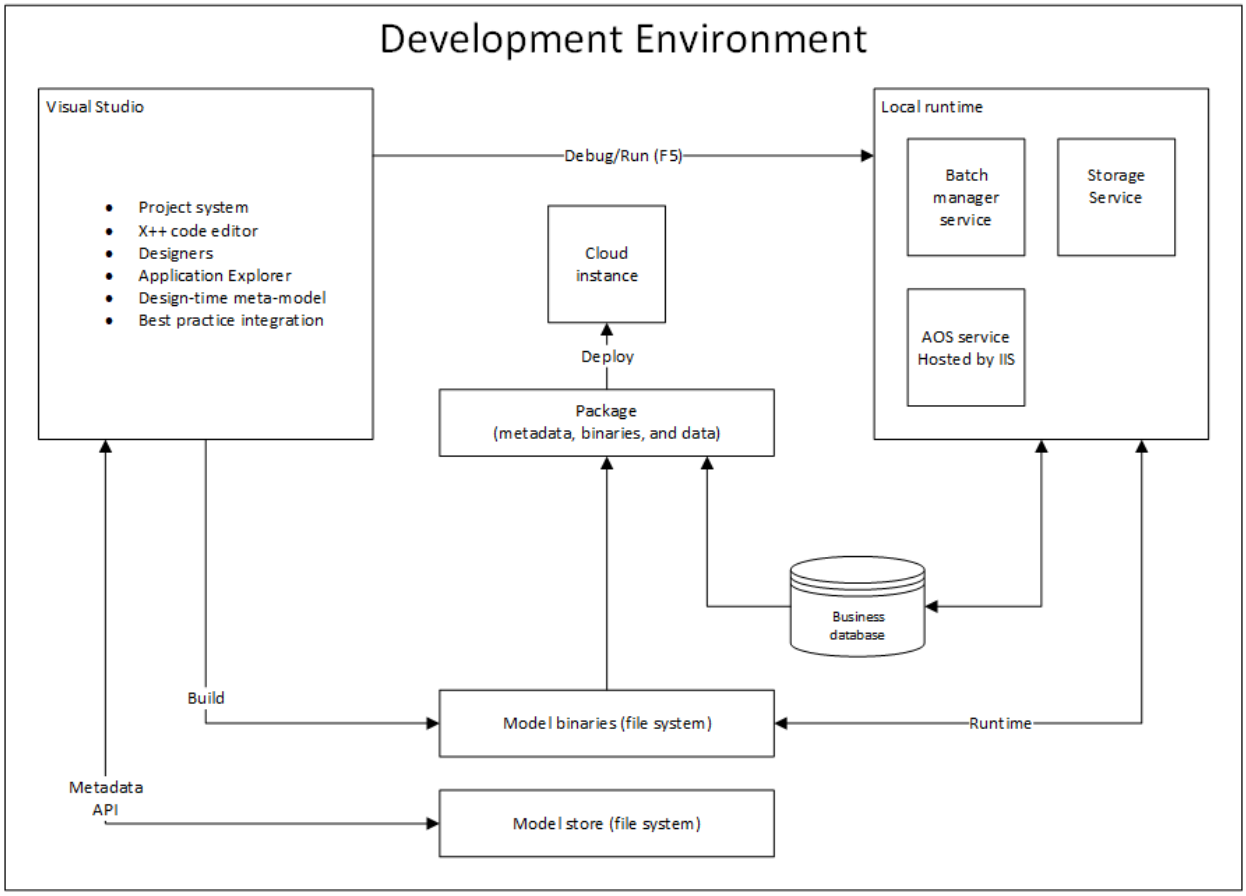
The cloud architecture includes services that automate software deployment and provisioning, operational monitoring and reporting, and seamless application lifecycle management. The cloud architecture consists of three main conceptual areas:

- **Lifecycle Services (LCS)** – LCS is a multi-tenant shared service that enables a wide range of lifecycle-related capabilities. Capabilities that are specific to this release include software development, customer provisioning, service level agreement (SLA) monitoring, and reporting capabilities.
- **Finance and Operations** – The VM instances are deployed through LCS to your Azure subscription. Various topologies are available: demo, development/test, and high-availability production topologies.
- **Shared Microsoft services** – A Finance and Operations application uses several Microsoft services to enable a "One Microsoft" solution where customers can manage a single sign-in, subscription management, and billing relationship with Microsoft across Finance and Operations applications, Microsoft 365, and other online services.

Many features of the Azure platform are used, such as Microsoft Azure Storage, networking, monitoring, and SQL Azure, to name a few. Shared services put into operation and orchestrate the application lifecycle of the environments for participants. Together, Azure functionality and LCS will offer a robust cloud service.

Development environment

The architecture of the development environment resembles the architecture of the cloud instance. It also includes the software development kit (SDK), which consists of the Visual Studio development tools and other components. Source control through Team Foundation Server or Visual Studio Online enables multiple-developer scenarios, where each developer uses a separate development environment. Deployment packages can be compiled and generated on a development environment and deployed to cloud instances by using LCS. The following diagram shows how the key components interact in a development environment.



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Get evaluation copies

2/18/2021 • 2 minutes to read • Edit Online

A public preview is available. You can sign up and deploy a cloud instance of the latest build. This public preview available through Microsoft Dynamics Lifecycle Services (LCS). These links provide more information about how to download and use the public preview:

- Sign up for preview subscriptions
- Service update availability
- Partner Trial
- How can I setup a solution trial instance in Azure with my customization and demo data?
- How do I login to the new AX as a demo user persona?

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Sign up for preview subscriptions

2/18/2021 • 7 minutes to read • Edit Online

This topic explains how to subscribe to the preview/partner offer and deploy an environment. The subscription that you create gives you a Microsoft Online Services test tenant and a Microsoft Dynamics Lifecycle Services (LCS) project where you can deploy an environment. This topic will also help you set up additional users in your Microsoft Online Services tenant and gain experience with service administration capabilities. Here are the skills that you will learn:

- Subscribe, and create a new Microsoft Online test tenant.
- Navigate to LCS projects.
- Use various features of LCS.
- Add users to Microsoft Azure Active Directory (Azure AD) and the client.
- View resources in your subscription email.

Key terms

- **Microsoft Online Services tenant** – A tenant is the group of all subscriptions and users for your organization. The tenant is created at the same time as your first subscription in Microsoft Online Services.
- **Subscription** – A subscription gives you an online cloud environment and experience. It also lets you see how customizations that you develop can be deployed to the cloud.
- **Microsoft Azure Active Directory** – The cloud environment includes Azure AD. Azure AD helps you manage users, groups, security roles, and licenses for online applications, much as you manage them for on-premise environments.
- **Users** – Users of the services that your organization has subscribed to are managed in Azure AD. Any users in your tenant can be added and assigned to security roles.
- **Developers and administrators** – Developers and administrators are users who also have access to LCS that lets them manage projects and environments. These users are also end users.
- **Organizational account** – Users receive Azure AD credentials. These credentials are separate from other desktop or corporate credentials. The Azure AD credentials are used to sign in to Microsoft 365 and other Microsoft cloud services. Users sign in by using their organizational account.

  > **IMPORTANT**
  > For this release, we ask that you not use any existing credentials that are associated with other online services, such as Microsoft 365 or Microsoft Dynamics CRM Online.

- **Microsoft account** – Microsoft accounts were formerly known as Passport accounts or Windows Live ID accounts. Currently, Microsoft accounts can't be used with Finance and Operations applications, Microsoft Dynamics 365 Commerce, or other Microsoft Online Services. However, Microsoft accounts are still required for Microsoft Connect and other Microsoft Business Solutions sites, such as CustomerSource, PartnerSource, Information Source, and Microsoft Dynamics Community. You will continue to use your Microsoft account to access these services.
- **Microsoft 365 admin center** – Microsoft 365 admin center is the subscription management portal

that Microsoft 365 provides for administrators. Microsoft 365 admin center is used to provide management functions for users and subscriptions.

- **Environments** – You can deploy as many single instances of a virtual machine (VM) as you require. We call these instances *environments*.

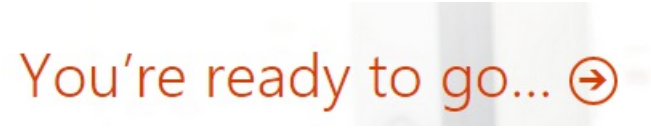
Prerequisites

1. You've received an email that invites you to participate in the preview.
2. If your company has an organizational account with Microsoft Online Services, and you're signed in, you must sign out before you continue. Alternatively, you can use **InPrivate Browsing** mode.
3. If you aren't sure whether you're signed in, delete your browser cookies, and then close your browser before you continue.

Subscribe

> **IMPORTANT**
> Only one person (tenant administrator) in an organization must perform this task. If you aren't the person who is subscribing to this release, wait until your organization has been signed up and you've received your user credentials. Then continue with the procedure.

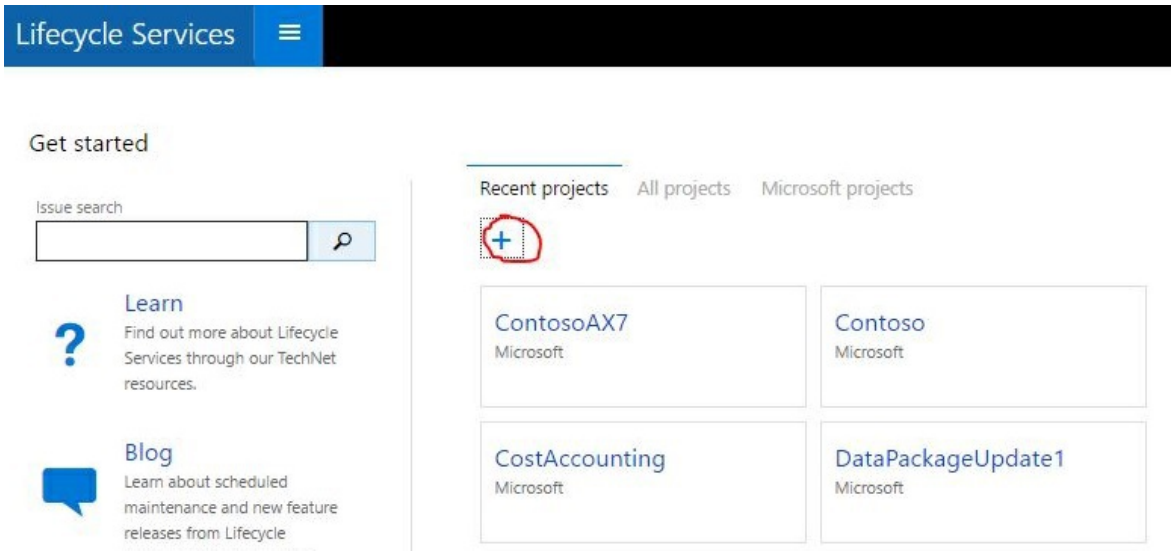
1. Finance and Operations applications and Retail are available only to existing Microsoft Dynamics 365 channel partners and customers who are currently enrolled in the Business Ready Enhancement Plan (BREP) service plan. To subscribe, visit PartnerSource Business Center.
2. On the **Account setup** page, in the **Country or region** field, select the country or region.
3. Follow the wizard and prompts to complete the sign-up, until you reach the last step.



Start a new project in LCS

To use LCS to manage your environments, you must create a new project.

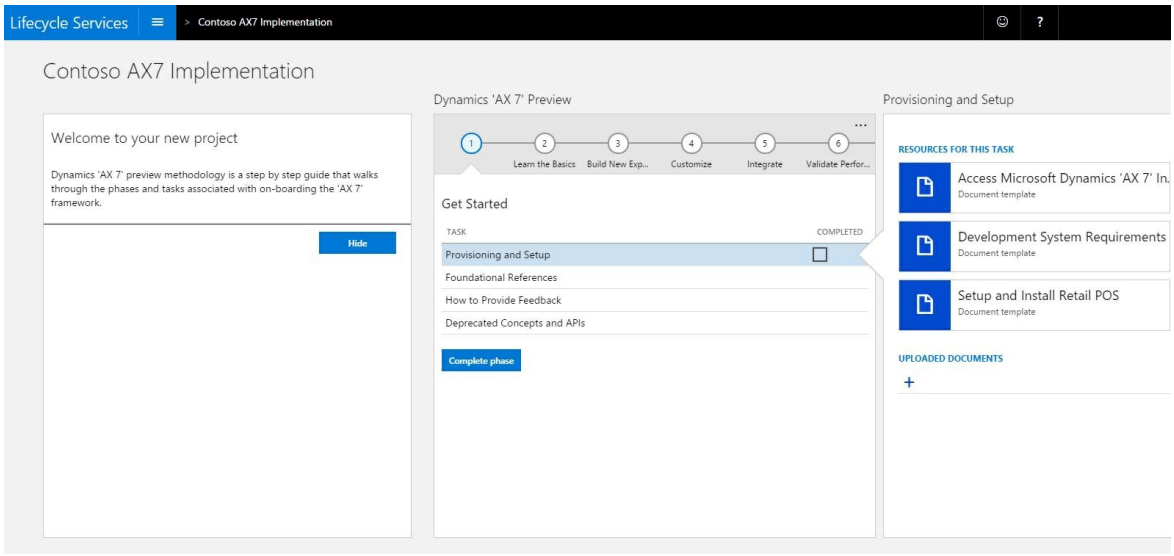
1. Go to https://lcs.dynamics.com/Logon/Index.
2. Select **Sign in**.
3. Sign in by using the account that you used to subscribe.
4. Select the plus sign (**+**) to create a new project.



5. Select the project type.
6. Enter the project information, and then select **Create**.

   If you plan to evaluate Commerce, be sure to select **Microsoft Dynamics 365 Commerce** in the **Product name** field.

   The new project for managing your instance is created.



Add users to LCS

You're already set up as a user of your LCS project. If you've also added other Microsoft 365 users, you must add them to this project. Other administrators and developers will then be able to deploy their own environments. These LCS users are team members who will actively work on the implementation. Don't confuse them with end users. Start on the project page in LCS.

1. Scroll to the right, and then, in the **More tools** section, select the **Project users** tile.
2. In the upper left, select the plus sign (**+**) to add a new user.
3. In the **Email** field, enter the email address of the user that you're adding. This email address should be the Microsoft 365 organization email address that you created earlier.
4. In the **Project role** field, select **Project Owner**.
5. Select **Invite**.
6. Repeat steps 2 through 5 for all users in your organization.

Deploy environments

Environments should be deployed to an existing Azure subscription.

> **NOTE**
> Each developer of an environment must deploy their own system to Azure. However, only the first project user must set up the Azure subscription for deployment.

You can create environments in two ways:

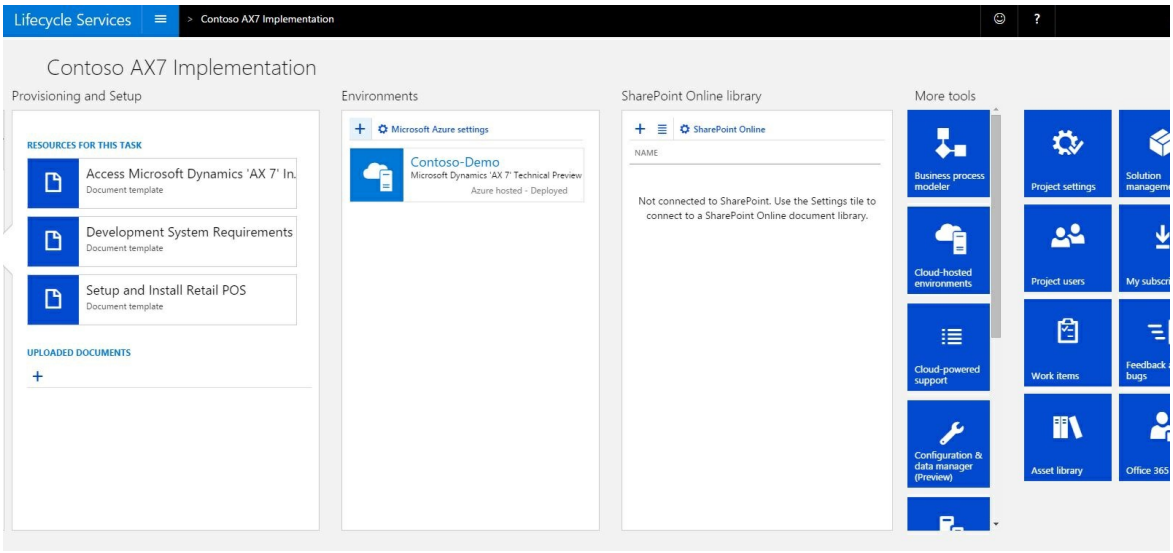
- Deploy to Microsoft cloud services (Azure).
- Download a local virtual hard disk (VHD).

Start on the project page in LCS.

1. In the **Environments** section, select the plus sign (**+**). The **Microsoft Azure setup** dialog box appears.
2. Enter your Azure subscription ID. You can find your Azure subscription ID in Azure Portal (https://ms.portal.azure.com/), under **Settings** in the lower left.
3. Select **Next**.
4. Download the Azure Management Certificate to a local folder on your computer, and then upload it to Azure Management Portal by going to **Settings** > **Management Certificates**. This certificate will enable LCS to communicate with Azure on your behalf.
5. Return to LCS, and select **Next**.
6. Select the Azure region to deploy in. The **West US** region will have the fastest deployments, but it's important that you select a data center that is close to where you plan to use this system.
7. Select **Connect**.
8. In the list of available topologies, select the topology to deploy. You can select either the **Download** link to download the VHD or **Next** to deploy on Azure. Azure is the preferred path.
9. Enter the name of the environment.
10. Read the pricing and licensing terms, and then select the check box to indicate that you understand them.
11. Select **Next**.
12. Confirm the details, and then select **Deploy**.

    > **NOTE**
    > Developers and administrators who will use their own environments must sign in and repeat these steps.

    After you deploy your environment, it's available in the **Environments** section.



13. Select the environment to view details about the deployment status. The first deployment will require a few hours, but each subsequent deployment will be much faster.

14. When the deployment status changes to **Deployed**, select **Login** to connect to the client, or select the name of the VM to connect to the development machine by using Remote Desktop. After the deployment is completed, you can find the base URL and the information that you require to connect to the environment via Remote Desktop.

Use the features of LCS

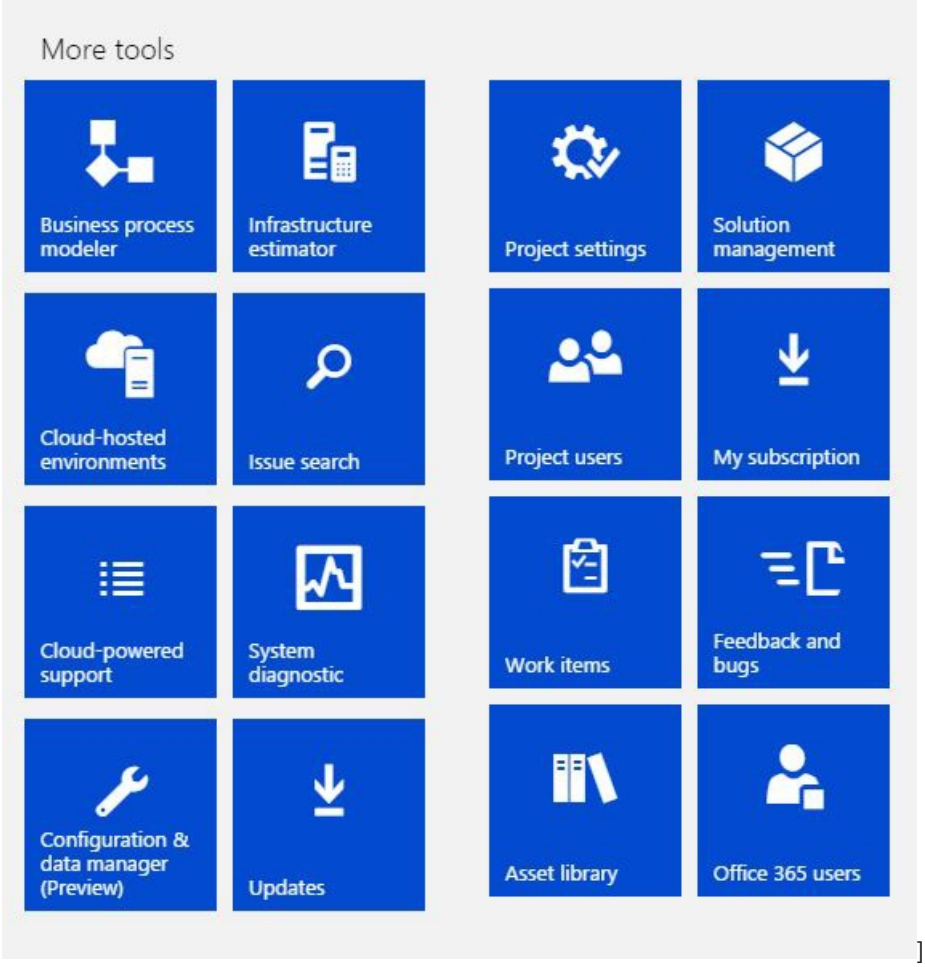
LCS is the starting point for performing online administrative activities. Here are some of these activities:

- Deploy VMs on Azure.
- Access materials.
- Access downloads of tools and resources.

Explore the LCS project

1. Review the methodology, and complete the tasks and phases as you progress through the life cycle. The phases and task information lets you view tools and resources that are available throughout your enterprise resource planning (ERP) experience.

2. Scroll to the right, and review the tiles.



]

The available tiles include various tools and services in LCS. They also include the following additional tiles:

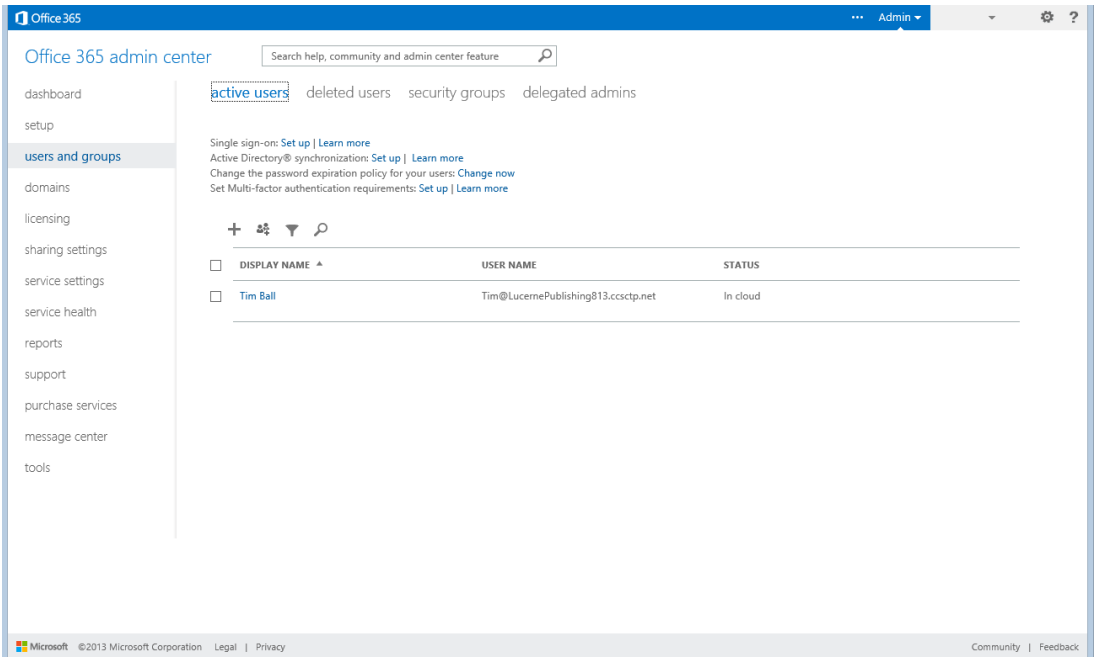
- **My subscription** – The Microsoft 365 subscription management portal is where you can view and work with your online subscriptions. By selecting **User and Groups** in the left navigation section of the page, you can also manage your online users.

  > **NOTE**
  > To access the page, you must be a member of the **Global Administrator** role for your organization's Microsoft Online Services tenant.

- **Feedback and bugs** – This tile opens the **General Feedback** page in Microsoft Connect. Use this page to record bugs, and to design change requests, feature requests, and suggestions.

- **Microsoft 365 users** – This tile opens the **Users and groups** page in Microsoft 365 admin center. You can add, update, and remove users, reset passwords, and assign licenses for other services.

  > **NOTE**
  > To access the page, you must be a member of the **Global Administrator** role for your organization's Microsoft Online Services tenant. The installing user is always a global administrator, but other users must be added to this role.



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Deploy and access development environments

2/18/2021 • 10 minutes to read • Edit Online

This topic describes how to access development instances, configure local development virtual machines (VMs), and find important configurations settings for developers and administrators.

Definitions

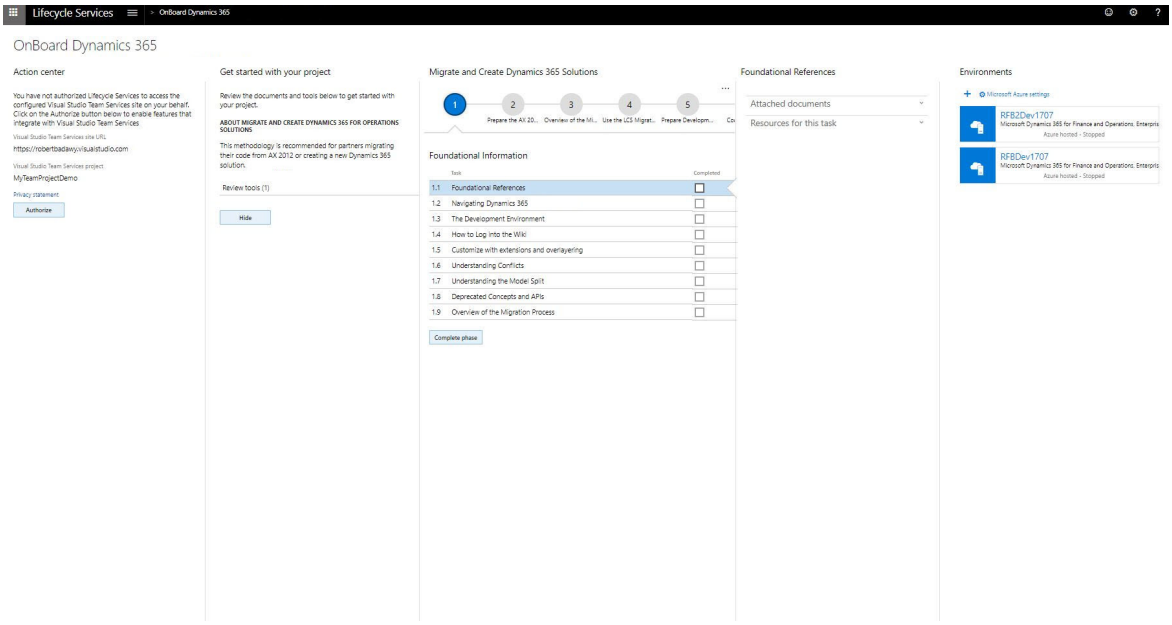
TERM	DEFINITION
End user	A user who accesses an instance through the web client. The end user must have Microsoft Azure Active Directory (Azure AD) credentials to access an instance and must be provisioned/added as a user of that instance.
Developer	A user who will develop code through the Microsoft Visual Studio environment. A developer requires Remote Desktop access to development environment (VM). The developer account must be an administrator on the VM.

Deploying cloud development environments

The process of deploying cloud hosted environments differs for Lifecycle Services (LCS) trial or partner projects and LCS customer implementation projects.

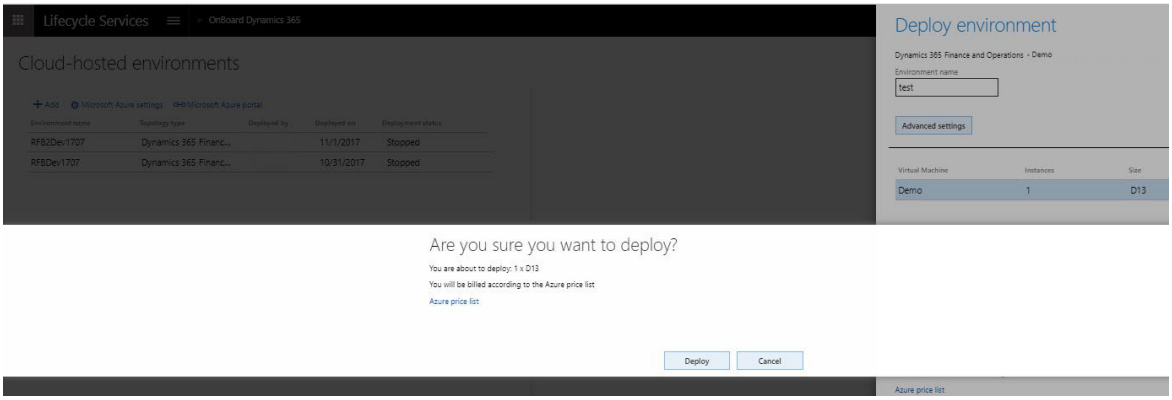
For a **Trial** or **Partner** project:

1. Create a connection between an LCS project and your Azure subscription. You will need your Azure subscription ID and authorize the use of the subscription.
2. Select **+** under **Environments** to deploy.



3. Select an application and platform version.
4. Select an environment topology. For more information, see Sign up for preview subscriptions.

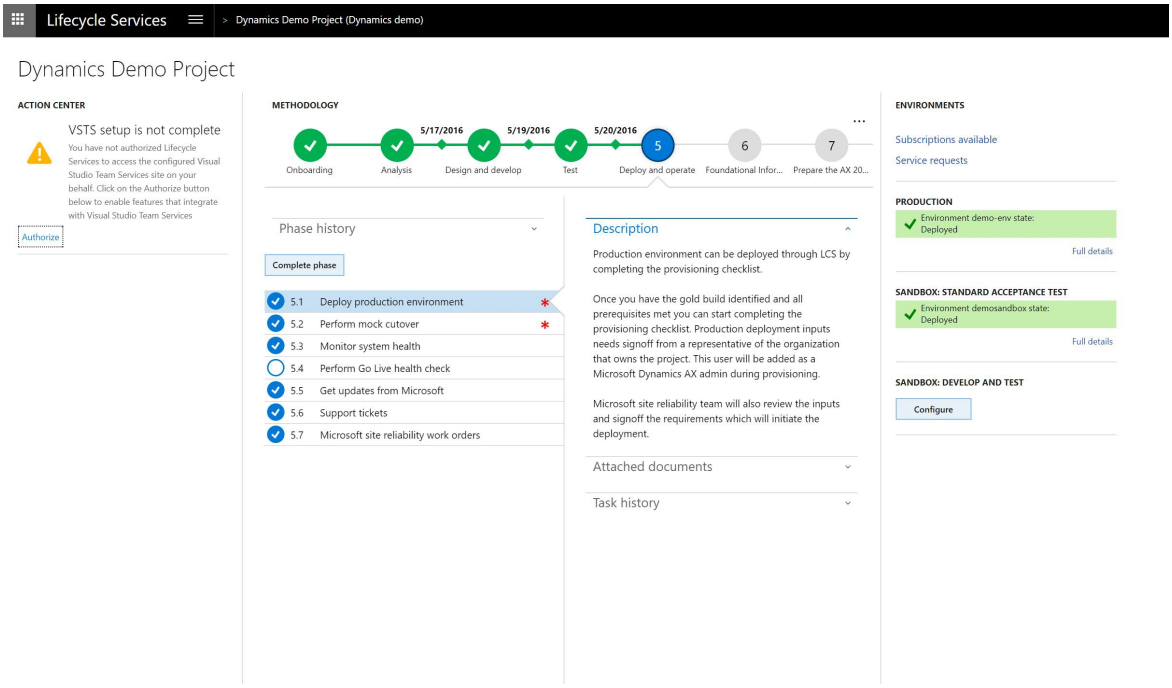
5. If you chose a cloud-hosted environment, select which Azure connector you want to use. Then select **Deploy**.



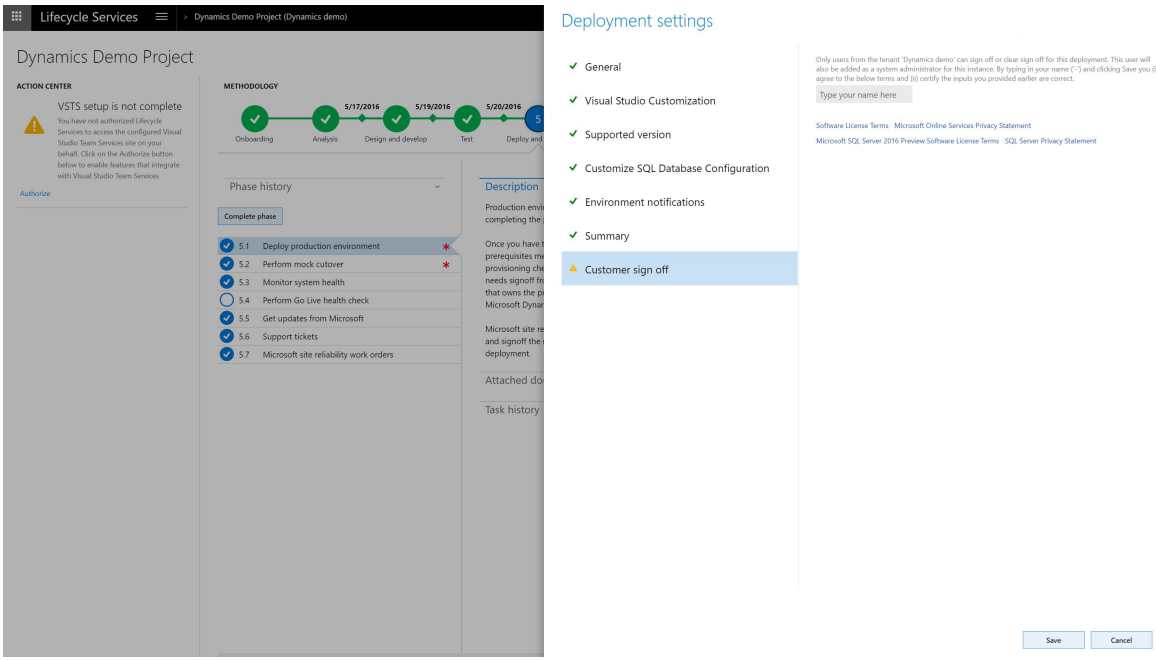
6. If you did not choose a cloud-hosted environment, select a VHD to download.

For a **Customer Implementation** project:

1. Sign in to your LCS Implementation project.
2. Select **Configure** to deploy.

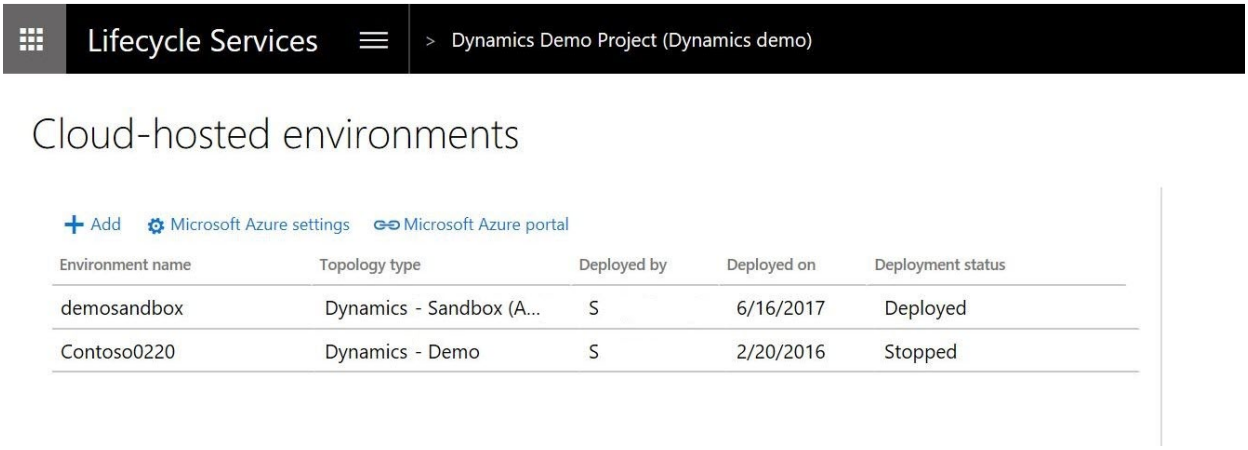


3. Select an application and platform version.
4. Specify the settings and select **Save**.



Customers are provided with one free "develop and test" environment hosted in Microsoft's Azure subscription. Under "Develop and test", there are two types of environments, **Develop** and **Build and Test**. For development and customization activities, configure a **Develop** environment. **Build and Test** environments are not supported for standard development activities. Instead, they are used for daily build and test automation. For more information, see Deploy and use an environment that supports continuous build and test automation.

Additional develop and build environments can either be purchased or hosted in your own Azure subscription. To deploy an environment in your own subscription, go to the **Cloud-hosted environment** page.



Cloud environment that is provisioned through LCS

When a cloud environment is provisioned through LCS:

- The user who requests the cloud environment is provisioned as the administrator in that environment.
- User accounts are provisioned on the development VM to allow access to the environment using Remote Desktop, these credentials are accessible on the environment page in LCS.

Accessing an instance through a URL

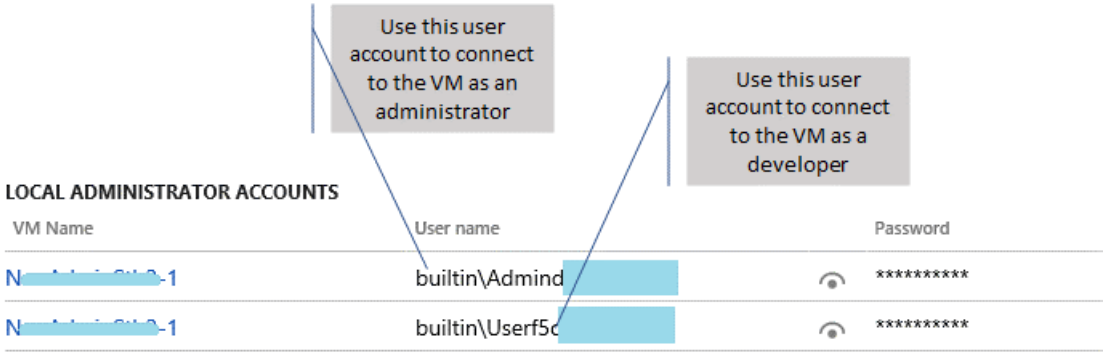
The system can be accessed by end users. The administrator can add users to this system by using the **Users** page in the instance. Note that these additional users don't have to be users in LCS. You obtain the base URL for the cloud environment from your LCS project site.

1. Go to your LCS project page.
2. In the **Environments** section, click the deployed environment.

3. When the environment page opens, you can access the application by clicking **Login** > **Log on to Finance and Operations** in the upper-right corner.
4. Use valid end user credentials to sign in to the application. If the current LCS user is the user who originally deployed the environment, that user is probably a valid end user and the administrator of the application.
5. In your browser, make a note of the base URL after you sign in. For example, the base URL might be `https://dynamicsAx7aosContoso.cloud.dynamics.com`.

Accessing the cloud instance through Remote Desktop

Cloud environments can be accessed both as an end user and as a developer. The developer gets access to the system through Remote Desktop credentials. The Remote Desktop credentials are obtained from the environment page on the LCS project site (see the illustration earlier in this topic).



For environments deployed **before Platform update 12**:

1. Click the VM name.
2. Use the local administrator user name and password that are shown to connect to the cloud VM through Remote Desktop. You can reveal the password by selecting the show password icon.

For any environments deployed **on or after Platform update 12** , there are distinct accounts, a developer account and an admin account. Customers will not have access to virtual machine admin accounts on development or build environments that are running in Microsoft subscriptions. Thus, the admin account will be hidden unless the environment is running in their Azure subscription. For more information, see Development and build VMs that don't allow admin access FAQ.

After you sign in to the environment through Remote Desktop, if you want to access the local application from the browser, use the same base URL that you use to access the application from a remote computer. The previous section explains how to obtain this base URL from LCS.

VM that is running locally

A virtual hard disk (VHD) is made available for download from LCS, so that you can set it up on a local machine. This system is intended to be accessed by a developer and is a pre-configured one-box development environment of Finance and Operations apps. The VHD is available in the Shared Asset library of LCS under the asset type **Downloadable VHD**.

1. Go to the LCS main page and select **Shared asset library** or go to Shared Asset Library.
2. Select the asset type **Downloadable VHD**.
3. Find the VHD you are looking for based on the desired Finance and Operation version. The VHD is divided into multiple file parts that you need to download. For example, the asset files that start with "VHD - 10.0.5" are the different files you need in order to install version 10.0.5.
4. Download all files (parts) associated with the desired VHD to a local folder.
5. After the download is complete, run the executable file that you downloaded, accept the software license agreement, and choose a file path to extract the VHD to.

6. This creates a local VHD file that you can use to run a local virtual machine.

Commerce configuration

Follow the steps in this section if you are also configuring for Commerce.

To use the downloadable VHD for POS customizations, you must also follow this step.

- On the host computer, enable Nested VM support. For more information, see Run Hyper-V in a Virtual Machine with Nested Virtualization.

Running the virtual machine locally

Follow these steps to run the VM from Hyper-V Manager.

1. To start the VM, select **Start**.
2. To open the VM in a window, select **Connect**.
3. Select the **Ctrl+Alt+Delete** button on the toolbar. The VM receives most keyboard commands, but Ctrl+Alt+Delete isn't one of them. Therefore, you must use the button or a menu command.
4. Sign in to the VM by using the following credentials:
   - User name: **Administrator**
   - Password: **pass@word1**

   > **TIP**
   > You can resize the VM window by changing the screen resolution. Right-click the desktop on the VM, and then click **Screen resolution**. Select a resolution that works well for your display.

5. Provision the administrator user. For more information, see the next section.
6. Start the Batch Manager Service. This step is required if you're running batch jobs or workflows.
   a. Open a **Command Prompt** window as an administrator.
   b. Type **net start DynamicsAxBatch**, and then press Enter.

   You can also start the service from the **Services** window.
7. Apply updates as needed.

Commerce configuration

For POS customizations, you must also follow these steps on the guest VM.

1. Download and install Microsoft Emulator for Windows 10 Mobile Anniversary Update.
2. Start the Hyper-V host service. For more information, see Hyper-V: The Hyper-V Virtual Machine Management service must be running. If errors occur during startup, you can also try to uninstall and reinstall the Hyper-V role on the guest VM.

Provisioning the administrator user

For developer access, you must be an administrator on the instance. To provision your own credentials as an administrator, run the admin user provisioning tool that is provided on the desktop, and provide your email address (Azure AD credentials) in the tool.

1. From the desktop, run the admin user provisioning tool as an administrator (right-click the icon, and then click **Run as administrator**).
2. Enter your email address, and then select **Submit**.

Commerce configuration

Follow the steps in this section if you are also configuring for Commerce.

For Dynamics 365 for Operations, Version 1611

1. Run the RetailTenantUpdateTool.

   - The icon for this tool is available on the desktop.
   - This tool is also available at the following location: C:\windows\System32\WindowsPowerShell\v1.0\PowerShell.exe -File C:\RetailSDK\Tools\RetailTenantUpdateTool.ps1

2. Double-click the icon to start this tool. You will be prompted for your Azure AD credentials. You must use the same credentials that you used in the admin user provisioning tool earlier.

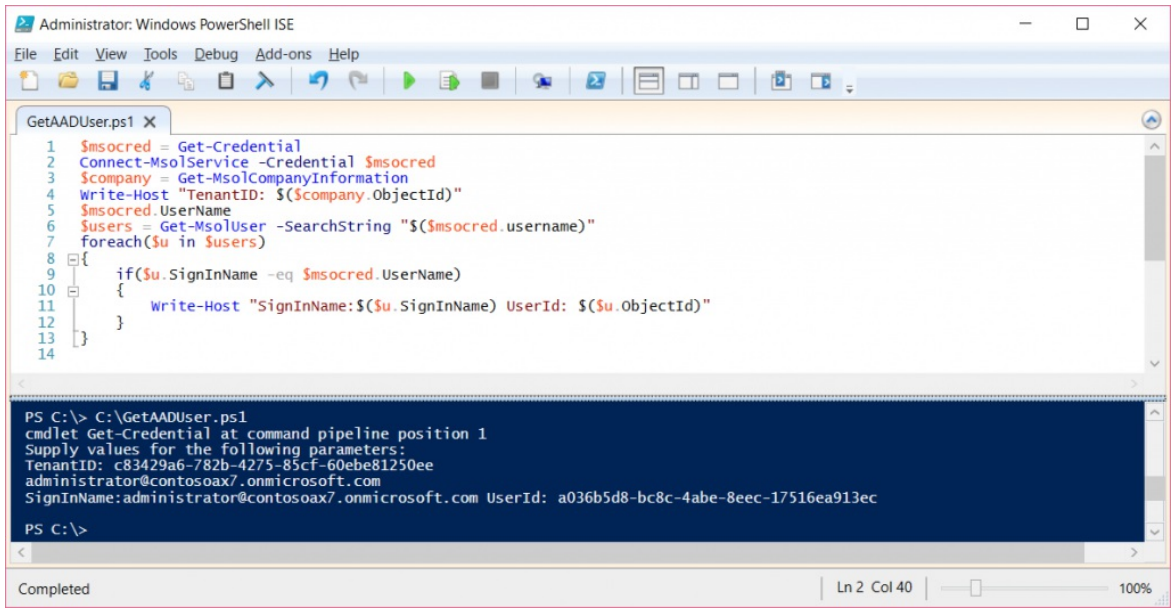
For Dynamics 365 for Operations 7.0

1. Install Microsoft Online Services Sign-In Assistant for IT Professionals RTW.

2. Install Azure Active Directory Module for Windows PowerShell (64-bit version).

3. Query Azure AD for your tenant and user ID. Open a Windows PowerShell Integrated Scripting Environment (ISE) window with administrative privileges, and run the following command. You will be prompted for Azure AD credentials. Use the same user account that you used in the admin user provisioning tool earlier.

```
$msocred = Get-Credential
Connect-MsolService -Credential $msocred
$company = Get-MsolCompanyInformation
Write-Host "TenantID: $($company.ObjectId)"
$msocred.UserName
$users = Get-MsolUser -SearchString "$($msocred.username)"
foreach($u in $users)
{
    if($u.SignInName -eq $msocred.UserName)
    {
        Write-Host "SignInName:$($u.SignInName) UserId: $($u.ObjectId)"
    }
}
```



4. Update the following SQL script, and run it in on AXDB for that environment. Supply values for the following parameters from the preceding Windows PowerShell script output:

   - **TenantID** – For example, c83429a6-782b-4275-85cf-60ebe81250ee
   - **UserId** – For example, a036b5d8-bc8c-4abe-8eec-17516ea913ec

```
DECLARE @TenantId NVARCHAR(1024)         DECLARE @UserId NVARCHAR(1024)
SET @TenantId = ‘‘
SET @UserId = ‘‘
IF(LEN(@TenantId) > 0 AND LEN(@UserId) > 0)
    BEGIN
    UPDATE AxDBRAIN.dbo.SYSSERVICECONFIGURATIONSETTING SET [VALUE] = @TenantId WHERE [NAME] =
‘TENANTID’
    UPDATE RetailHoustonStore.ax.SYSSERVICECONFIGURATIONSETTING SET [VALUE] = @TenantId WHERE [NAME]
= ‘TENANTID’
    UPDATE AxDBRAIN.dbo.RETAILSTAFFTABLE SET EXTERNALIDENTITYID = @TenantId, EXTERNALIDENTITYSUBID =
@UserId WHERE STAFFID = ‘000160’
    END
ELSE
    BEGIN
    RAISERROR (15600, -1, -1, ‘TenantId and UserId must be set before running this script’)
    END
```

5. Reset Internet Information Services (IIS) by running **IISRESET** in an elevated **Command Prompt** window.

6. Update the Real-time service profile to use the new admin user.

    a. Go to **Retail and Commerce** > **Headquarters setup** > **Commerce scheduler** > **Real-time service profiles**.
    b. Edit the JBB record so that it uses the user that you used earlier (for example, `administrator@contosoax7.onmicrosoft.com`).
    c. Run CDX Job 1070 (Staff) for the default channel database.
    d. Verify that the job succeeded by viewing the **Download Sessions** page on the client.

Base URL of the local application

After the user is provisioned as an administrator, that user can access the instance on the computer by navigating to the following base URL: `https://usnconeboxax1aos.cloud.onebox.dynamics.com`. If you're using version control and plan to connect multiple development VMs to the same Azure DevOps project, rename your local VM. For instructions, see Rename a local development (VHD) environment.

Commerce configuration

The URL of the POS app is `https://usnconeboxax1pos.cloud.onebox.dynamics.com/`.

The URL of the Cloud POS app is `https://usnconeboxax1pos.cloud.onebox.dynamics.com`. After you complete the configuration steps, this VM is provisioned with your Azure AD tenant. Your Azure AD admin account is mapped to a cashier worker account in demo data. You can use this cashier account to easily activate a POS device in this environment.

- Cashier user ID: **000160**
- Cashier password: **123**
- Cashier LE: **USRT**
- Cashier store: **Houston**

Location of packages, source code, and other AOS configurations

On a VM, you can find most of the application configuration by opening the web.config file of AOSWebApplication.

1. Start IIS.
2. Go to **Sites** > **AOSWebApplication**.
3. Right-click, and then click **Explore** to open File Explorer.
4. Open the web.config file in Notepad or another text editor. The following keys are of interest to many

developers and administrators:

- **Aos.MetadataDirectory** – This key points to the location of the packages folder that contains platform and application binaries, and also source code. (Source code is available only in development environments.) Typical values are: c:\packages, c:\AosServicePackagesLocalDirectory, and J:AosServicePackagesLocalDirectory.
- **DataAccess.Database** – This key holds the name of the database.
- **Aos.AppRoot** – This key points to the root folder of the Application Object Server (AOS) web application.

Commerce configuration

The software development kit (SDK) is available at C:\RetailSDK. For more information about how to use and customize applications, see the following topics:

- Retail software development kit (SDK) architecture
- Point of sale (POS) device activation

Redeploying or restarting the runtime on the VM

To restart the local runtime and redeploy all the packages, follow these steps.

1. Open File Explorer, and go to C:\CustomerServiceUnit.
2. Right-click **AOSDeploy.cmd**, and then click **Run as administrator**.

This process might take a while. The process is completed when the cmd.exe window closes. If you just want to restart AOS (without redeploying the runtime), run **iisreset** from an administrator **Command Prompt** window, or restart AOSWebApplication from IIS.

Configure one-box development environments

2/18/2021 • 2 minutes to read • Edit Online

This article describes recommended configurations of your one-box developer environment.

Setup

1. Start Visual Studio, and on the toolbar, click **Dynamics 365 **> **Options**.
2. Expand the **Microsoft Dynamics** node, and then click **Projects**.
3. Verify that the **Organize projects by element type** check box is selected, and click ***OK.***
4. To view the line numbers in your code editor, select **Tools** > **Options** > **Text Editor** > **All Languages**.
5. Select the **Line numbers** check box.

Debugging

For better performance of the X++ debugger, you might want to turn off IntelliTrace. IntelliTrace collects the complete execution history of an application. It is not supported for X++ debugging and causes performance issues in the IDE when debugging large packages like Application Suite. To turn off Intellitrace, click **Options** > **IntelliTrace** > **Enable IntelliTrace**, clear the check box, and then click **OK**. Note that Intellitrace is only available in the Enterprise version of Visual Studio.

Create new users on development machines

2/18/2021 • 2 minutes to read • Edit Online

This article explains how to enable another user account as a developer on a development VM.

> **NOTE**
> This topic only applies to downloaded local virtual machines (VMs) or VMs that are hosted in a customer's subscription. You can't create new users on Microsoft-managed developer and build machines where developers have no administrator acces to the VM (Platform update 12 or newer).

When an environment is first deployed, only one user account is enabled as a developer on the virtual machine (VM). This user is preconfigured by Microsoft Dynamics Lifecycle Services (LCS) or is the local administrator account on downloaded virtual hard disks (VHDs). However, you can enable a new user account to develop on the VM. Even after you enable a new account, only one developer can develop at a time on the same VM/application.

Prerequisites

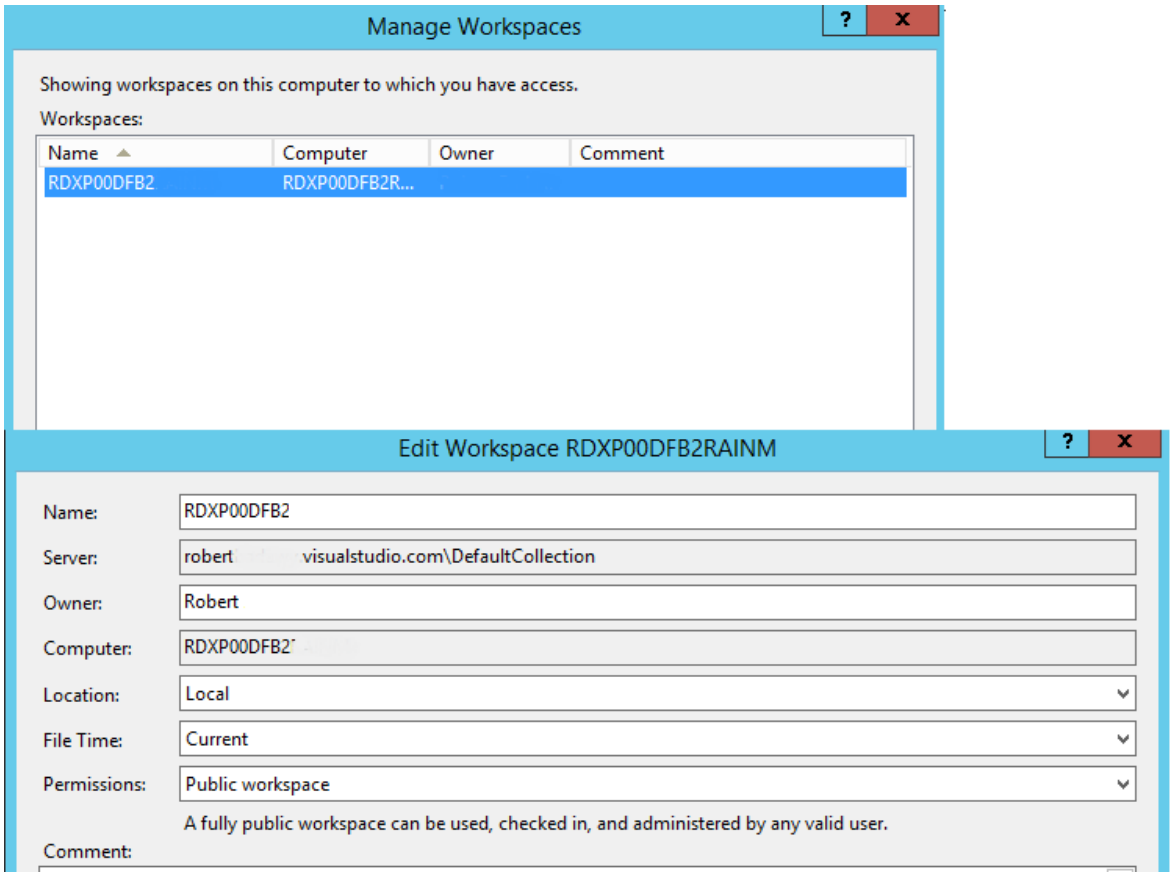
To enable a new user account to develop on the VM, the user account must be an administrator on the VM. Additionally, you must log on to the VM by using the credentials of the default developer account. If the VM is a Microsoft Azure VM, the account information is available on the environment page in LCS. If the VM is a local VM that runs on the downloaded VHD, use the local administrator account. For more information, see Deploy and access development environments.

Steps

1. Download the following script: ProvisionAxDeveloper.ps1, the script is available at https://github.com/Microsoft/Dynamics-AX-Scripts.
2. Open a Microsoft Windows **PowerShell Command Prompt** window as an administrator.
3. Run the ProvisionAxDeveloper.ps1 script. Specify the following parameters:
   - **DatabaseServerName** – Typically, this is the machine name.
   - **Users** – Use the following format: <domain or machine name>\user1, … <domain or machine name>\user n

   Examples

   - `ProvisionAxDeveloper.ps1 RDXP00DB20RAINM RDXP00DB20RAINM\username1`
   - ```
     ProvisionAxDeveloper.ps1 -databaseservername RDXP00DB20RAINM -users
     RDXP00DB20RAINM\username1,RDXP00DB20RAINM\username2
     ```
4. If more than one user account will be developing on the same version control workspace, you need to make the workspace public.
   a. In Visual Studio, open **Source Control Explorer**, select the workspace drop-down and select **Manage workspaces**.
   b. Select the application workspace, click **Edit,** then click **Advanced** and set the workspace to **Public workspace**.
     ```



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Development and build VMs that don't allow admin access FAQ

2/18/2021 • 6 minutes to read • Edit Online

This topic provides answers to frequently asked questions (FAQ) about virtual machines (VMs) that don't allow administrator access.

How can I install a deployable package?

Whenever possible, use Microsoft Dynamics Lifecyle Services (LCS) to install a deployable package. You can install a deployable package by using the **-devinstall** option. Remember that this option requires manual database synchronization.

For more information about how to install a deployable package, see Install deployable packages from the command line.

Is the Finance and Operations website accessible when Visual Studio isn't running?

Yes, you can access the Finance and Operations website when Microsoft Visual Studio isn't running. Microsoft Internet Information Services (IIS) Express is an .exe file that runs as the user. However, when you close Visual Studio, the XPPC agent starts regular IIS (not IIS Express) before it closes. This behavior helps to ensure that you can remotely access the Application Object Server (AOS) instance and the website, even when you sign out or the machine is restarted. We recognize that many people use these developer machines as test machines, and that they expect the AOS instance always to be running. However, IIS Express doesn't support this behavior.

What about the other services?

You can restart Microsoft Windows services such as Microsoft SQL Server, SQL Server Reporting Services (SSRS), SQL Server Integration Services (SSIS), SQL Server Analysis Services (SSAS), Batch, Financial reporting (formerly Management Reporter), and IIS. (For IIS, you must restart the World Wide Web Publishing Service because you can't use iisreset.exe.)

Can I clean up the service volume drive?

Yes, you have full access to the service volume drive. Therefore, you can clean up the monitoring data, and so on.

What are the alternatives to VMs that don't allow administrator access?

Both a Microsoft Azure environment on a private Azure subscription and a local virtual hard disk (VHD) allow administrator access. However, you must run Visual Studio as an administrator. This requirement applies because the administrator has access to these alternatives only through the **administrator** group, not explicitly.

Can I run Visual Studio as an administrator?

You are not required to run Visual Studio as an administrator. You cannot use the Remote Desktop Protocol (RDP) to connect as an administrator to VMs that are under a Microsoft-owned Azure subscription. These VMs include the Tier 1 VM that is included in the subscription and Tier 1 add-on VMs. However, if you're connecting

as an administrator to a VM that isn't under a Microsoft-owned subscription, you must still run Visual Studio as an administrator.

A "get latest" operation in Visual Studio failed because files are blocked by the AOS instance. How do I start and stop IIS?

You must use IIS Express. See the next question for more information.

What are the instructions for using IIS Express?

When IIS Express is started, an icon appears in the notification area (near the clock). When you right-click on the IIS Express icon, all the running sites are listed. You can stop IIS Express from that menu. Some actions in Visual Studio cause IIS Express to be started, but you can also explicitly start IIS Express from Visual Studio by selecting **Restart IIS Express** on the **Dynamics 365** menu.

To ensure that debugging functions properly with IIS Express and Finance and Operations Visual Studio projects, we recommend the following Internet Options settings:

- Go to **Control Panel** > **Internet Options** > **Security** tab > **Internet**, and clear the **Enable Protected Mode** check box.
- Go to **Control Panel** > **Internet Options** > **Security** tab > **Restricted sites**, and clear the **Enable Protected Mode** check box.

Can I install additional development tools (such as Fiddler and Pepper)?

No, you can't install additional development tools.

Is there a way to run Windows PowerShell and command prompt commands as an administrator?

No, you can't run Windows PowerShell commands and commands at a prompt command as an administrator.

Is the Trace Parser supported?

Trace Parser currently requires the user to be an administrator. It is not supported on dev/test environments that are managed by Microsoft that do not allow administrator access.

Is the Admin user provisioning tool supported?

The **Admin user provisioning** tool currently requires the user to be an administrator. The **Admin user provisioning** tool is typically used to change the tenant of the environment, but that should not be necessary. You can update the sign in information in the database for the Admin user or any other user. You only need the SID and network alias (email address) from a user that can access the environment or another environment on the same tenant. In many cases, the SID and network alias can be found in the database that came with the environment originally. Run the following commands to get the good SID and network alias from the source environment and update them in the target environment, respectively.

```
-- get value from source env.
select ID, SID, NETWORKALIAS from USERINFO where ID = 'Admin'

-- update value in target env.
update USERINFO set SID = 'new_SID', NETWORKALIAS = 'new_NetworkAlias' where ID = 'Admin'
```

Can the system be put into maintenance mode?

You can put the system into maintenance mode to change the license configuration. However, the procedure that is described in Maintenance mode isn't supported. Self-service support for maintenance mode in all environments will be added to LCS in the future. Until this support is available in LCS, you can follow these steps to put a system into maintenance mode.

1. Establish an RDP connection to the developer machine.
2. On the developer machine, sign in to SQL Server by using the credentials for the axdbadmin user from LCS. Then switch to the AXDB database, and run the following command.

   ```
   update SQLSYSTEMVARIABLES SET VALUE = 1 where PARM = 'CONFIGURATIONMODE'
   ```

3. Restart the **World Wide Web Publishing Service** to reset IIS.

   After the service is restarted, the system will be in maintenance mode.

4. When you've completed your maintenance mode activities, repeat steps 2 and 3, but set the value to **0** in step 2.

Can I install a license deployable package?

Development environments

Use LCS to install a license deployable package on any cloud development environment.

Build environments

LCS does not allow AOT or license deployable packages to be installed on build environments. To work around this, remote into the VM and use the **-devinstall** option to install a license deployable package from the command line as described in the topic, Install deployable packages from the command line. This command line install works as of platform update 17. If you are running on a platform version that is older than Platform update 17, and you do not have admin access to your build environment, create a support request and ask Microsoft to install your license deployable package.

Is licensing Visual Studio by entering a product key supported?

Entering a product key directly in Visual Studio is not supported. Instead, use Visual Studio subscription licensing and sign in to Visual Studio with the email address (user account) associated with the license. You can link a Visual Studio license to a user account by assigning an MSDN license to the user account or by assigning a license to the user account by using https://www.visualstudio.com/subscriptions-administration.

Can I upgrade my database to a new application release?

As of the February 2018 release of Lifecyle Services (LCS), you can execute the data upgrade package from the LCS environment page of a development environment. Executing the data upgrade package from LCS does not require you to be an administrator on the VM.

The process described in Upgrade data in development or demo environments runs the data upgrade package from the command line. This requires you to be an administrator on the VM.

What do I need to know if I am developing for Commerce?

If you are developing for Dynamics 365 Commerce, configuration steps and other important information is described in Development in cloud-hosted development environments without admin access.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Rename a local development (VHD) environment

2/18/2021 • 2 minutes to read • Edit Online

A local development (VHD) environment must be renamed for the following scenarios:

- **Accessing a single Microsoft Azure DevOps project across multiple machines:** Azure DevOps is required for version control. In development topologies, multiple virtual machines (VMs) can't access the same Azure DevOps project if they have the same machine name. Azure DevOps uses the machine name for identification. If you're developing on local VMs that were downloaded from Microsoft Dynamics Lifecycle Services (LCS), you might encounter issues.
- **Installing One Version service updates:** One Version service updates, such as 8.1.x, must be installed in VHD environments by using a runbook. To help guarantee that the runbook is completed successfully, the VHD environments must be renamed. Additional steps that are described in this topic must also be completed.

Rename the machine

Rename and restart the machine before you start development or connect to Azure DevOps. Make sure that the new name is unique among all the machines that are used with the Azure DevOps project.

Update the server name in SQL Server

Update the server name in Microsoft SQL Server 2016 by running the following commands.

```
sp_dropserver [old_name];
GO
sp_addserver [new_name], local;
GO
```

In these commands, be sure to replace **old_name** with the old name of the server and **new_name** with the new name. By default, the old name is **MININT-F36S5EH**, but you can run **select @@servername** to get the old name. Additionally, be sure to restart the SQL Server service after the commands have finished running.

Update SQL Server Reporting Services

Update the SQL Server Reporting Service (SSRS) database by using the Reporting Services Configuration Manager. Select **Database**, select **Change Database**, and use the new server name. Make sure that you use Reporting Services Configuration Manager for SQL Server 2016.

Additional steps to install One Version service updates

The following additional steps are required in order to install One Version service updates in a VHD environment.

Update the Azure Storage Emulator

Update the Azure Storage Emulator, and make sure that it's running. From the **Start** menu, open **Microsoft Azure Storage Emulator - v4.0**, and run the following commands.

This command starts the emulator.

```
AzureStorageEmulator.exe start
```

This command verifies that the emulator is running.

```
AzureStorageEmulator.exe status
```

Try the **init** option with the **-server** switch or the **-forcecreate** switch. Be sure to replace **new_name** with the new name.

```
AzureStorageEmulator.exe init -server new_name
AzureStorageEmulator.exe init -forcecreate
```

If the **init** command fails, delete the storage emulator database by using SQL Server Management Studio. Then try the following command.

```
AzureStorageEmulator.exe init
```

When you run this command, you might receive the following error message: "Error: Cannot create database." However, the emulator will usually still start. You just need the emulator to start.

Update financial reporting

Update the server name for financial reporting by using a script that is included in the One Version service update. To get the command, you must download and expand the One Version service update.

Open a Microsoft Windows PowerShell command window as an admin, and run the following command. This command contains the default passwords that might have to be updated. Be sure to replace **new_name** with the new name.

```
cd <update folder>\MROneBox\Scripts\Update
.\ConfigureMRDatabase.ps1 -NewAosDatabaseName AxDB -NewAosDatabaseServerName new_name -NewMRDatabaseName
ManagementReporter -NewAxAdminUserPassword AOSWebSite@123 -NewMRAdminUserName MRUser -NewMRAdminUserPassword
MRWebSite@123 -NewMRRuntimeUserName MRUSer -NewMRRuntimeUserPassword MRWebSite@123 -NewAxMRRuntimeUserName
MRUser -NewAxMRRuntimeUserPassword MRWebSite@123
```

Development system requirements

2/18/2021 • 2 minutes to read • Edit Online

This topic lists the system requirements for development.

Development environments can be hosted locally or in Microsoft Azure. The build process, X++ compilation, and generation of cross reference information, typically run satisfactorily on machines with 16 GB of memory and two CPU cores. The compiler uses available resources, so more RAM and more cores can speed up compilation, especially if there is contention for the resources from other concurrent processes. If you are running concurrent processes, then we recommend 24 GB of memory with four cores. At a minimum, two CPU cores are recommended because the developer environment contains many components that may be running concurrently. The components include the AOS web application, Visual Studio, Management Reporter, and SQL Server.

Version control, metadata search, and navigation

2/18/2021 • 11 minutes to read • Edit Online

This tutorial will walk you through configuring Microsoft Azure DevOps to enable source control on your models. It will also help you learn about other productivity features in the development tools, including the ability to create and organize TODO task, search metadata and source code, navigate between related model elements, and create a project from a model.

Configure your Azure DevOps organization and project

In this section, you'll create a new project in Azure DevOps. This project will host the source code of your model. You'll use the Fleet Management model as an example. If you don't have a Azure DevOps organization, you'll create one.

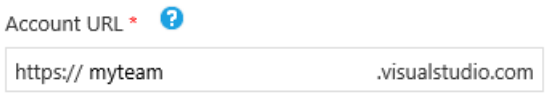
Sign up to Azure DevOps, create an account, and create a new project

Navigate to https://www.visualstudio.com/ to sign up for Azure DevOps. Click **Sign up**. If you already have an account in Azure DevOps, go to the Create a Azure DevOps project section later in this topic.

1. Sign in with your Microsoft account.

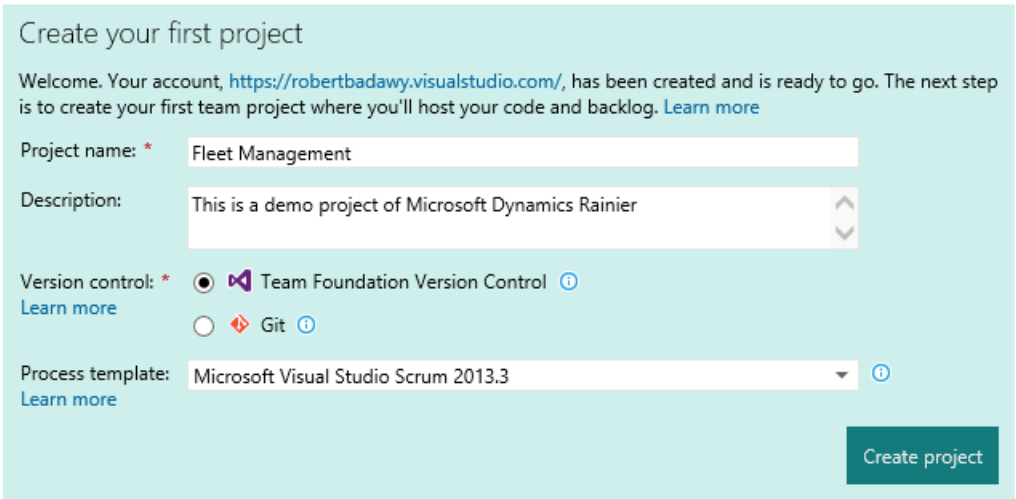
   > **NOTE**
   > You can also use an organizational account (Microsoft 365 domain).

2. Create a Azure DevOps organization, and select a URL for your account. You'll use this URL to connect from your development computer when you're configuring source control in Visual Studio. The following image is an example of the account URL.

   When the account is created, you're directed to your account main page where you're prompted to create your first project.

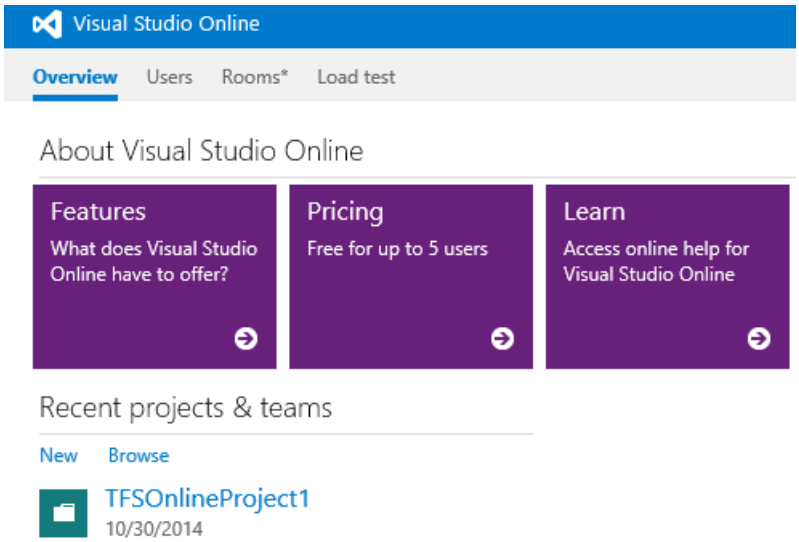
3. Create a demo **Fleet Management** project.

Create a Azure DevOps team project

If you already have a Azure DevOps organization, go to your account using Internet Explorer. This topic uses **.visualstudio.com** as the example URL for illustration purposes.

1. Go to https://www.visualstudio.com/.
2. Under **Recent projects & teams**, click **New** to create a new project.



3. In the **Project name** field, enter **Fleet Management**, enter a **Description**, and then click **Create project**.

Create the recommended folder structure in your team project

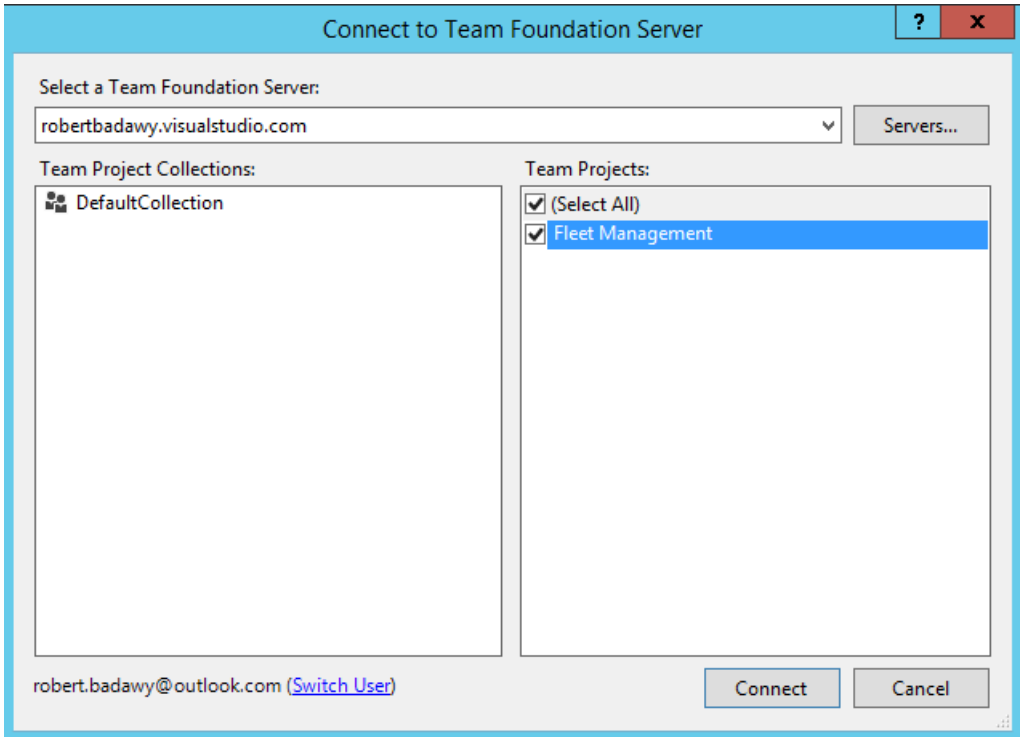
If you have migrated your code from a previous version using the Lifecycle Services (LCS) automated code upgrade tool, the following folder structure is automatically created in your Azure DevOps team project.

The **Metadata** folder contains your source XML files organized by packages and models and the **Projects** folder contains Visual Studio projects. If you are not migrating code and are starting from scratch, create a similar folder structure on the server in your team project before you start development.

Configure Visual Studio to connect to your team project

1. Start Visual Studio. If you are logged into the machine as an administrator, then you must start Visual Studio as an administrator.
2. Click **Tools > Options > Source Control > Plug-in Selection**.
3. In the Current source control plug-in field, select **Visual Studio Team Foundation Server**.
4. Select **Team > Connect to Team Foundation Server**.
5. In **Team Explorer**, click **Select Team Projects**.
6. In the **Select a Team Foundation Server** drop-down list, select the **Azure DevOps organization** that hosts the Fleet Management project, or click **Servers** if it isn't in the menu.
    a. When the **Add/Remove Team Foundation Server** dialog opens, click **Add**.
    b. Enter the URL of your Azure DevOps organization.
    c. Click **OK**.
    d. If prompted, enter your Microsoft Account username and password.

7. Select the **Fleet Management** check box under **Team projects**, and then click **Connect**.

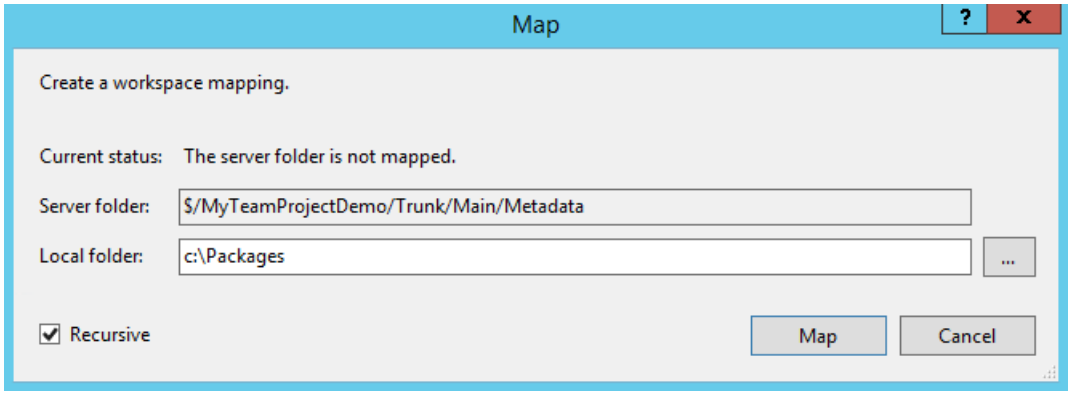


Map your Azure DevOps project to your local model store and projects folder

Your model store root folder contains source files of all packages and models that are part of your application. During deployment, you'll probably use source files from more than one model across more than one package. We recommend that you map your model store root folder to the Azure DevOps team project metadata folder.

1. In Visual studio **Team Explorer**, connect to the team project as described earlier in this document.
2. Open **Source Control Explorer** from **Team Explorer**.
3. Map the **Metadata** folder of your team project to the root folder of the model store on your local drive (Typically K:\AOSService\PackagesLocalDirectory), an example is shown in the image below.

> **NOTE**
> Your model store may be located under I:\AosService\PackagesLocalDirectory or another drive, depending on your machine configuration.



4. Click **Map**, and on the next dialog, click **No**.
5. Similarly, map the **/Trunk/Main/Projects** server folder to the **local projects folder** that will hold your Visual Studio solution and project files.

Scenario 1: Open the fleet management solution and add it to Azure DevOps source control

This section describes the steps needed to add a solution to Azure DevOps source control. This scenario is relevant when you have started development on a new model and you are adding it to source control for the first time. For code migration scenarios or in the case you are synchronizing new models that have been created by another developer, refer to scenario 2 below.

Open the FleetManagement solution

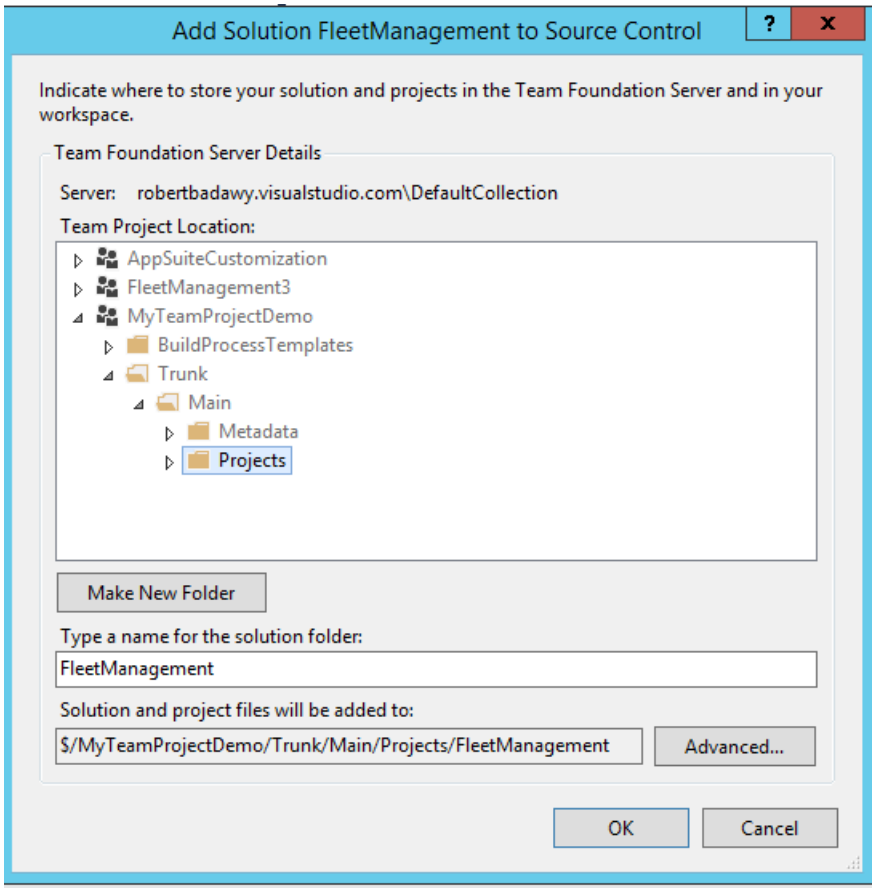
> **NOTE**
> This project is only an example. You can open any project/solution to learn about the process of adding a solution to source control.

1. On the **File** menu, point to **Open**, and then click **Project/Solution**.
2. Browse to the desktop and open the **FleetManagement** folder.
3. Select the solution file named **FleetManagement**. The file type listed is Microsoft Visual Studio Solution. If the solution file is not on your computer, the steps to create it are listed in Tutorial: Create a Fleet Management solution file out of the Fleet Management models in the AOT.
4. Click **Open**. Loading the solution may take some time.

Add the FleetManagement solution to source control

1. In **Solution Explorer**, right-click the Fleet Management solution, and select **Add Solution to Source Control.**
2. On the next dialog, select **Team Foundation Version Control**, and then click **Next**.
3. In the **Team Project Location**, select **Projects**, as shown in this image.

> **NOTE**
> If you have already mapped the server Projects folder to a local folder that contains the FleetManagement solution, steps 2 and 3 are omitted.



4. Click OK.

5. Go to **Team Explorer** > **Pending changes**, and then click **Check-in** to check-in your solution and its model element to the Azure DevOps source control.

Add the model descriptor file to source control

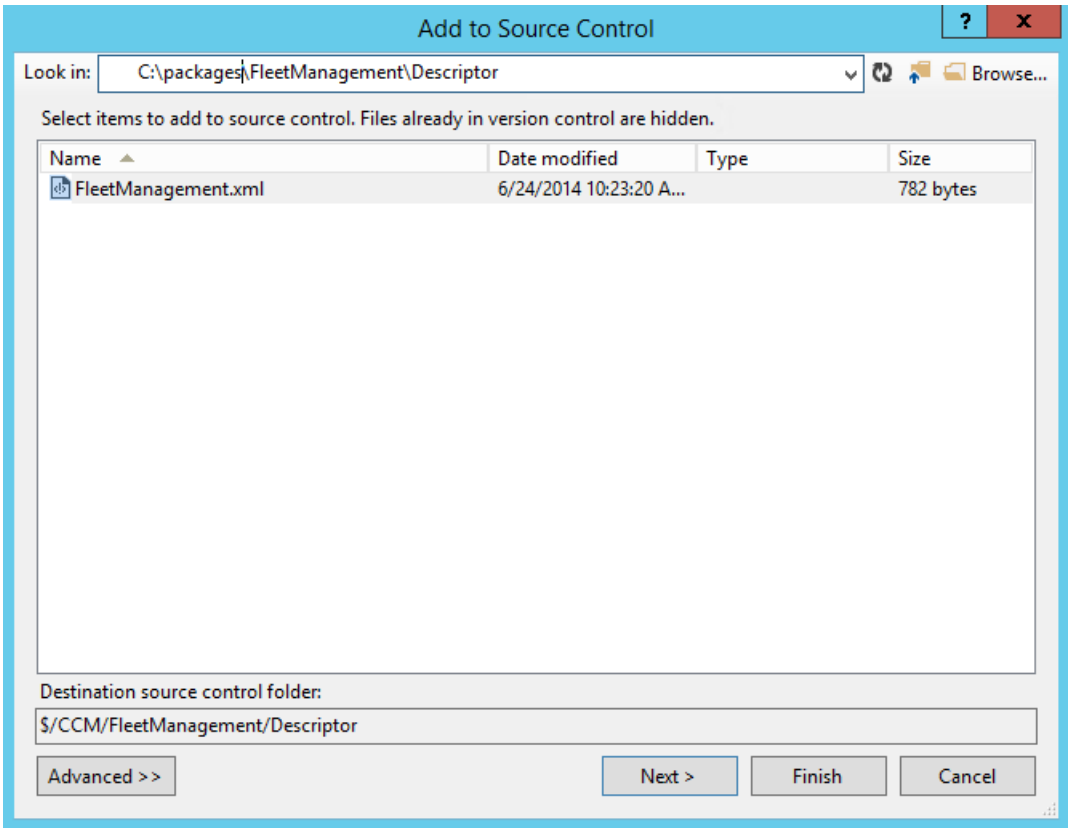
All Visual Studio projects belong to models. Models are source code distribution and deployment units that are typically larger in scope than a Visual Studio project. In the previous section, you added element files of the fleet management solution to source control. Because this was the first time that you added elements of the Fleet Management models to source control, you'll also need to check-in the model descriptor file.

1. In Visual Studio, in **Team Explorer**, open **Source Control Explorer**, and then right-click on the metadata folder (for example, **\Trunk\Main\Metadata**).

2. In the **Source Control Explorer** toolbar, click **Add Item to Folder**.

3. Select your model descriptor file. The model descriptor file is the XML file manifest of your model. It's located in the **Descriptor** folder of the package that the model belongs to. The following image shows an example of where the model descriptor file of the Fleet Management model exists (c:\packages\FleetManagement\Descriptor\FleetManagement.xml).

> **NOTE**
>
> Your model store may be located under I:\AosService\PackagesLocalDirectory or c:\AosService\PackagesLocalDirectory or another drive, depending on your machine configuration.



4. Click **Finish**.

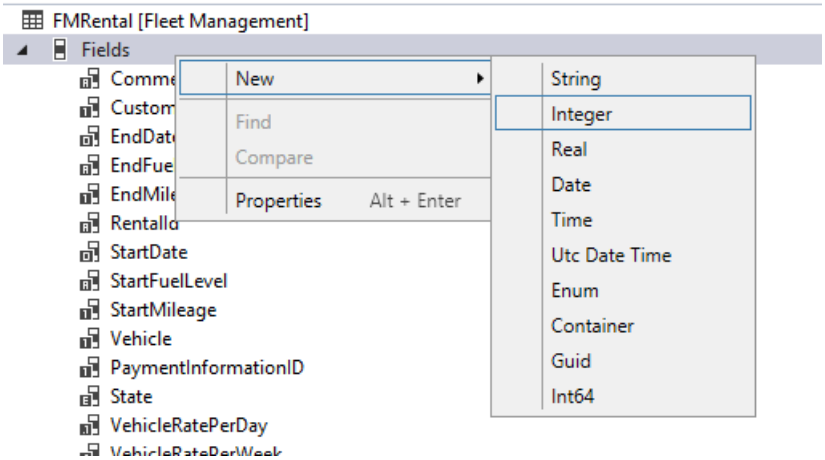
> **NOTE**
>
> Because your solution contained elements from two models, you'll need to add an additional model descriptor file to source control:
> K:\AOSService\PackagesLocalDirectory\FleetManagementExtension\Descriptor\FleetManagementExtension.xml

5. Check-in your pending items. Your item is now ready for development of the fleet management application using a state-of-the-art, cloud-based source control system, and many other application lifecycle features of Azure DevOps.

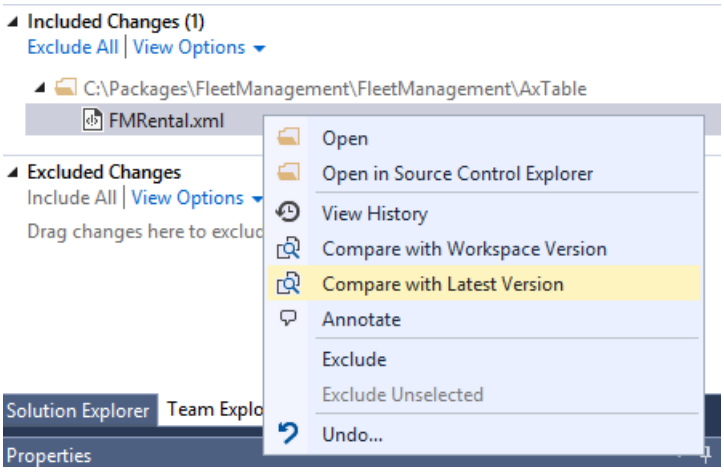
Experiment with source control

In this section, you'll make minor changes to the **FMRental** table and compare your changes with the latest version in your source code repository.

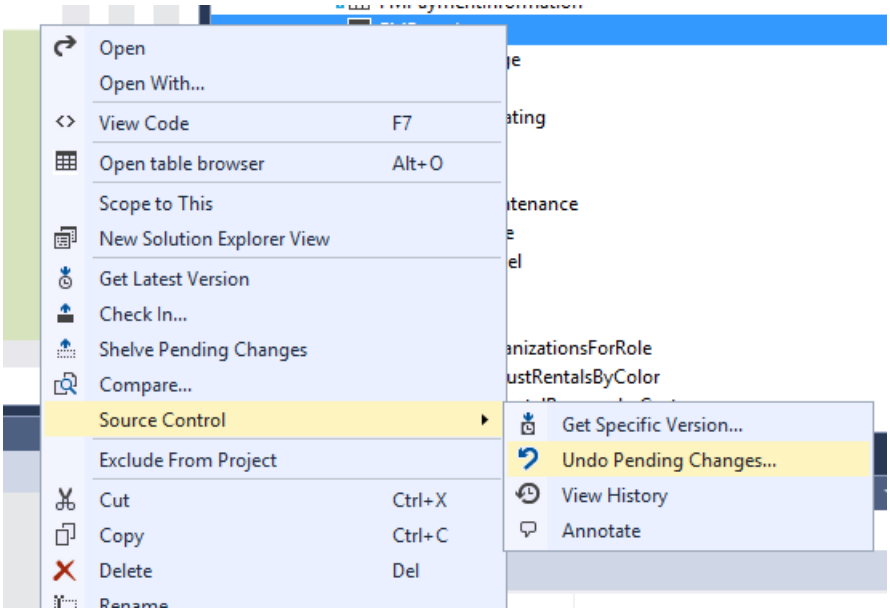
1. In **Solution Explorer**, select **Fleet Management Migrated** > **Data model** > **Tables** > **FMRental**.
2. Double-click **FMRental** to open the designer.
3. Right-click the **Fields** node, and then click **New** > **Integer**.



4. Right-click **Methods**, and add a new method.
5. In the X++ code editor, enter a comment in the new method.
6. Enter a comment in any existing method.
7. Save the **FMRental**.
8. In **Team Explorer**, right-click **FMRental.xml**, and select **Compare with Latest Version**.



9. Browse through the differences in the **comparison (Diff)** window.
10. In **Solution Explorer**, right-click on the **FMRental** table, and select **Source control > Undo > Pending Changes** to revert your changes.



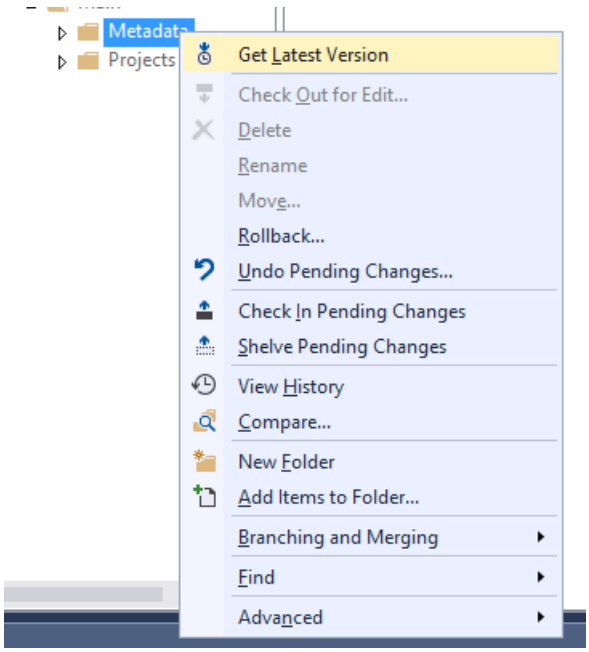
11. Confirm the undo on the next dialog and close the **diff** window.

Scenario 2: Synchronize models from source control

In this section, you will synchronize existing models and model elements from your Azure DevOps project. Synchronization is relevant in the following cases: 1) You have migrated your code from a previous version via LCS, or 2) another developer has checked-in a new model or new model elements and you would like to synchronize them to your development environment.

1. In Source Control Explorer, right-click on Metadata and select **Get Latest Version**. Getting the latest version will synchronize you local packages folder with the latest code.

2. Alternatively you can use the **Advanced** menu to synchronize specific build version or change sets.



3. Once synchronization is complete, and if the synchronization leads to synchronizing new models to your environment, you need to refresh your metadata from Visual Studio.

4. Go to **Dynamics 365 > Model Management > Refresh models**.

Organize TODO tasks in a project

This section describes how you can create a Visual Studio project out of tasks (TODO comments) embedded in your X++ code.

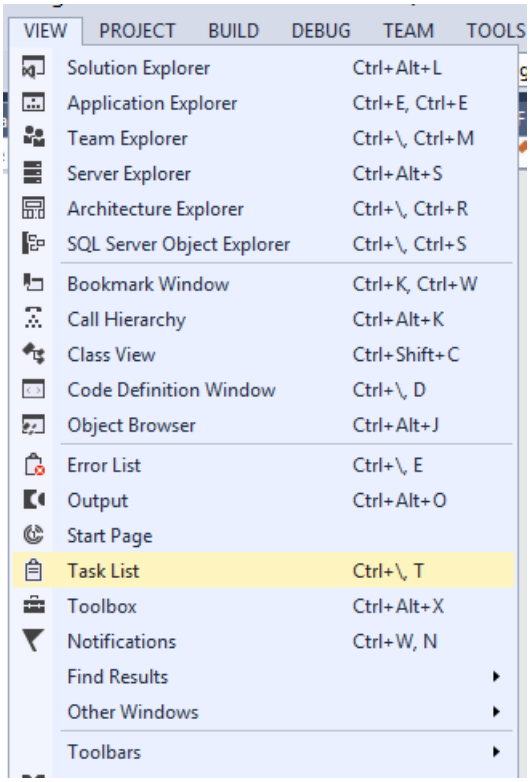
1. In **Solution Explorer**, select **Fleet Management Migrated > Code > Classes > FMDataHelper**, and then double-click **FMDataHelper**. The X++ code editor opens.

2. Enter a TODO comment (//TODO: my comment) inside any method.

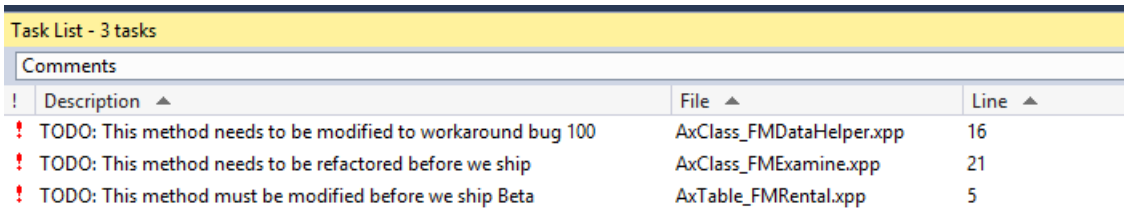
```
/// <summary>
/// Create a number sequence reference for the default scope.
/// </summary>
/// <param name = "_edt">The extended data type of the number seqeunce.</param>
private static void initializeNumberSequence(ExtendedTypeId _edt, boolean resetWizardDefaults)
{
    // TODO: This method needs to be modified to workaround bug 100
    NumberSequenceReference sequenceReference;
    NumberSequenceTable sequence;
    NumberSeqDatatype dataTypeObject;
    NumberSequenceDatatype dataTypeRecord;
    NumberSeqScope scope;
```

3. Open other Fleet Management classes or tables and add more TODO comments.

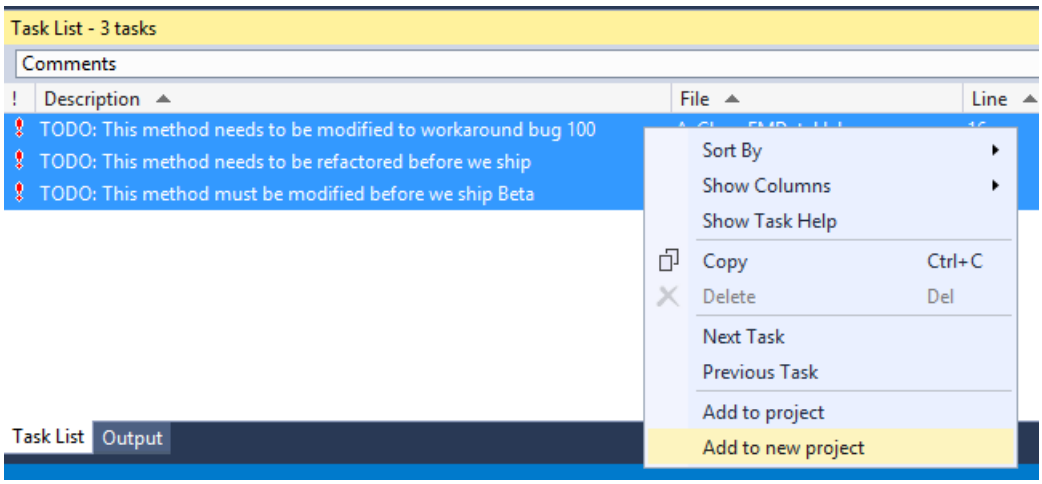
4. Rebuild the **FleetManagement Migrated** project.
5. Select **View** > **Task List**, to open the Visual Studio **Task List** window.



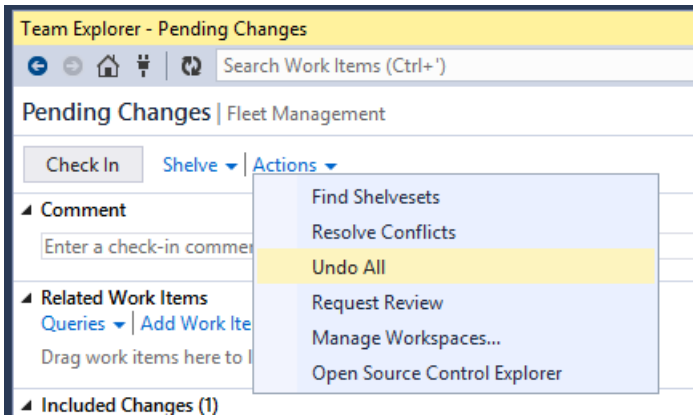
6. Select **Comments** from the drop-down list.



7. Select all TODO items, right-click, and select **Add to new project**.



8. Adding the items will open the **New project** dialog and enable you to create a new project that contains all of your TODOs.
9. You can save this project as a working project to manage your TODO list.
10. When you're finished, undo all of your pending changes in **Team Explorer**.



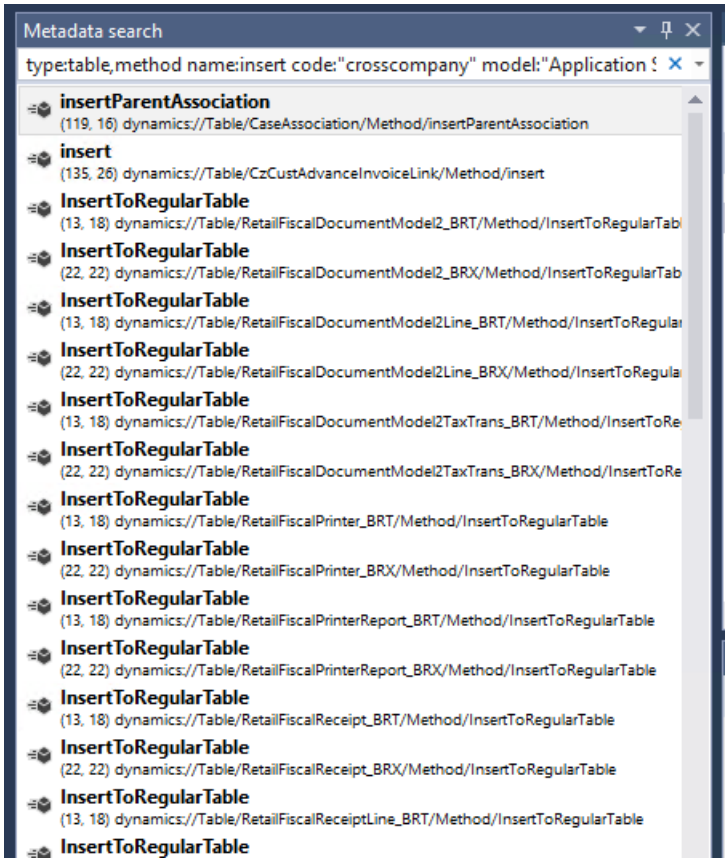
11. Click **File > Close Solution**, to close the FleetManagement solution.

Use metadata search and navigation tools to find elements and create projects

This section demonstrates how you can perform meta-data based searches throughout your application.

Use the Metadata search window

1. Click **Dynamics 365 > Metadata search**.
2. In the **Metadata search** window, in the **Search** field, enter the following text to find all of the table insert methods in the Application Suite model that contain a cross-company query.
   `type:table,method name:insert code:"crosscompany" model:"Application Suite"`.
3. Wait for the search to complete. It may take a while.



4. Double-click a result in the list. The code editor will open and place the cursor at the line that matches your search criteria.
5. Select several elements in the results list by holding down the Ctrl key for multiple selections, right-click, and then select **Add to new project**. Adding the elements will let you to create a new solution and

project containing the selected elements.

Try other search examples

You don't need to wait for the search to complete before you interact with search results. You can double-click results at any time to view the metadata or source code that matches your search criteria. The following are some suggested search examples:

- Find vertical tab controls defined in view mode and autowidth mode in the model Application Suite.

```
type:form,formtabcontrol property:arrangeMethod=Vertical,ViewEditMode=view,WidthMode=Auto
model:"Application Suite"
```

- Find all grid controls in forms that aren't editable and with the property heightmode = column.

```
type:form,formgridcontrol:allowedit=no,heightmode=column
```

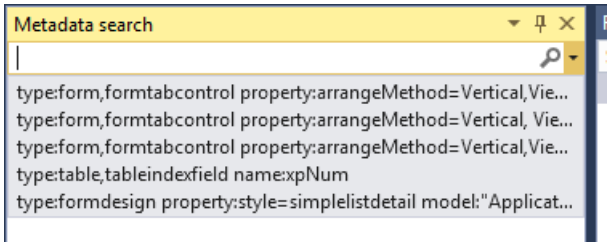
- Find all SimpleListDetail forms in the Application Suite model.

```
type:formdesign property:style=simplelistdetail model:"Application Suite"
```

- Find all tables that have an index field name that contains the keyword xpNum.

```
type:table,tableindexfield anem: xpNum*
```

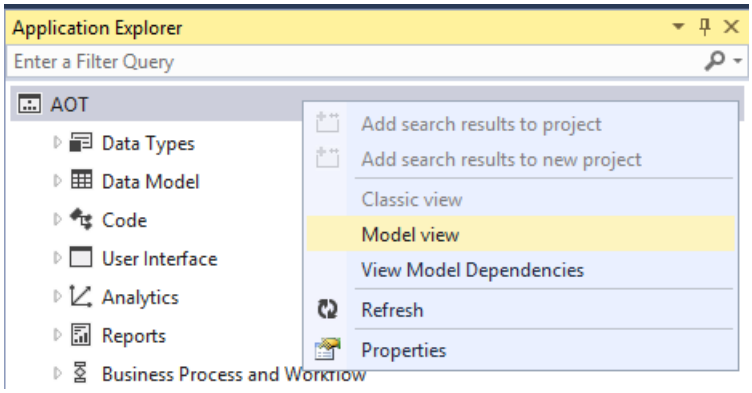
- Use the search bar drop-down menu to access previous searches.



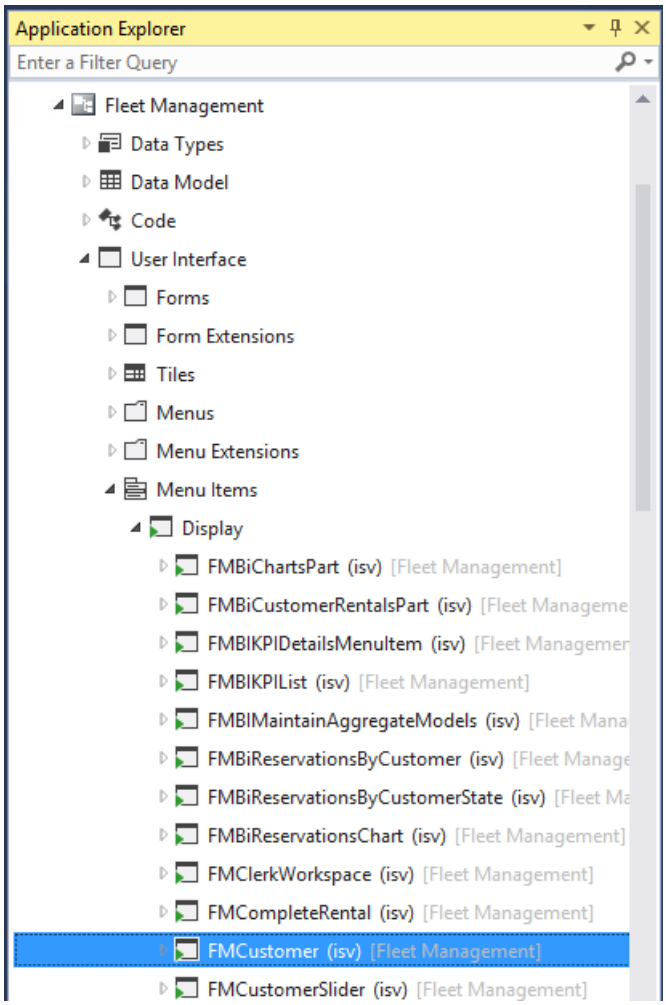
Navigate to related elements

This section highlights a feature that enables you to move from one element to a related element without having to find the related elements in **Application Explorer** or **Solution Explorer**.

1. Open **Application Explorer**, and switch the view to **Model View**.

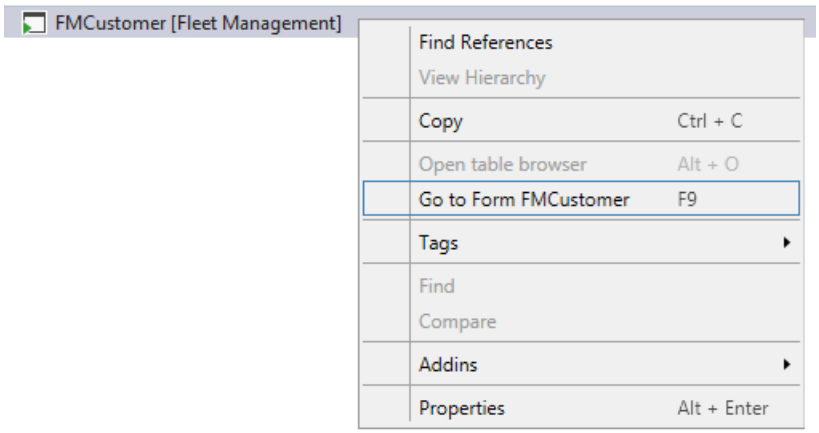


2. Under the **Fleet Management** model, click **User Interface** > **Menu items** > **Display Menu Items** > **FMCustomer**.



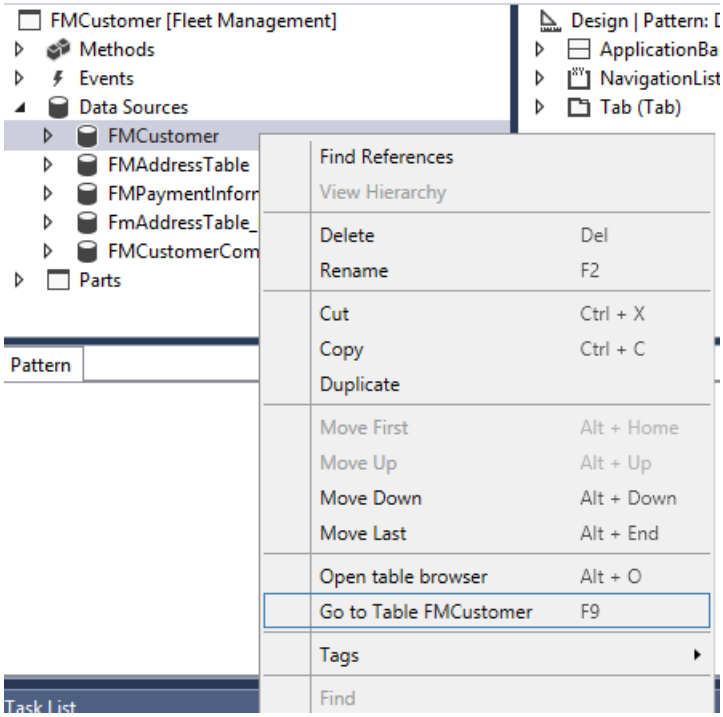
3. Right-click **FMCustomer**, and then select **Open designer**.

4. In the **FMCustomer** menu item designer, right-click the root node, and then select **Go to Form FMCustomer**.



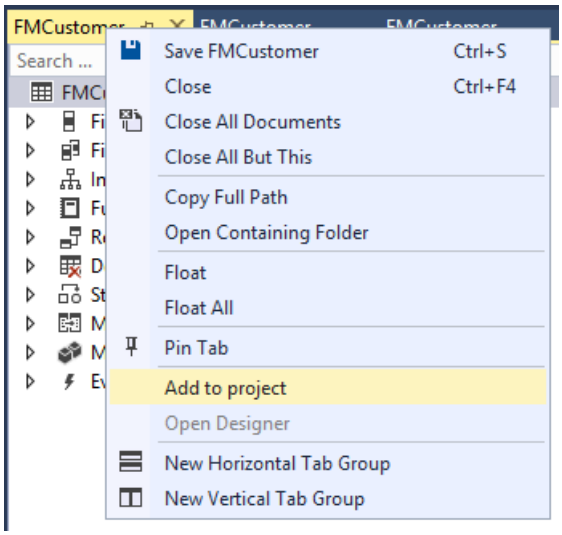
The **FMCustomer** form designer will open.

5. In the designer of the **FMCustomer** form, expand **Data sources**, right-click **FMCustomer**, and then select **Go to Table FMCustomer**



The **FMCustomer** table designer will open.

6. Using the same methodology, you can navigate to the EDT element that a table field references. **Tip**: Press F9 instead of opening the context menu. F9 will open the designer of the referenced element. **Tip**: You can add an element to the current project by right-clicking on the document tab and selecting **Add to project**.

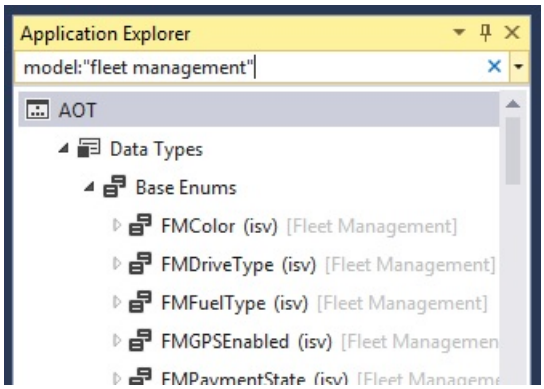


Use Application Explorer to create a project from a model

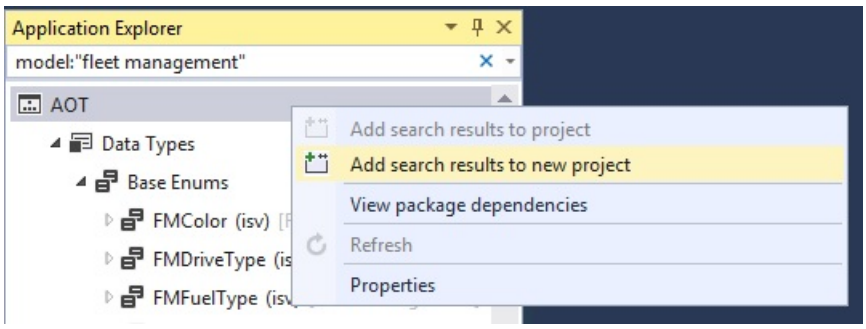
You can use Application Explorer to search for all or some elements of a model and create a project out of the search results.

1. Make sure the option to organize a project by element type is on. Go to **Dynamics 365** > **Options** > **Projects**.

2. Go to Application Explorer and search for elements in the desired model. For example, enter *model:"fleet management"* and click **Enter**.



3. When the search is complete, right-click the AOT root node and select **Add search results to new project.**



4. Specify your project properties in the new project dialog and click **OK** to create the project.

> **TIP**
>
> To create a project from search results, you can add any type, name, or other filters to your search as long as all results are in the same model. For example: *model:"Fleet Management" type:Table name:^FM* will return all tables in the Fleet Management model with a name starting with the letters FM.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Build automation that uses Microsoft-hosted agents and Azure Pipelines

2/18/2021 • 6 minutes to read • Edit Online

You can automate the process of building X++ code and creating deployable packages on any build agent that run on Microsoft Windows. These agents include Microsoft-hosted agents. This approach helps you avoid the setup, maintenance, and cost of deploying build virtual machines (VMs). It also lets you reuse the existing setup of build agents to run other .NET build automation.

> **NOTE**
>
> This feature is limited to compilation and packaging. There is no support for X++ unit testing (SysTest), database synchronization, or other features that require the runtime (Application Object Server [AOS]) or its components.

Prerequisites for building X++ code

Build projects

To use .NET tools for building X++ in Azure DevOps, the Microsoft Build Engine (MSBuild) and custom X++ targets are used. Your X++ source code repository must contain an X++ project for each package that you have to build. You can optionally use a solution file to group the projects, including C# project dependencies, and provide an explicit build order. If the repository doesn't already contain a project, you can create a project in Visual Studio.

> **NOTE**
>
> When you use an existing X++ project (rnrproj), make sure that you created it, or opened and saved it, by using Visual Studio tools on Platform update 27 or later.
>
> Although a package can contain multiple models, it must always be built in its entirety. Therefore, only one project for just one of the models is required to build the whole package. Additionally, although the project doesn't have to contain any objects, it can contain them.

NuGet packages

To build X++ code, the basic developer tools such as the X++ compiler (xppc.exe) are required. Additionally, any referenced packages, such as the Application Platform or Application Suite, must be available in a compiled format. To enable this process, the Shared asset library in Microsoft Dynamics Lifecycle Services (LCS) provides NuGet packages that are required to do an X++ build.

The following packages can be downloaded from the Shared asset library:

- **Microsoft.Dynamics.AX.Platform.CompilerPackage** – This package contains the X++ compiler and related tools that are required to do a build.
- **Microsoft.Dynamics.AX.Platform.DevALM.BuildXpp** – This package contains the compiled X++ code for the Application Platform and related modules. This code is optimized for building.
- **Microsoft.Dynamics.AX.Application.DevALM.BuildXpp** – This package contains the compiled X++ code for the Application Suite and related modules. This code optimized for building.

Download these packages from LCS, and add them to an Azure Artifacts feed in the Azure DevOps organization where the builds will run. For more information about how to create an Azure Artifacts feed and add NuGet

packages, see these topics:

- Get started with NuGet packages in Azure DevOps Services and TFS
- Create a feed
- Create and publish your own NuGet package

> **NOTE**
> Free Azure DevOps organizations have limited storage for Azure Artifacts. Consider deleting old and unused versions to free up storage capacity. For more information, see Sign up for Azure Artifacts.

To identify which packages should be used during the build, and where they can be found, you must provide a nuget.config file and a packages.config file during the build. We recommend that you create these files and add them to the source control repository. The files can be stored anywhere in source control, because the paths of these files are explicit inputs for the NuGet command.

The nuget.config file provides NuGet with the source feed where the packages can be found. The packages.config file specifies the packages and their versions. To build against a newer version, you just have to update the versions in the packages.config file. For more information, including a sample nuget.config file, see Restore Package Management NuGet packages in Azure Pipelines.

The following example shows a packages.config file for the three main packages that are required for a typical X++ build.

```
<?xml version="1.0" encoding="utf-8"?>
<packages>
    <package id="Microsoft.Dynamics.AX.Platform.DevALM.BuildXpp" version="7.0.5644.16778"
targetFramework="net40" />
    <package id="Microsoft.Dynamics.AX.Application.DevALM.BuildXpp" version="10.0.464.13"
targetFramework="net40" />
    <package id="Microsoft.Dynamics.AX.Platform.CompilerPackage" version="7.0.5644.16778"
targetFramework="net40" />
</packages>
```

Creating the pipeline

Azure DevOps provides pipelines that can be used to automate builds. There are two types of pipelines: YML and Classic. YML pipelines are available only when you use Git source control repositories. Classic pipelines must be used to build Team Foundation Version Control (TFVC) repositories. For more information, see Azure Pipelines.

This section describes the steps that are required in a pipeline to build X++ code. In the Dynamics365-Xpp-Samples-Tools GitHub repository, you can find a sample pipeline that can be imported into an existing Azure DevOps project.

Creating a basic build pipeline

A basic pipeline for compiling X++ requires only two steps:

1. Install the NuGet packages.
2. Build the solution or projects.

To simplify use of the extracted NuGet packages, consider using the **NuGet install** option and specifying the **-ExcludeVersion** NuGet command-line option. In that way, the extracted package paths can be used in the build, regardless of the version of the packages. Use the **NuGet Installer** task, and set the **Installation type** field to **Install**. Finally, specify the path of the packages.config and nuget.config files that you created earlier.

The following example of NuGet arguments will prevent a subfolder from being created for the package

versions and will extract the NuGet packages into **$(Pipeline.Workspace)\NuGets**.

```
-ExcludeVersion -OutputDirectory "$(Pipeline.Workspace)\NuGets"
```

To build X++ by using MSBuild, you must supply several arguments. In the pipeline step that builds the solution, you can specify these arguments in the **MSBuild Arguments** field.

ARGUMENT	DESCRIPTION
/p:BuildTasksDirectory	The path of the extracted Compiler Tools NuGet package, including the subfolders in the DevAlm folder.
/p:MetadataDirectory	The path of the X++ source code.
/p:FrameworkDirectory	The path of the extracted Compiler Tools NuGet package.
/p:ReferenceFolder	A semicolon-separated list of paths that contain binaries of X++ packages that are referenced and required for compilation (for example, Application Platform and Application Suite). If the code that will be compiled has multiple packages that reference each other, the output directory should also be included here.
/p:ReferencePath	A semicolon-separated list of paths that contain any non-X++ binaries that are referenced and required for compilation. You should include the location of the extracted Compiler Tools NuGet package, because it might contain required references.
/p:OutputDirectory	The path where the compiler will create folders and binaries.

The following example of MSBuild arguments assumes that the NuGet packages are installed in **$(Pipeline.Workspace)\NuGets**, the X++ source code is in **$(Build.SourcesDirectory)\Metadata**, and the output of the compiler should go in **$(Build.BinariesDirectory)**.

```
/p:BuildTasksDirectory="$(Pipeline.Workspace)\NuGets\Microsoft.Dynamics.AX.Platform.CompilerPackage\DevAlm"
/p:MetadataDirectory="$(Build.SourcesDirectory)\Metadata"
/p:FrameworkDirectory="$(Pipeline.Workspace)\NuGets\Microsoft.Dynamics.AX.Platform.CompilerPackage"
/p:ReferenceFolder="$(Pipeline.Workspace)\NuGets\Microsoft.Dynamics.AX.Platform.DevALM.BuildXpp\ref\net40;$(
Pipeline.Workspace)\NuGets\Microsoft.Dynamics.AX.Application.DevALM.BuildXpp\ref\net40;$(Build.SourcesDirect
ory)\Metadata;$(Build.BinariesDirectory)"
/p:ReferencePath="$(Pipeline.Workspace)\NuGets\Microsoft.Dynamics.AX.Platform.CompilerPackage"
/p:OutputDirectory="$(Build.BinariesDirectory)"
```

In the pipeline samples, variables for NuGet package names and paths are used to simplify these commands.

Creating a full pipeline that includes packaging

To be useful, the pipeline should include a versioning step and a packaging step. Before you can add these steps to a pipeline, the Dynamics 365 Finance and Operations Tools extension for Azure DevOps must be enabled and installed in the Azure DevOps account. For information about how to install an extension for an organization, see the Azure DevOps documentation.

A full pipeline should consist of at least the following steps:

1. Install the NuGet packages.
2. Update the model versions.
3. Build the solution or projects.

4. Install NuGet 3.3.0 or earlier on the agent. (This step is required for the step that creates the deployable package.)
5. Create the deployable package.
6. Publish the deployable package artifact as the build output.

For the deployable package to be created, NuGet must be readily available on the build agent. Therefore, the **NuGet tool installer** task in Azure DevOps must be run before the step that creates the package.

> **NOTE**
>
> Because of semantic versioning features in NuGet version 3.4 and later, make sure that the task installs version 3.3.0 or earlier. Currently, deployable package generation doesn't support semantic versioning.

Sample pipeline for X++ developers

In the Dynamics365-Xpp-Samples-Tools GitHub repository, you can find a sample pipeline that can be imported into an existing Azure DevOps project.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Add license files to a deployable package in Azure Pipelines

2/18/2021 • 2 minutes to read • Edit Online

When you update an environment by using a deployable package, a license might be required for independent software vendor (ISV) or partner X++ solutions. ISVs can create pipelines to automatically include licenses in release or build pipelines. Customers can create their own pipelines to combine the ISV deployable package and the license file.

This topic assumes a working knowledge of Azure Pipelines.

> **NOTE**
>
> Before you can add these steps to a pipeline, the Dynamics 365 Finance and Operations Tools extension for Azure DevOps must be enabled and installed in the Azure DevOps account. For more information about how to install an extension for an organization, see Install extensions.

Adding the task to a pipeline

To add the task to your build for the YML or Classic pipeline, search the task list for **Add Licenses to Deployable Package**. The following table describes the options that are available for this task.

INPUT NAME	MANDATORY	DESCRIPTION
Search pattern for license files to add to the package	Yes	A list of license files on the build agent, or a search pattern for files on the build agent. To make the license files available on the build agent, you can add them to source control. Alternatively, they can be downloaded or generated in an earlier step of the pipeline. For more information, see File matching patterns reference.
Filename and path of the deployable package to update	Yes	The path and file name of an existing deployable package zip file that the license files should be added to.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Create deployable packages in Azure Pipelines

2/18/2021 • 2 minutes to read • Edit Online

If you want to deploy customizations to an environment, a deployable package is required in Microsoft Dynamics Lifecycle Services (LCS). You can create this package by using Azure Pipelines during a build or release process.

This topic assumes a working knowledge of Azure Pipelines.

> **NOTE**
>
> Before you can add these steps to a pipeline, the Dynamics 365 Finance and Operations Tools extension for Azure DevOps must be enabled and installed in the Azure DevOps account. For more information about how to install an extension for an organization, see Install extensions.
>
> This Azure DevOps task requires that the X++ compiler tools be available on the agent. Either run this task on a build virtual machine (VM) agent, or use the Compiler Tools NuGet package. For more information about the NuGet package and how to install it in a pipeline, see Build automation using Microsoft-hosted agents and Azure Pipelines.

Add the task to a pipeline

To add the task to the build of your YML or Classic pipeline, search the task list for **Create Deployable Package**. The following table describes the options that are available for this task.

INPUT NAME	MANDATORY	DESCRIPTION
X++ Tools Path	Yes	The path of the location of the X++ tools. This location is either the **PackagesLocalDirectory\bin** folder location on a build VM, or the location of the extracted NuGet file of the Compiler Tools NuGet package.
Location of the X++ binaries to package	Yes	The path that contains the folders that contain all the binaries for the X++ packages (modules) that you want to include in the deployable package. If this task is used in a build pipeline, this folder is typically the same as the compiler output folder.
Search pattern for binaries to package	Yes	Provide a name matching pattern for X++ package (module) names inside the path that is specified in the **Location of the X++ binaries to package** option. You can also specify a list of names instead of search patterns, or you can specify exclusion filters so that, for example, test packages aren't included. For more information, see File matching patterns reference.

INPUT NAME	MANDATORY	DESCRIPTION
Filename and path for the deployable package	Yes	The path and file name of the deployable package. The output file is a zip file, and the file name typically includes version information to make the file easy to identify.

NuGet dependency

When this task is run on the build VM, NuGet is already available, and no action is required. However, when this task is run on hosted agents or other private agents, NuGet must be installed. In this case, Azure DevOps has the NuGet Tool Installer task that you can run before you run the task to create the package.

> **NOTE**
>
> Because of the introduction of semantic versioning in NuGet version 3.4 and later, you must install version 3.3.0 or earlier.

X++ model-versioning in Azure Pipelines

2/18/2021 • 2 minutes to read • Edit Online

During build automation, X++ model versions can be updated so that they match or are linked to the build number of the pipeline. These updates make it easier for customers to identify the version of the X++ packages that they are running. They also let developers track versions back to the build pipeline and the version of the source code files.

This topic assumes a working knowledge of Azure Pipelines.

> **NOTE**
> Before you can add these steps to a pipeline, the Dynamics 365 Finance and Operations Tools extension for Azure DevOps must be enabled and installed in the Azure DevOps account. For more information about how to install an extension for an organization, see Install extensions.

Add the task to a pipeline

To add the task to the build of your YML or Classic pipeline, search the task list for **Update Model Version**. The following table describes the options that are available for this task.

INPUT NAME	MANDATORY	DESCRIPTION
X++ Source Location	Yes	The path of a parent folder that contains X++ source code. The **Descriptor Search Pattern** option will be run in this path and subfolders. The default value, **$(Build.SourcesDirectory)**, points to the root of the source code repository.
Descriptor Search Pattern	Yes	Provide a file matching pattern to find the descriptor files inside the path that is specified in the **X++ Source Location** option. You can also specify a list of full paths of descriptor files instead of search patterns. For more information, see File matching patterns reference.
Lowest Layer to Update	Yes	When using search patterns, this task may find descriptors for ISV or partner code in your source control. Those descriptor versions should likely not be updated. This option allows you to define an additional filter to define the lowest layer that the task will update.

INPUT NAME	MANDATORY	DESCRIPTION
Version number in format #.#.#.# to update	Yes	The version number to write into the descriptors. Note that the format must consist of four digits that are separated by periods (.). By using the build ID or build number, you allow for traceability. Note that the default build number isn't in the correct format. You can change the format in the classic editor, under **Options** > **Build number format**. You can also change it by adding a `name:` tag at the top of the YML file. An example of a valid build number that uses the year, month, and date, and also the build count revision number, is `$(Date:yy.MM.dd)$(Rev:.r)`. For more information, see Configure run or build numbers.

Download assets by using Azure Pipelines

2/18/2021 • 2 minutes to read • Edit Online

You can automate the download of assets from the Asset library in Microsoft Dynamics Lifecycle Services (LCS) by using the **Deploy Lifecycle Services (LCS) Asset Download** task in Azure DevOps.

This topic assumes that you have a working knowledge of Azure Pipelines.

> **NOTE**
>
> Before you can add these steps to a pipeline, the Dynamics 365 Finance and Operations Tools extension for Azure DevOps must be enabled and installed in the Azure DevOps account. For more information about how to install an extension for an organization, see Install extensions.

Add the task to a pipeline

To add the task to the build of your YML or Classic pipeline, search the task list for **Dynamics Lifecycle Services (LCS) Asset Download**.

The following table describes the options that are available for this task.

INPUT NAME	MANDATORY	DESCRIPTION
LCS Connection	Yes	Select or create a service connection to LCS. For more information, see Create an LCS connection in Azure Pipelines.
LCS Project ID	Yes	Enter the ID of the project in LCS that contains both the asset to deploy and the target environment. You can find the project ID at the end of the URL of your project's dashboard.
Path to download to	Yes	Enter the path to download the asset to.

INPUT NAME	MANDATORY	DESCRIPTION
Search Pattern	Yes	Select the type of search pattern that should be used to find the asset in the Asset library in LCS. Depending on the value that you select, the following options are available: • **Asset ID (guid)** – If you select this value, in the **LCS File Asset Id(s)** field, enter the asset ID or a semicolon-separated list of asset IDs. Asset IDs are globally unique identifiers (GUIDs). • **Name** – If you select this value, in the **LCS File Asset Type** field, select the asset type. Then, in the **LCS File Asset Name** field, specify the name to search for. You can use an asterisk (*) as a wildcard character in the name. For example, you might enter **MyPackage***.

After a successful download, an output variable can be used to capture a list of the file paths. If there are multiple files, a semicolon-separated list of file paths is assigned to the output variable. For more information about output variables in Azure DevOps, see Use output variables from tasks.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Upload assets by using Azure Pipelines

2/18/2021 • 2 minutes to read • Edit Online

You can automate the upload of assets to the Asset library in Microsoft Dynamics Lifecycle Services (LCS) by using the **Deploy Lifecycle Services (LCS) Asset Upload** task in Azure DevOps. This task is available only in **Releases** pipelines.

This topic assumes you have a working knowledge of Azure Pipelines.

> **NOTE**
> Before you can add these steps to a pipeline, the Dynamics 365 Finance and Operations Tools extension for Azure DevOps must be enabled and installed in the Azure DevOps account. For more information about how to install an extension for an organization, see Install extensions.

Add the task to a pipeline

To add the task to the build of your YML or Classic pipeline, search the task list for **Dynamics Lifecycle Services (LCS) Asset Upload**.

The following table describes the options that are available for this task.

INPUT NAME	MANDATORY	DESCRIPTION
LCS Connection	Yes	Select or create a service connection to LCS. For more information, see Create an LCS connection in Azure Pipelines.
LCS Project ID	Yes	Enter the ID of the project in LCS that contains both the asset to deploy and the target environment. You can find the project ID at the end of the URL of your project's dashboard.
Type of asset	Yes	Select the type of asset to upload.
File to upload	Yes	Enter the path of the file that you want to upload into the Asset library in LCS.
LCS Asset Name	No	Enter the name to show for the asset in the Asset library. If you don't enter a name here, the file name will be used.
LCS Description	No	Enter the description to show for the asset in the asset details.
Wait for Validation	No	For asset types that require validation, use this check box to instruct the task to wait until validation of the asset has either succeeded or failed.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Deploy assets by using Azure Pipelines

2/18/2021 • 2 minutes to read • Edit Online

You can use the **Deploy Lifecycle Services (LCS) Asset Deployment** task in Microsoft Azure DevOps to automate the deployment of assets that are stored in the Asset library in Microsoft Dynamics Lifecycle Services (LCS) to specific environments. However, this task has the following limitations that you should consider:

- The task is available only in **Releases** pipelines.
- The deployment of software deployable packages to production environments can't be automated.
- Software deployable packages can't be deployed to build environments.
- Software deployable packages can't be deployed to local business data (LBD) environments on-premises.

This topic assumes that you have a working knowledge of Azure Pipelines.

> **NOTE**
> Before you can add these steps to a pipeline, the Dynamics 365 Finance and Operations Tools extension for Azure DevOps must be enabled and installed in the Azure DevOps account. For more information about how to install an extension for an organization, see Install extensions.

Add the task to a pipeline

To add the task to the build of your YML or Classic pipeline, search the task list for **Dynamics Lifecycle Services (LCS) Asset Deployment**. If your target environments include self-service environments, be sure to select task version 1.* or later.

The following table describes the options that are available for this task.

INPUT NAME	MANDATORY	DESCRIPTION
LCS Connection	Yes	Select or create a service connection to LCS. For more information, see Create an LCS connection in Azure Pipelines.
LCS Project ID	Yes	Enter the ID of the project in LCS that contains both the asset to deploy and the target environment. You can find the project ID at the end of the URL of your project's dashboard.
LCS Environment ID	Yes	Enter the ID of the target environment. The environment ID is a globally unique identifier (GUID) that you can find on the environment's details page, under **Environment Details** > **Environment ID**.

INPUT NAME	MANDATORY	DESCRIPTION
LCS File Asset ID	Yes	Enter the asset ID of the software deployable package to deploy. The asset ID is a GUID that you can find in the Asset library. Select the row of the asset that you want to deploy, and then, under **Additional Details**, look in the **Asset ID** field. Typically, this ID comes dynamically from other pipeline steps, such as the Dynamics Lifecycle Services (LCS) Asset Upload task.
Name for the update	Yes	Enter the name that is shown for the update in the environment history in LCS.
Wait for Completion	Cleared (No)	Use this check box to instruct the task to wait until the deployment of the asset has either succeeded or failed. If it's cleared (**No**), the task will only start the deployment. If the task is instructed to wait, a pipeline time-out might occur during long-running deployments. For more information about time-out options, see Timeouts.

Create an LCS connection in Azure Pipelines

2/18/2021 • 3 minutes to read • Edit Online

The Dynamics 365 Finance and Operations Tools extension for Microsoft Azure DevOps has several pipeline tasks that let you perform actions in Microsoft Dynamics Lifecycle Services (LCS). For example, you can upload assets, download assets, and service an environment. For the connection with LCS to work, you must set up a new service connection in Azure DevOps. This service connection provides the authentication details that are required to connect to LCS. For more information about service connections in Azure DevOps, see Service connections.

This topic assumes that you have a working knowledge of Azure Pipelines.

> **NOTE**
>
> Before you can add these steps to a pipeline, the Dynamics 365 Finance and Operations Tools extension for Azure DevOps must be enabled and installed in the Azure DevOps account. For more information about how to install an extension for an organization, see Install extensions.

Prerequisites

You must have the credentials for a user who has access to one or more LCS projects that you want to interact with from Azure DevOps. Make sure that this user has successfully signed in to LCS before, and has opened the dashboard for the projects that you want to interact with.

> **NOTE**
>
> LCS doesn't support service-to-service authentication. Therefore, only regular user credentials (that is, a user name and password) can be used. Because the pipelines don't run interactively, multifactor authentication must not be set up for the account that you use. We recommend that you set up a separate user account that has limited access and strong credentials that can regularly be rotated for security purposes.

To enable direct connections from Azure DevOps to LCS on a user's behalf, you must register an application in your Azure Active Directory (Azure AD).

1. Follow the instructions in Quickstart: Register an application with the Microsoft identity platform, and add a new redirect URI:
   a. Select **Public client/native (mobile & desktop)**.
   b. Enter any valid URI, such as `http://localhost`.
2. Add permissions to the application registration to access the LCS web APIs. Follow the instructions in Add permissions to access your web API. When you request the API permissions, select **APIs my organization uses**, and search for **Dynamics Lifecycle services**.
3. Make sure that the account that you will use has given consent for the application registration in Azure AD. Follow the instructions in Configure the way end-users consent to an application in Azure Active Directory. You can either enable a specific user or grant admin consent for the whole tenant.

Create the Dynamics Lifecycle Services service connection

You can create a new service connection either directly from a pipeline task or from your project's settings page.

For more information about how to create service connections, see Create a service connection. In the dialog box for the **Dynamics Lifecycle Services** service connection, provide the following information:

- **Authentication Endpoint** – The default value works for all Azure AD tenants in the Azure cloud. If your Azure AD is in a national cloud, see National clouds to find the correct authentication endpoint.
- **Lifecycle Services API Endpoint** – Provide the endpoint.
- **Username** – Provide the email alias for the user credentials.
- **Password** – Provide the password for the user credentials.
- **Application (Client) ID** – Provide the ID that was previously created for your application registration in Azure AD.
- **Service connection name** – Provide a meaningful name for the connection. This name will appear in the connection field for pipeline tasks that require an LCS connection.
- **Description** – Provide an optional description of this connection.

Update a legacy pipeline in Azure Pipelines

2/18/2021 • 2 minutes to read • Edit Online

> **NOTE**
> This documentation does not apply to the new build pipeline, even if you run it on the build virtual machine.

> Even though **Visual Studio 2017** is available on virtual machines deployed with version 10.0.13 or later, the build scripts needed to support unit testing using **VSTest** require version 10.0.15.

An **Azure Pipelines** pipeline explicitly specifies the versions of **Visual Studio** tools such as **MSBuild** and **VS Test**. To continue supporting newer versions of these products, new virtual machines will be deployed with newer versions of **Visual Studio** pre-installed. Any existing pipelines need to be manually updated to use the newer version.

Determine if your pipeline needs to be updated

Build and development virtual machines deployed with version 10.0.13 include **Visual Studio 2017**. Support for Visual Studio 2015 will be deprecated in the April 2021 release. If your build virtual machine does not include **Visual Studio 2017** you must plan to deploy a new build virtual machine with version 10.0.13 or later, or consider using the new hosted build pipeline.

If your build virtual machine has Visual Studio 2017 installed, you can use the newer versions of **MSBuild** and **VS Test**. If your pipeline was created by a virtual machine deployment prior to version 10.0.13, you will have to manually update the pipeline.

Finally, you can check your build pipeline on the **Build the solution** step. The **MSBuild Version** property indicates the version of **Visual Studio** in use.

MSBUILD VERSION	VISUAL STUDIO VERSION
MSBuild 14.0	Visual Studio 2015
MSBuild 15.0	Visual Studio 2017

Updating the Azure Pipelines pipeline

> **NOTE**
> To update an **Azure Pipelines** pipeline to use **Visual Studio 2017**, ensure your build virtual machine has been updated to version 10.0.15 or newer.

The following four properties, in three tasks in the pipeline, need to be updated.

TASK NAME	TASK PROPERTY	OLD VALUE	NEW VALUE
Build the solution	MSBuild Version	MSBuild 14.0	MSBuild 15.0
Database Sync	MSBuild Version	MSBuild 14.0	MSBuild 15.0
Execute Tests	Test platform version	Visual Studio 2015	Visual Studio 2017

TASK NAME	TASK PROPERTY	OLD VALUE	NEW VALUE
Execute Tests	Other console options	`/Platform:X64 /InIsolation /UseVsixExtensions:true`	`/Platform:X64 /InIsolation /TestAdapterPath:"$(VsixExtensionFold`

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

End-to-end scenario for the Fleet Management sample application

2/18/2021 • 8 minutes to read • Edit Online

This tutorial walks you through an end-to-end scenario that the Fleet Management sample application is designed to support.

In this tutorial, you'll take a tour of the Fleet Management sample. The overviews in this tutorial provide some background knowledge and contextual info. You'll walk through an end-to-end scenario that this sample application is designed to support. This is information that you should have before proceeding to other tutorials.

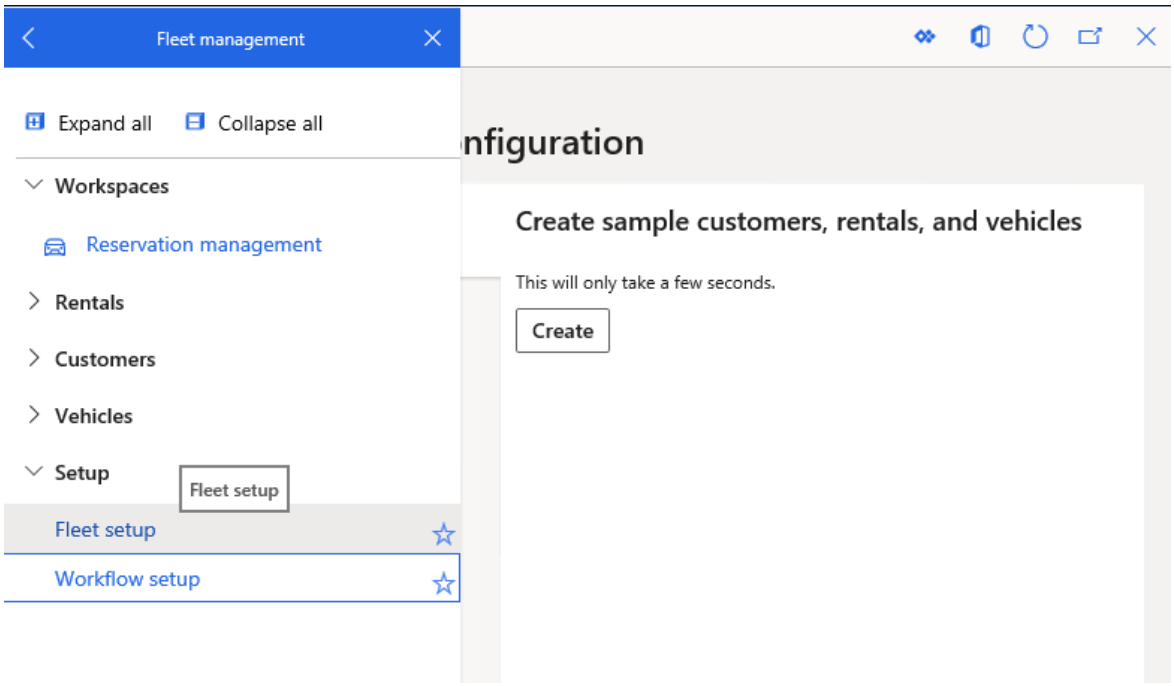
Prerequisite

- You must first be provisioned as an end user before you start this tutorial.
- This tutorial mainly explores the FleetManagement Migrated project and the application that it builds.

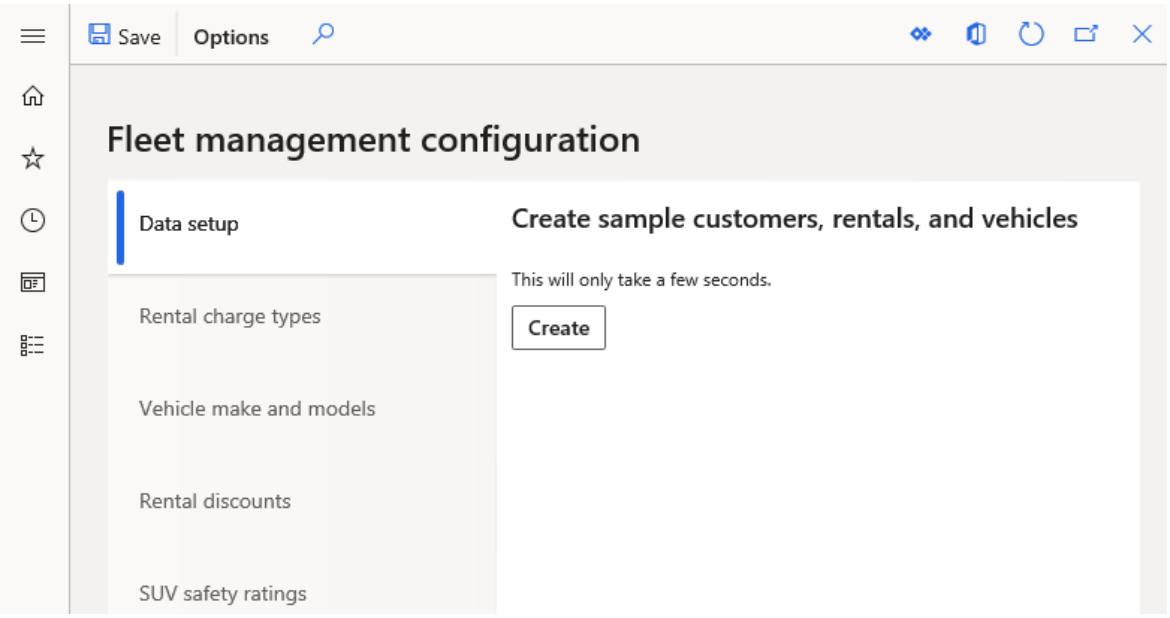
Installing the demo data

To work with the sample, you must install the provided demo data.

1. In the virtual machine (VM), open Internet Explorer and navigate to the application's base URL.
2. Sign in.
3. On the dashboard, open the navigation pane and go to **Fleet Management** > **Setup**> **Fleet setup**.



4. On the **Data setup** tab, select **Create**.



5. If you're prompted to reload the demo data, click **Yes**.
6. When the data is finished loading, select **Close**.

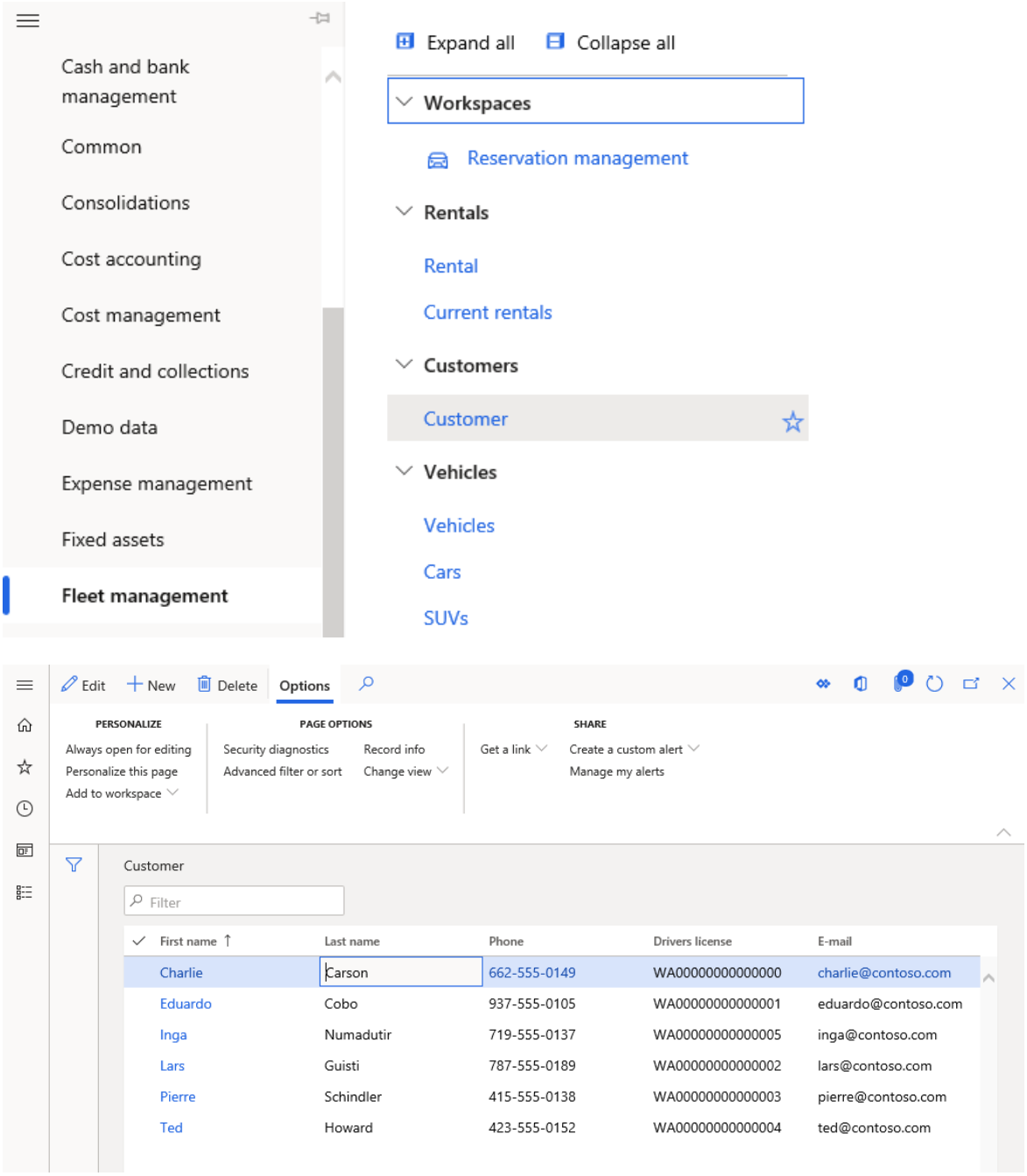
Use the Fleet Management application to rent a vehicle

Remember that you're working with the migrated app in this section. The forms that you see are directly ported from the Microsoft Dynamics AX 2012 version of the sample. Although they have been modified and restyled, they have not been reimagined.

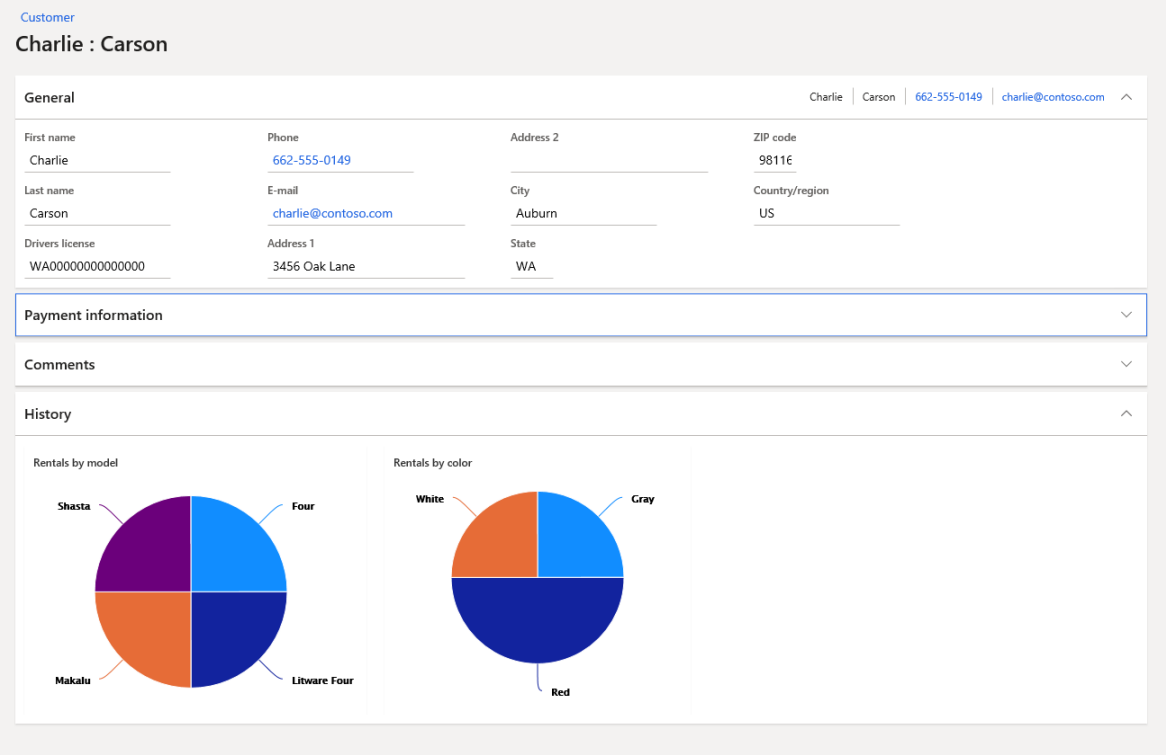
1. Open Internet Explorer, and sign into the Finance and Operations application.
2. To return to the **Dashboard**, select the product name in the top-left corner of the page.

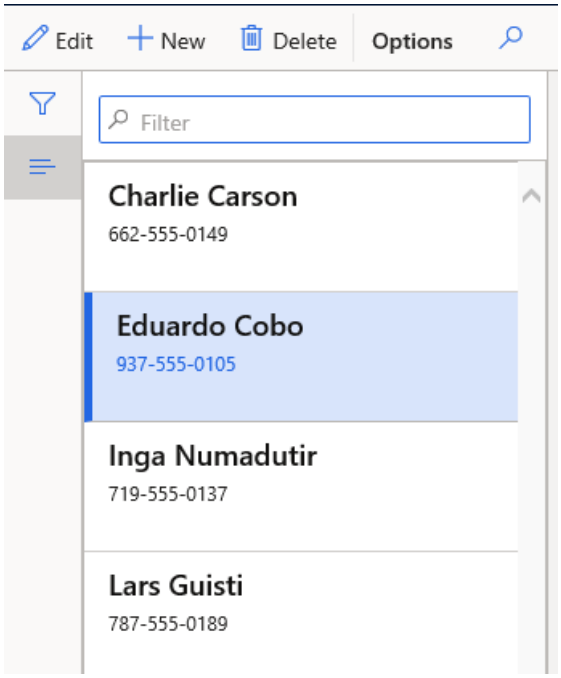
   The dashboard is the main working hub. You can see the various tiles, organized into sections, which lead to parts of the application. The dashboard is designed for horizontal scrolling, which is an optimization for working well on modern devices. The button to the right of the dashboard shows the navigation bar.
3. From the Dashboard, open the navigation bar and go to **Fleet Management** > **Customers** > **Customer**.



4. To switch to the **Details** view, select a value in the **First Name** column. This view shows detailed information for a single customer.

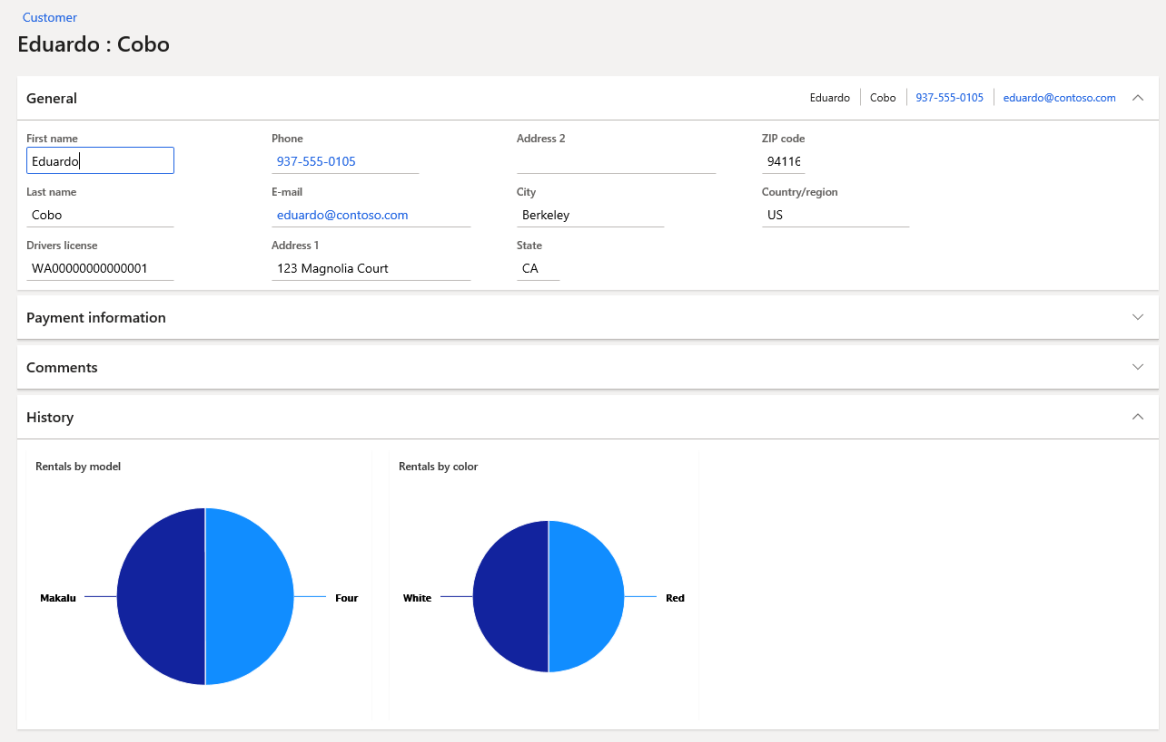


5. Click **Show list** to show the navigation list.



6. Click the various customer names in the navigation list in the side pane, and watch as the detailed information about each customer changes.

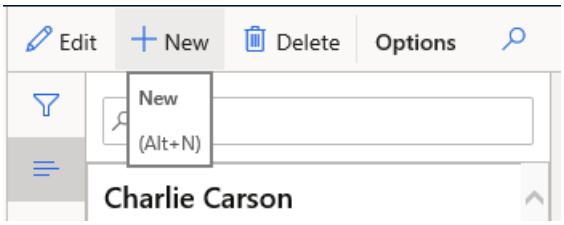
7. Select the customer **Eduardo Cobo**. You'll notice the charts update to indicate Eduardo's previous rental preferences.



8. Hover over the pie slices to see the details. You'll notice that, in the past, Eduardo has often rented red SUVs. This might give the sales clerk a cue to look for available red SUVs the next time Eduardo makes a reservation. This is a simple example of proactively providing insights.

9. Add yourself as a customer.

   - On the Action Pane, click **New**.



   - Fill in the form to add yourself as a customer. Make sure that you provide your name, a 16-digit number in the credit card field, and address information, at a minimum. **Note**: You don't have to take any action to save a new record.

10. Create a new rental.

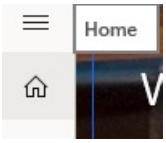
    a. On the navigation bar, go to **Fleet management** > **Rentals** > **Rental**.
    b. In the **Rental** form, on the Action Pane, click **New.**
    c. In the **Vehicle** field, select a vehicle.
    d. In the **Customer** field, select your name.
    e. In the **To** field, pick an end date.
    f. In the **Start** field, enter **35,000**.
    g. In the **Pickup** field, enter **Full**.
    h. When you are done, click **Save**.

11. Start the rental period.

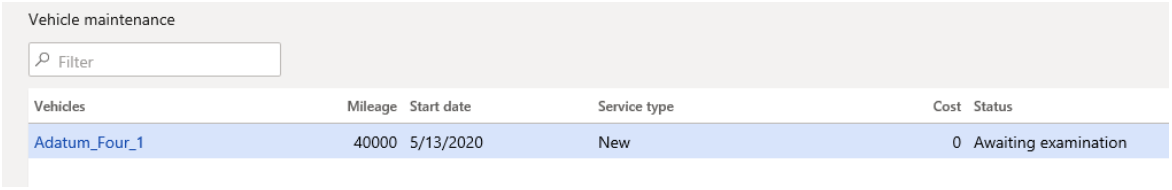
    a. On the Action Pane, click **Start rental**.
    b. In the dialog box, verify the values in the fields and click **OK**.

Use Fleet Management to run a workflow

1. Click the **Home** icon to return to the dashboard.

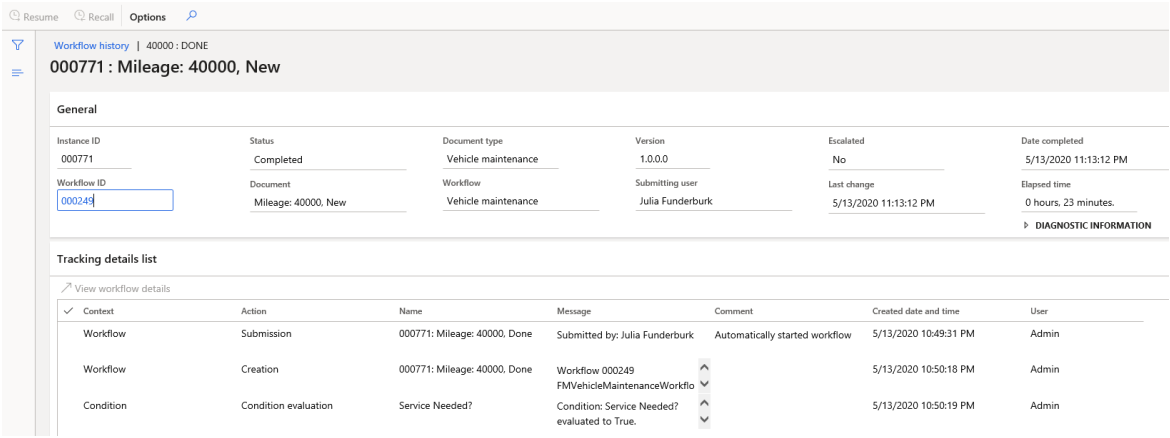


2. Find the **Reservation Management** tile and select it to open the Reservation Management workspace.
3. Click **Current rentals**.
4. On the **Rentals** form, click the ID of your rental.
5. On the Details view of the **Rentals** form, on the Action Pane, click **Complete rental**.
6. In the **New mileage** field, enter **40,000**, and then click **OK**.
7. Click the **Home** icon to return to the dashboard.
8. On the navigation bar, navigate to **Fleet management** > **Vehicles** > **Vehicle Maintenance**. In the **Vehicle Maintenance** form, the **Status** field shows that your rental is awaiting examination by the service department.



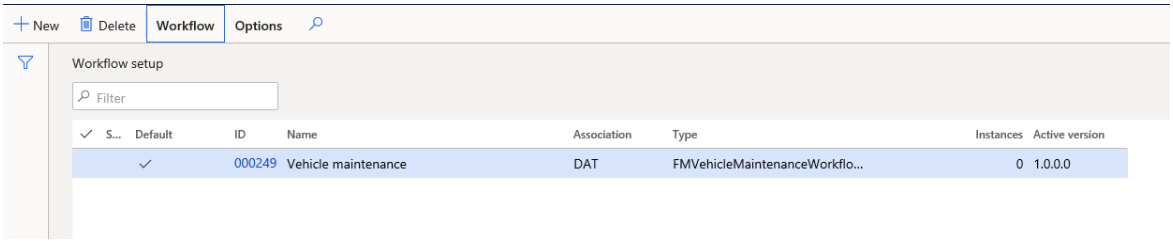
> **NOTE**
>
> You might need to wait up to two minutes for the batch framework to change the status of the vehicle. On the Action Pane, click **Refresh** periodically to update the view, until you see the status change. Keep in mind that a different person usually handles each step in a workflow; the brief delay introduced by the batch framework is not an issue in a real-world application.

9. Select the row that contains your rental. On the Action Pane, click **Workflow**, and then click **Examination complete**. You may need to refresh the page to get the full set of options under Workflow.
10. Enter a comment, and then click **Examination complete**.
11. You might again need to wait up to two minutes for the batch framework to process the change. On the Action Pane, click **Refresh** periodically, until you see the **Status** field change. Notice that the vehicle now has a status of **Awaiting Service**.
12. Optionally, you can continue to repeat these workflow steps to take the vehicle through the service and cleaning phases. After cleaning is completed, the final status is **Done**.
13. Click **Workflow**, and then click **View history**. The **Workflow history** form provides information about the vehicle workflow.
14. Click **Tracking details** to see the activities.



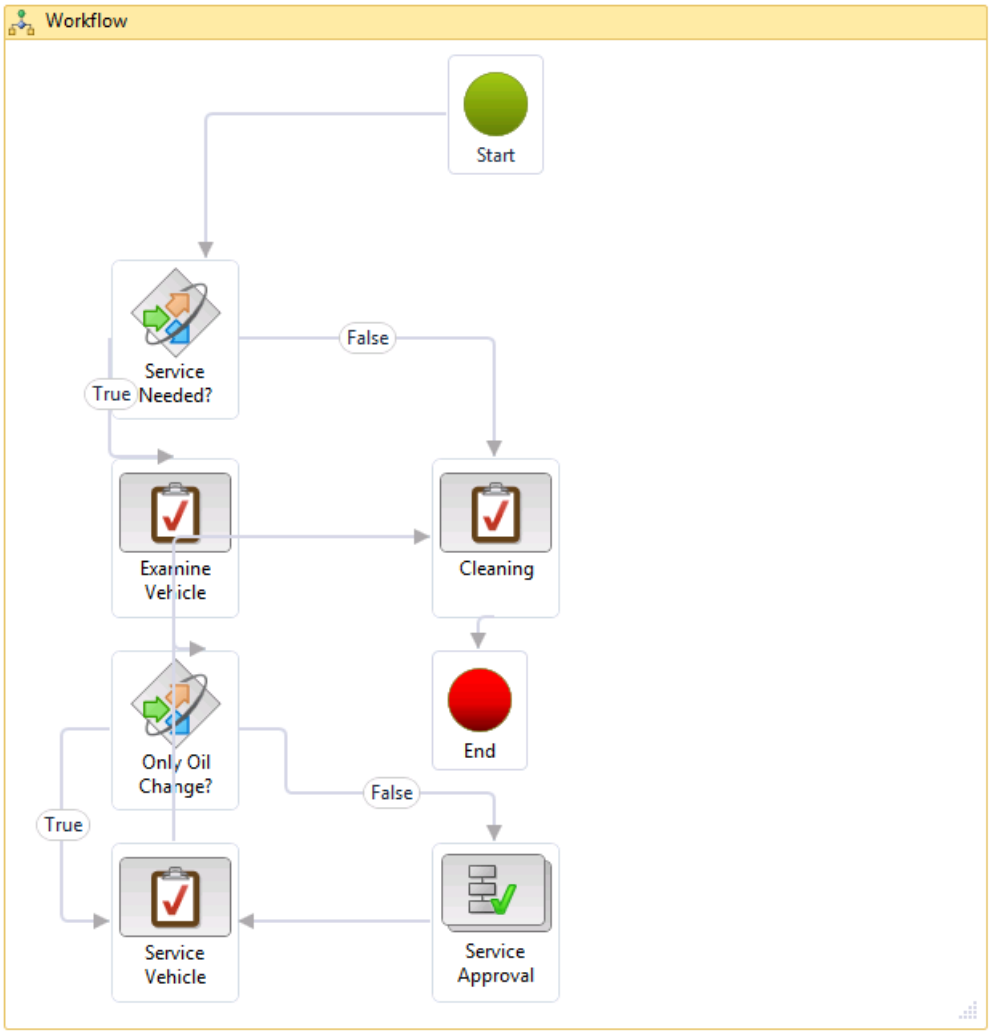
To view the setup behind the workflow

1. On the dashboard, navigate to **Fleet Management** > **Setup** > **Workflow setup**. The **Workflow Setup** page shows the list of workflows.



2. In the **Workflow ID** column, click the ID of your vehicle maintenance workflow.

3. Accept any prompts that ask you for permission to run code. After a short wait, the workflow editor opens. This step works on the one-box environment, but not in the cloud. You can view the workflow diagram in the workflow editor. The following illustration shows the workflow.



4. When you are done, close the **Workflow** window.

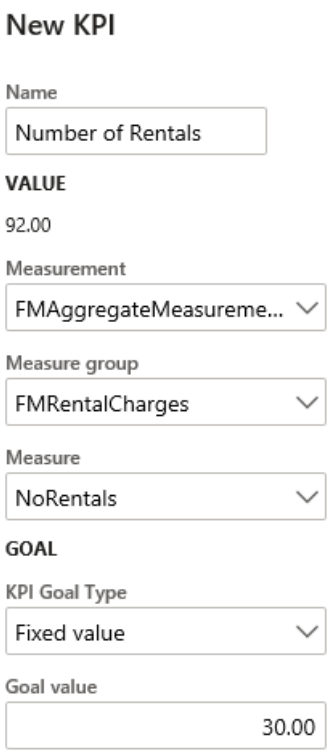
Create a new KPI definition

The web client enables users who have appropriate permissions to modify KPI definitions that have been modeled and deployed by developers. Users also have the ability to create new KPI definitions in the client. In this walkthrough, you create a new KPI definition in the client.

1. Open the **Reservation Management** workspace. On the navigation bar, go to **Fleet Management > Workspaces > Reservation Management**.
2. Notice the Total revenue KPI tile shown on the bottom left of the workspace. Click the **Total Revenue KPI** tile. Details of the total revenue KPI tile along with charts indicating top and bottom contributors to revenue will be shown on screen.
3. Next, you will define a new KPI to monitor the number of rentals.
4. On the Action Pane, click **New**. The **New KPI** dialog will open.
5. Enter following values for the new KPI definition.

FIELD	VALUE
Name	Number of Rentals
Measurement	FMAggregateMeasurements

FIELD	VALUE
Measure group	FmRentalCharges
Measure	NoRentals
KPI Goal Type	Fixed Value
Goal value	30



6. Click **Save**.

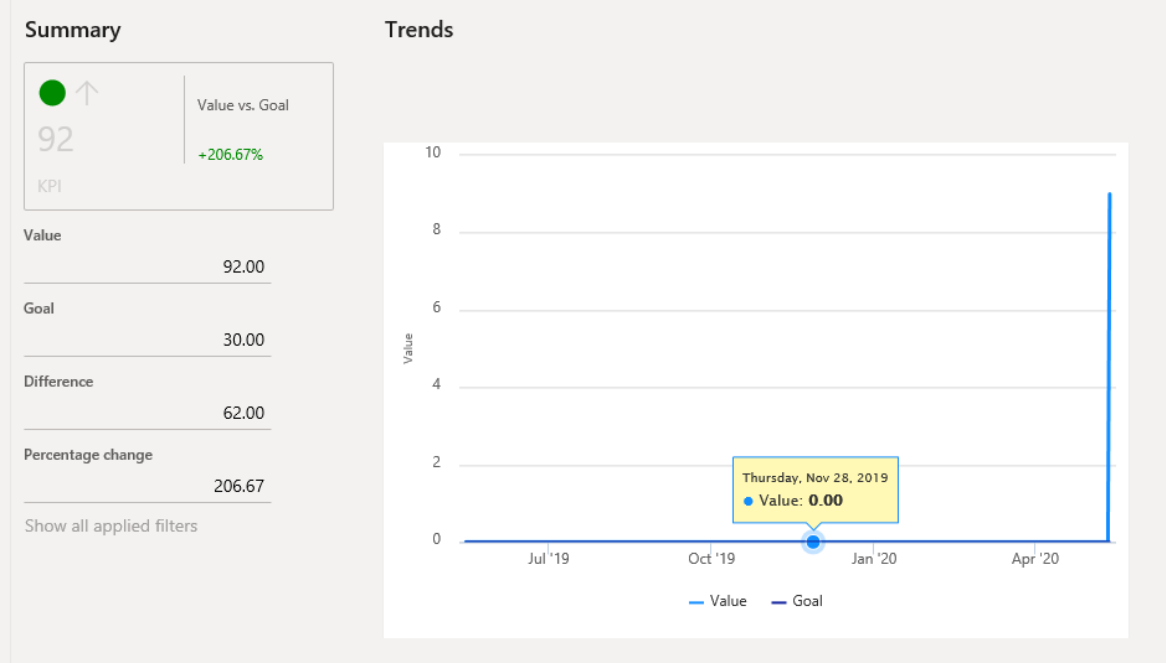
> **NOTE**
> If the **Save** button isn't visible in the **New KPI** dialog box, use a higher screen resolution so that you can see the entire dialog. You can see the KPI details page that contains details about the KPI that you created. You can make changes in the **Details** section. You will modify the default threshold values so that if the value is less than 90% of the goal, the KPI will show red and if the value is over 110% of the goal, the KPI will show green.

7. Click **Edit**.

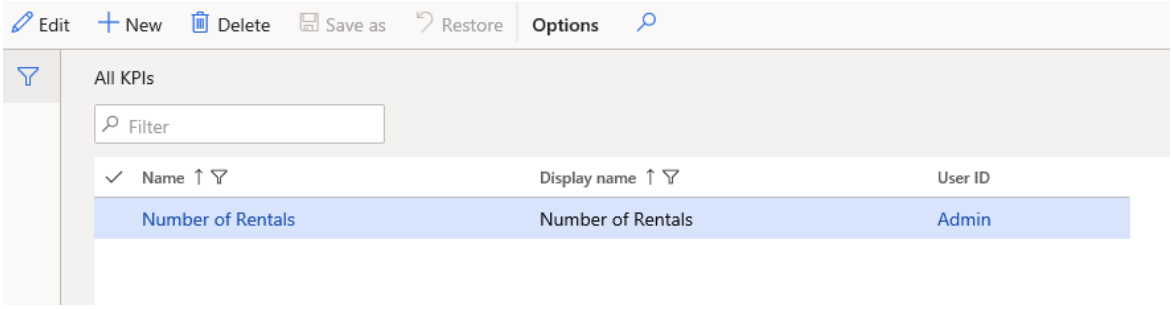
8. Scroll to the right of the screen, and modify the values in the thresholds fields as follows.

PROPERTY	VALUE
Bad if less than	90
Good if more than	110

9. In the application bar, click **Save**.



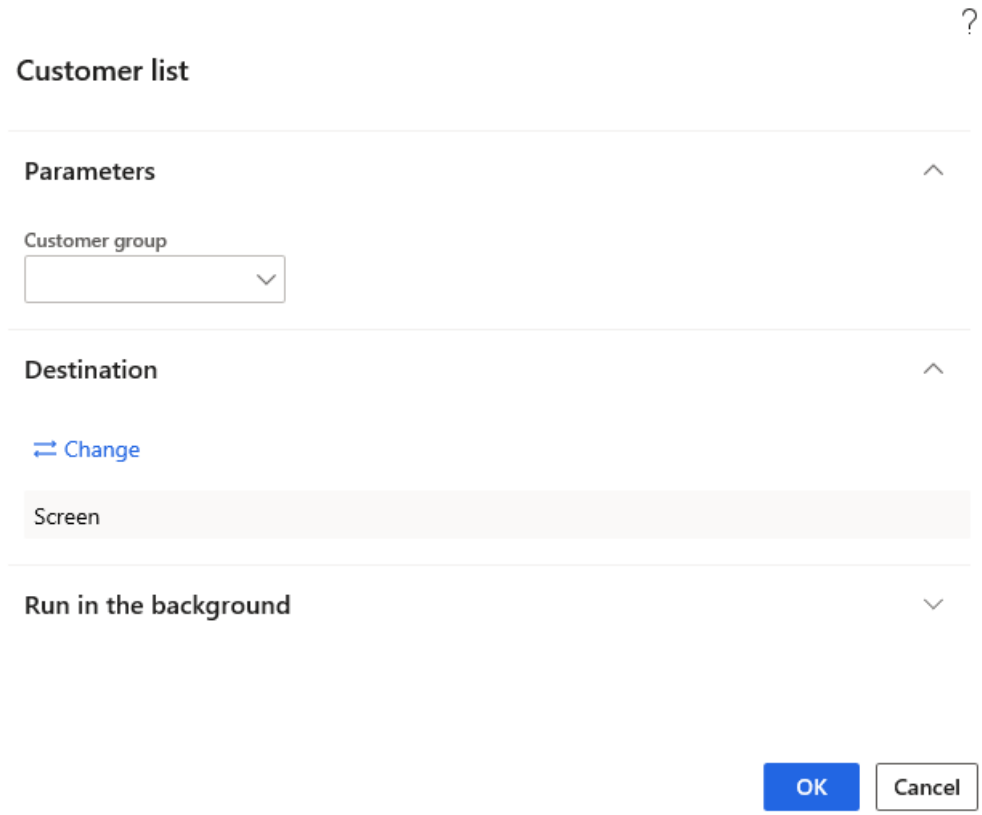
10. Click the form caption to return to the grid view.
11. Click the **Name** column header, change the filter operator to **contains**, and update the filter field value to **Number**. You will see the new KPI is available in the list.



Launch an operational report

In this tutorial, you'll launch an operational report that contains a list of customers who are currently renting vehicles.

1. Use the dashboard to open the **Reservation management** workspace.
2. On the right side of the page, under **Reports** click **Customer list**. Do not enter anything in the parameter for **Customer group**.



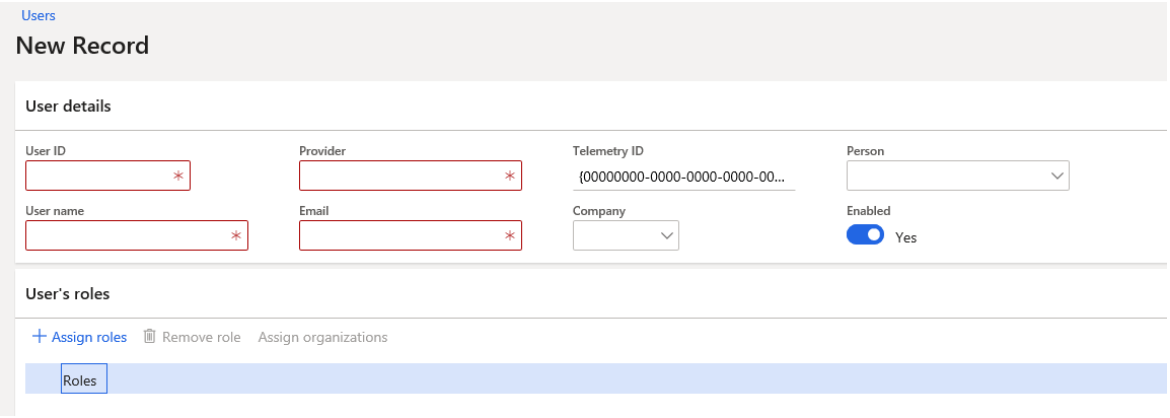
3. Click **OK** to close the dialog box. The report will be rendered and show the list of customers. The report may take a minute to render.

Secure access using the role-based security system

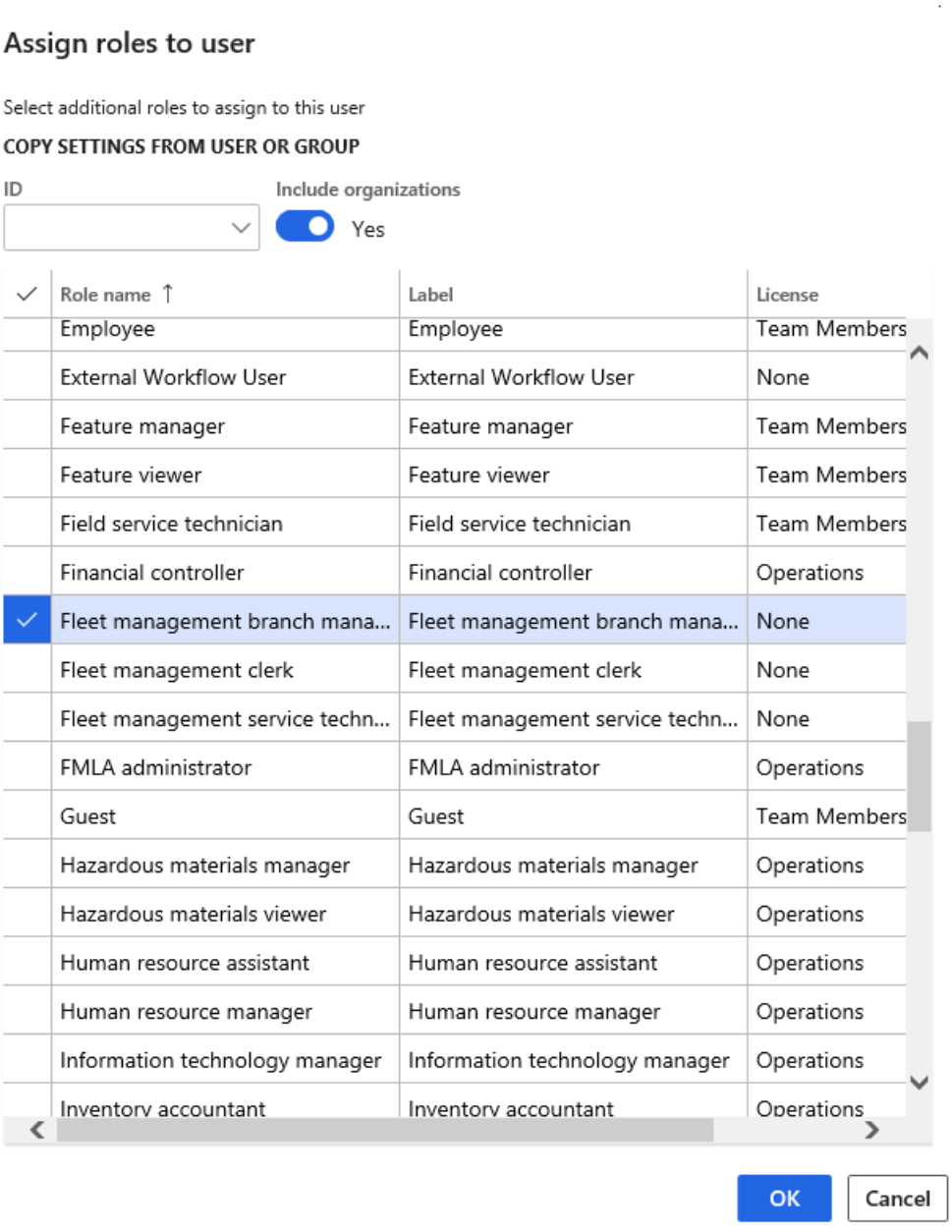
In this tutorial, you'll access the system as a user that has been assigned a different security role. This tutorial requires that you have created at least one additional end user.

1. On the dashboard, in the **System administration** section, click **Users** and then **Users**.
2. On the Action Pane, click **New**.
3. Enter the following field information.

PROPERTY	VALUE
User ID	Eight character unique ID
User name	The first name of the user
Provider	`urn:Federation:MicrosoftOnline`
Email	Use an email alias that you can login with for testing.
Company	`DAT`
Enabled	Verify that this slider is set to **Yes**.



4. Click **Assign Roles**.



5. Select **Fleet management branch manager**, and then click **OK**.
6. Click the user name on the top right, and then click **Sign Out**. You'll be redirected back to the sign-n page
7. Sign in using the credentials for the user who you assigned the security role to in the steps above.
8. Notice that in the dashboard, this user can see only items that are related to the branch manager security role. Items that system administrators can see are now hidden.

9. Click **Sign out** to sign out of the session.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Fleet Management sample application

2/18/2021 • 2 minutes to read • Edit Online

This topic is an overview of the Fleet Management sample application.

The Fleet Management sample application has been provided to showcase development and foundation capabilities. Fleet Management represents a solution that an ISV might create for a car-rental agency. Fleet Management data includes vehicles which are available for renting, and customers who can rent and return these vehicles. Employees can also run a maintenance workflow on these vehicles.

For some tutorials, you will need to create the FleetManagement solution if it is not on your computer. The steps to create it are listed in Tutorial: Create a Fleet Management solution file out of the Fleet Management models in the AOT.

For some tutorials, you must download the Fleet Management tutorial code and other artifacts from https://github.com/Microsoft/FMLab.

Fleet Management is provided as a Visual Studio solution that demonstrates platform capabilities, such as:

- Forms
- Workflow
- Security
- Labels
- Resources
- Data
- Business Intelligence
- Extensions

The Fleet Management solution includes two separate projects: one for the base model and the other one for extensions to the base model. The project named FleetManagement Migrated demonstrates how a migrated application might appear after migrating code from Dynamics AX 2012. This version shows how forms that have been migrated from Microsoft Dynamics AX 2012 R3 work on a web client. These forms have been created using automated migration tools and some other manual migration steps in Visual Studio. These forms bind to X++ tables and use the X++ programming model. The project named FleetManagement Discounts (or FleetManagementExtension) demonstrates how to use extensions to customize an application. This project extends the Fleet Management sample by extending controls and tables, handling data events, and replacing business logic using a plug-in. The tutorials that accompany this article provide a more-detailed look at the Fleet Management sample. These include a Fleet Management tutorial, End-to-end scenario for the Fleet Management sample application, and a tutorial that walks through extensions, Customize model elements through extension.

Additional resources

Using the Fleet Management sample

Customize model elements using extensions

Develop and customize home page

Download the FMLab sample code

Create the FleetManagement solution

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Development tools in Visual Studio

2/18/2021 • 2 minutes to read • Edit Online

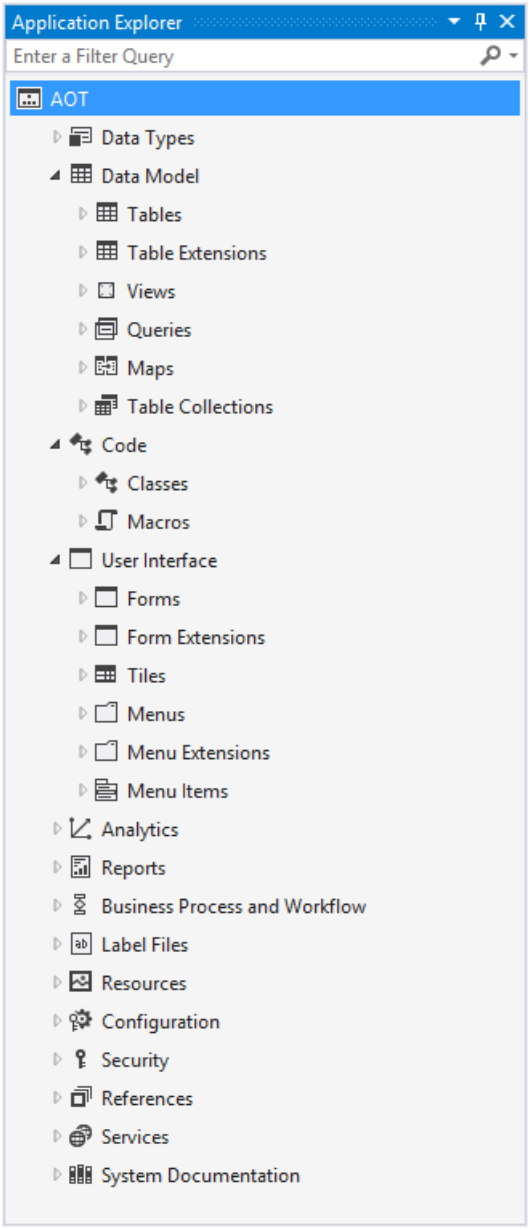
What are the development tools?

A notable change from Microsoft Dynamics AX 2012 is that the Finance and Operations applications do not include a rich-client application (ax32.exe). From a development perspective, this means that the Microsoft Dynamics AX 2012 development environment, MorphX, is no longer used. In its place, the application development is carried out exclusively in Visual Studio. The development tools support all of the development tasks, including debugging and local testing scenarios. A primary goal of the development experience is to keep familiar Microsoft Dynamics AX 2012 concepts, and seamlessly adapt them to the Visual Studio framework and paradigms.

Visual Studio is the exclusive integrated development environment (IDE) for development. All of your application development work will be performed with it. This section is an overview of the main features that are added to Visual Studio when the development tools are installed.

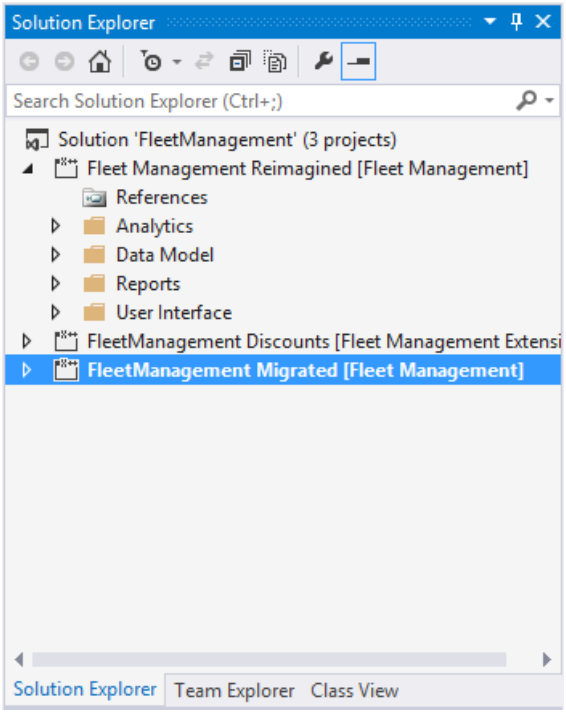
Application Explorer

In Visual Studio, the model store is represented by the Application Explorer. On the **View** menu, click **Application Explorer** to open it. The Application Explorer corresponds to the Application Object Tree (AOT) that you may be familiar with in Microsoft Dynamics AX 2012. Use the Application Explorer to browse and interact with the elements in the model store that define the applications. The following illustration shows the Application Explorer. For more details, see Application Explorer.



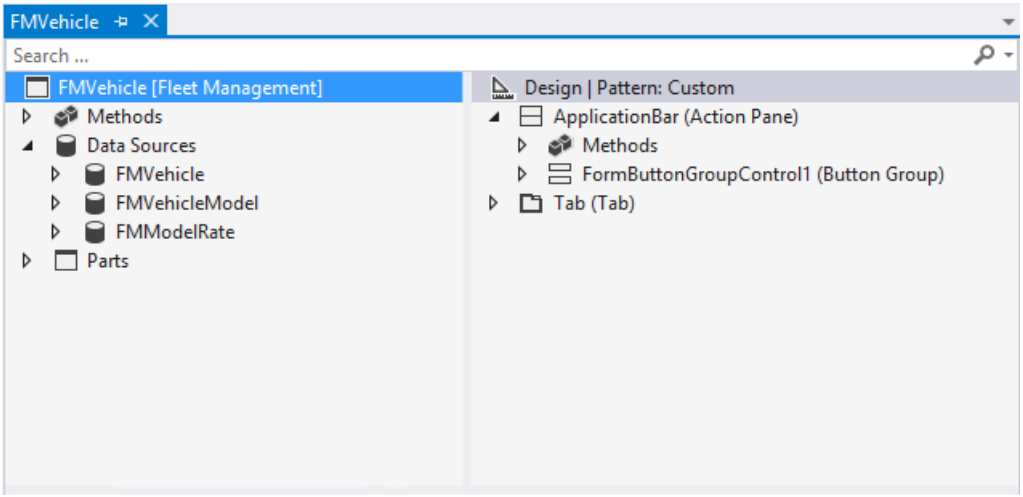
The project template

Even a simple application can have a large number of elements in its model. The **Operations Project** template has been added to Visual Studio to help you organize and manage the elements that you are working with for a model. You will use the project to design, build, and test model elements. It's common to have several projects within a single Visual Studio solution. The following illustration shows three projects in a Visual Studio solution. For more details, see Finance and Operations project type in Visual Studio.



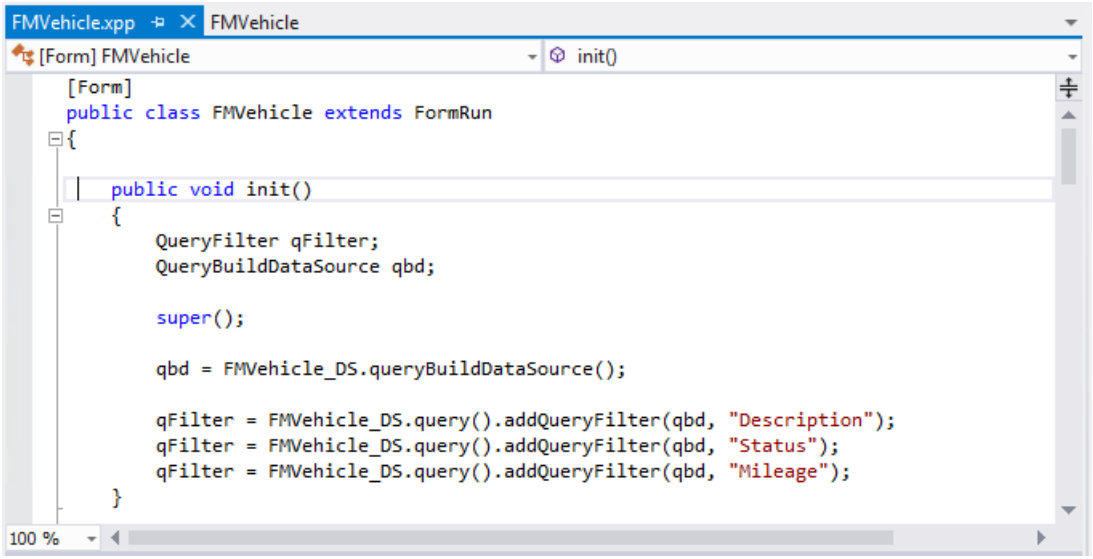
Element designers

The Visual Studio tools contain designers for each kind of element in the application. You will use these designers when you create or modify elements. The following illustration shows the element designer for a form element. For more details, see Element designers.



Code editor

The X++ code is written in the code editor for Visual Studio. The standard features that a developer expects from the code editor are supported. For example, sections of code are collapsible. IntelliSense provides guidance as you write or modify code. For more details, see Code editor features.



Dynamics 365 menu

The tools add the **Dynamics 365** menu to Visual Studio. Several tools that you will use during the development process are found here. For example, the tools for managing models are accessed from the menu.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Development tools tutorial

2/18/2021 • 7 minutes to read • Edit Online

This tutorial tours the Fleet Management solution in Visual Studio and introduces you to the development tools.

In this tutorial, you'll take a tour of the Fleet Management solution in Visual Studio. You'll see how a project is organized in the Visual Studio development environment. Much of what you'll see uses standard Visual Studio features, plus you will notice that we've added some new customized features. Along the way we'll point out some of these new and customized features and how they ease development. This tutorial will focus on:

- Visual Studio projects and their features.
- Files used in development.

Prerequisites

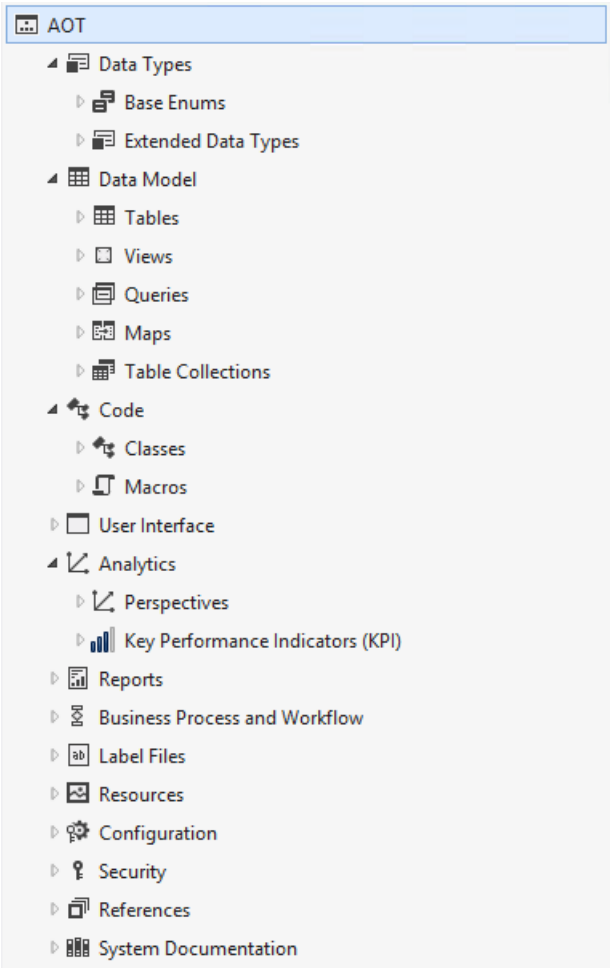
This tutorial requires you to access the environment using Remote Desktop and to be provisioned as an administrator for the instance.

Setup

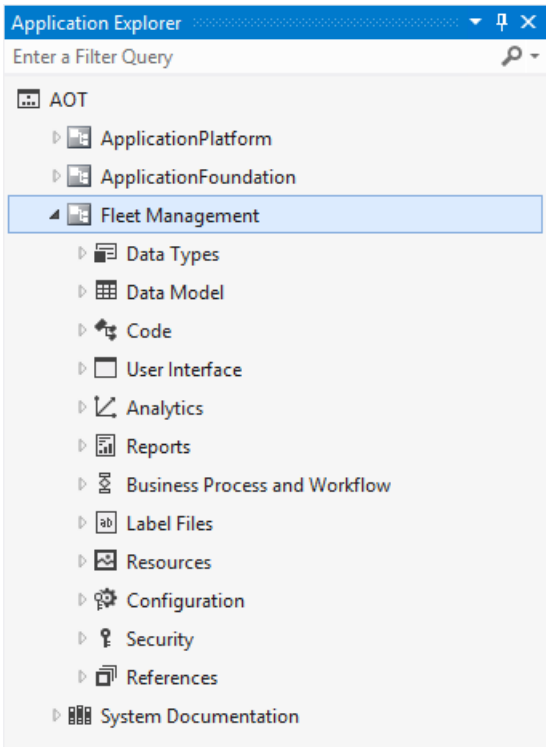
1. Start Visual Studio using **Run as an administrator**.
2. On the **File** menu, point to **Open** and then click **Project/Solution**.
3. Browse to the **Desktop,** and then open the **FleetManagement** folder. If the solution file is not on your computer, the steps to create it are listed in Tutorial: Create a Fleet Management solution file out of the Fleet Management models in the AOT.
4. Select the solution file named **FleetManagement**. The file type listed is Microsoft Visual Studio Solution (**SLN** file).
    - The fleet management solution file is available on the downloadable
    - VHD.
5. Click **Open**. The solution may take some time to load.

View the FleetManagement model

1. On the **View** menu, click **Application Explorer**. **Application Explorer** opens in Classic view. This view provides a familiar view of the **Application Object Tree** (**AOT**), which is similar to what you see in MorphX. The new AOT categorizes model element types a little differently than Microsoft Dynamics AX 2012. For example:
    - Items that were previously found in the **Data Dictionary** node are now under **Data Model** or **Data Types**.
    - Classes and macros are under **Code**.
    - Forms, menus, and other GUI elements are under **User Interface**.
    - Business intelligence components are under **Analytics**.



2. In **Application Explorer**, right-click **AOT**, and then click **Model view**.



   Model view organizes programmable objects according to their model. For these tutorials, the application suite models have been removed. The core foundation and platform components have been separated from the application suite. This separation is what allowed the application suite models to be removed.

3. Double-click **Fleet Management**, or click the arrow to expand the model's tree node. Model view provides a familiar way to work with a set of programmable objects. It's similar to what you saw in Classic view, but the tree displays only the objects that are part of that particular model.

4. Double-click **User Interface**, and then double-click **Forms** to expand the forms node. In the Fleet Management sample, the names of forms and other programmable objects are prefixed with "FM" so that they're easy to identify. The name of each object is followed by the name of its layer, which in this case is "isv," and then by the name of the model it belongs to, which is "Fleet Management."

5. You can continue to expand the nodes in the tree. For example, double-click **FM Rental** to view the parts, data sources, and methods for the form.

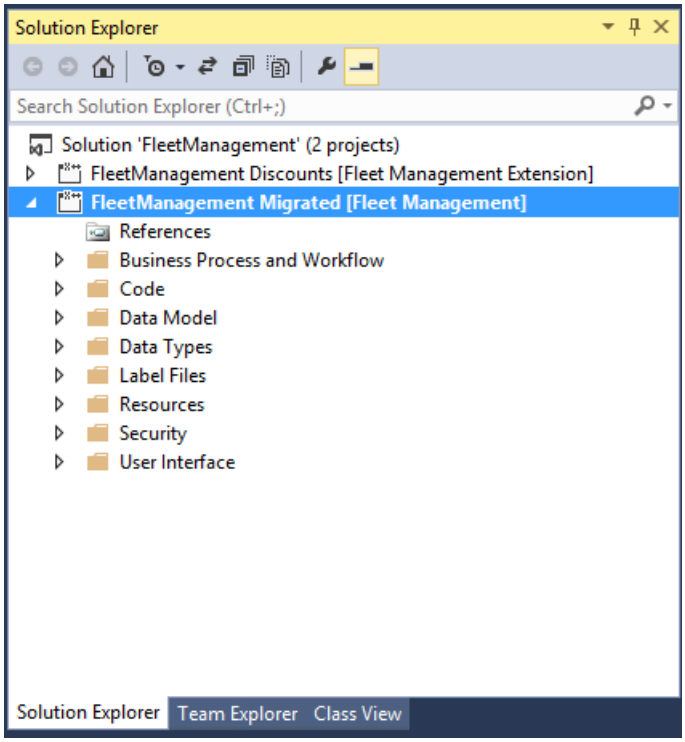
6. From **Application Explorer**>, you can open the Visual Studio code editor and view the source code for the form. For example, right-click **FMRental**, and then click **View code**.

7. You can also open the **Form Designer**. For example, right-click **FMRental**, and then click **Open designer**. You can expand the tree nodes in the **Form Designer** to see and edit the form metadata.

8. You can also preview the form in the **Preview** pane. Click on a control in the preview to it in the Form Designer. Similarly, click on a control in the Form Designer to highlight it in the form preview.

View the FleetManagement solution and its projects

This section of the tutorial describes the Fleet management projects and solution. Projects enable you to build your model elements (compile, synchronize the database, generate RDL files, etc.), test your application, and debug your code. We recommend that you don't make a change to a model element unless it's a part of a project; otherwise, your changes may not be compiled until you do a full build of your models, which can be a lengthy operation. In **Solution Explorer**, you can see the sample projects, two of which are named **FleetManagement Discounts** and **FleetManagement Migrated**. These projects are contained in a single Visual Studio solution, named **FleetManagement**.

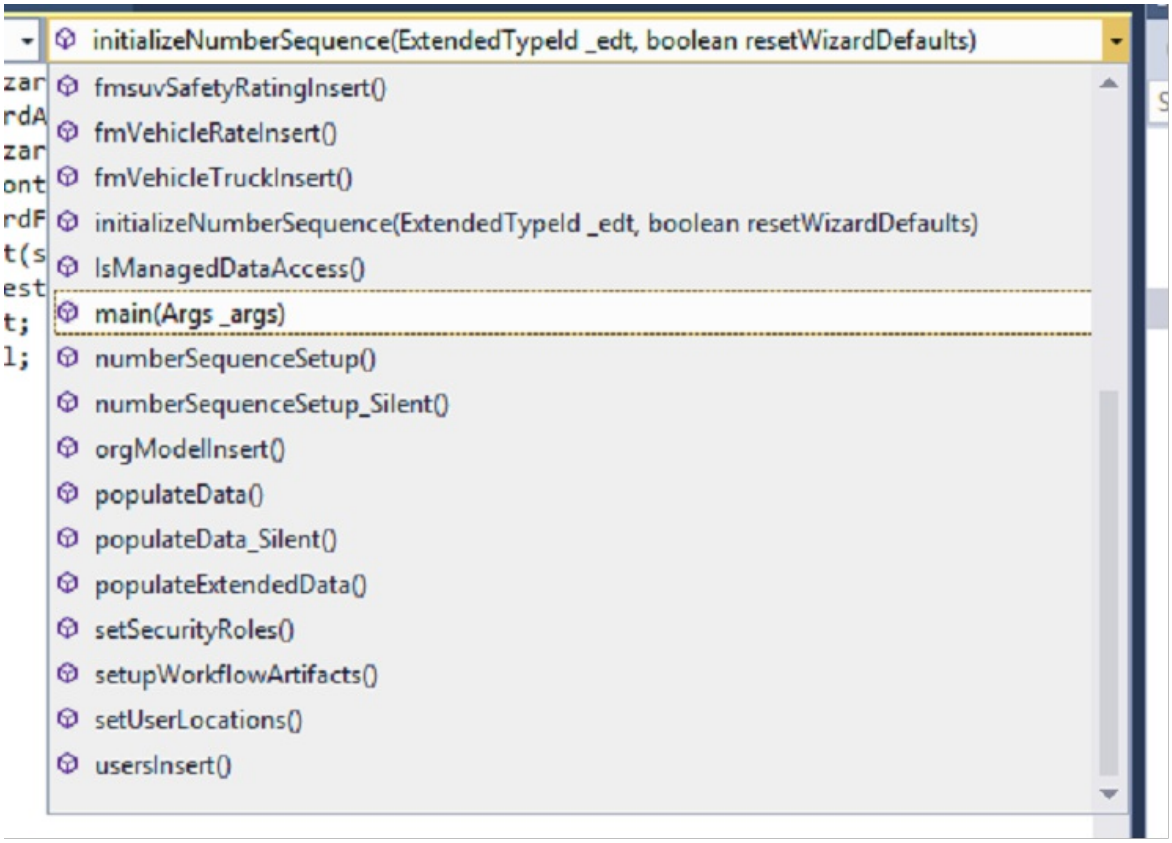


1. In **Solution Explorer**, right-click **Fleet Management Migrated**, and then click **Properties**.
2. In the **Property Pages** dialog box, review the listed properties. In the **Startup Object** field, you can see the name of the first form that runs when you run or debug your project. You can see that the **Startup Object type** field is set to **Form**. You can also view the model name and its layer. A project always belongs to one model.
3. Don't change anything in this dialog box right now. Click **Cancel** to close it.

View the source file for a model element

The code in a solution is stored as XML. The following instructions show you how to view the code in Visual Studio and the source XML in Internet Explorer.

1. In **Solution Explorer**, be sure that the **Fleet Management Migrated** project node is expanded.
2. Double-click **Code**, and then double-click **Classes**, to open the folder that contains the list of classes for the **Fleet Management Migrated** project.
3. In the list of classes, double-click **FMDataHelper** to open the code editor. Here, you can see the implementation of the **FMDataHelper** X++ class.
4. Scroll down in the code to locate the **main** method. **Tip**: You can also go the main method using the method navigation menu located on the top right of the code editor window.



If this class is set as the startup object of the project, the **main** method will be the execution entry point when you run or debug the project.

```
FMDataHelper.xpp
FmDataHelper                                    main(Args _args)
public class FmDataHelper
{
    #define.FMSvcTechUserId('FMSvcTec')
    #define.FMClerkUserId('FMClerk')
    #define.FMManagerUserId('FMMgr')
    #define.FMSvcTechUserGrpId('FMSvcTech')
    #define.FMClerkUserGrpId('FMClerk')
    #define.FMManagerUserGrpId('FMManager')

    /// <summary>
    /// Create a number sequence reference for the default scope.
    /// </summary>
    /// <param name = "_edt">The extended data type of the number seqeunce.</param>
    private static void initializeNumberSequence(ExtendedTypeId _edt, boolean resetWizardDefaults)
    {
        NumberSequenceReference sequenceReference;
        NumberSequenceTable sequence;
        NumberSeqDatatype dataTypeObject;
        NumberSequenceDatatype dataTypeRecord;
        NumberSeqScope scope;

        if (_edt)
        {
            sequenceReference = NumberSequenceTable::autoCreate(_edt);
            if (sequenceReference && resetWizardDefaults)
            {
                ttsbegin;
                sequence = NumberSequenceTable::find(sequenceReference.NumberSequenceId, true);
100 %
```

5. In Windows, open **File Explorer**, and then browse to the following folder: C:\Packages\FleetManagement\FleetManagement\AxClass

6. Double-click the file named FMDataHelper.xml. If you're prompted to choose a program to open the file with, click **More options**, and then click **Internet Explorer**. Otherwise, open it in Notepad.

```
C:\Packages\FleetManagement\FleetManagement\AxClass\FmDataHelper.xml
C:\Packages\FleetManage...

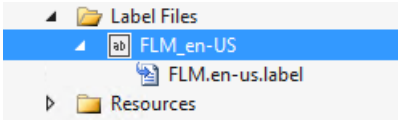
<?xml version="1.0" encoding="UTF-8"?>
- <AxClass xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
    <Name>FmDataHelper</Name>
  - <SourceCode>
    - <Declaration>
        <![CDATA[ public class FmDataHelper { #define.FMSvcTechUserId('FMSvcTec') #define.FMClerkUserId('FMClerk') #define.FMManagerUserId
        ('FMMgr') #define.FMSvcTechUserGrpId('FMSvcTech') #define.FMClerkUserGrpId('FMClerk') #define.FMManagerUserGrpId('FMManager') } ]]>
      </Declaration>
    - <Methods>
      - <Method>
          <Name>initializeNumberSequence</Name>
        - <Source>
            <![CDATA[ /// <summary> /// Create a number sequence reference for the default scope. /// </summary> /// <param name =
            "_edt">The extended data type of the number seqeunce.</param> private static void initializeNumberSequence(ExtendedTypeId _edt,
            boolean resetWizardDefaults) { NumberSequenceReference sequenceReference; NumberSequenceTable sequence;
            NumberSeqDatatype dataTypeObject; NumberSequenceDatatype dataTypeRecord; NumberSeqScope scope; if (_edt)
            { sequenceReference = NumberSequenceTable::autoCreate(_edt); if (sequenceReference && resetWizardDefaults) { ttsbegin;
            sequence = NumberSequenceTable::find(sequenceReference.NumberSequenceId, true); if (sequence) { dataTypeObject =
            NumberSeqDatatype::construct(); dataTypeObject.find(_edt); scope = NumberSeqScopeFactory::createDefaultScope
            (dataTypeObject); dataTypeRecord = NumberSequenceDatatype::find(dataTypeObject.parmRecId()); sequence.AllowChangeDown =
            dataTypeRecord.WizardAllowChangeDown; sequence.AllowChangeUp = dataTypeRecord.WizardAllowChangeUp;
            sequence.AnnotatedFormat = dataTypeRecord.wizardAnnotatedFormat(scope, dataTypeRecord.wizardHighest, true);
            sequence.Continuous = dataTypeRecord.WizardContinuous; sequence.FetchAheadQty = dataTypeRecord.WizardFetchAheadQty;
            sequence.Format = dataTypeRecord.wizardFormat(scope, dataTypeRecord.WizardHighest, true); sequence.Highest =
            dataTypeRecord.WizardHighest; sequence.Lowest = dataTypeRecord.WizardLowest; sequence.Manual =
            dataTypeRecord.WizardManual; sequence.NextRec = 1; sequence.update(); } ttscommit; } } } ]]>
          </Source>
        </Method>
      - <Method>
          <Name>main</Name>
        - <Source>
            <![CDATA[ public static void main(Args _args) { FmDataHelper::cleanupBrokenNumberSequences();
            FMDataHelper::ConfigureTotalsEngine(); FmDataHelper::populateData_Silent(); } ]]>
          </Source>
        </Method>
      - <Method>
          <Name>ConfigureTotalsEngine</Name>
        - <Source>
            <![CDATA[ public static void ConfigureTotalsEngine() { FMParameters FMParam = FMParameters::find(true); ttsbegin;
            FMParam.TotalsEngine = "Default"; FMParam.update(); ttscommit; } ]]>
          </Source>
        </Method>
```

In this file, you can see XML code that contains the metadata that describes the **FMDataHelper** class. For example, you can see that the class named **FMDataHelper** contains a set of methods. You can see the code that implements the **intializeNumberSequence** method for example, which is contained by an XML element. The <![CDATA]> tag ensures that the contained text isn't interpreted or changed by the XML parser. This metadata contains the source code that you viewed in the Visual Studio code window. When you develop a solution, you always work with code that's stored as XML. This means that the code files are stored on your computer, not in the database. There isn't an active connection to an application

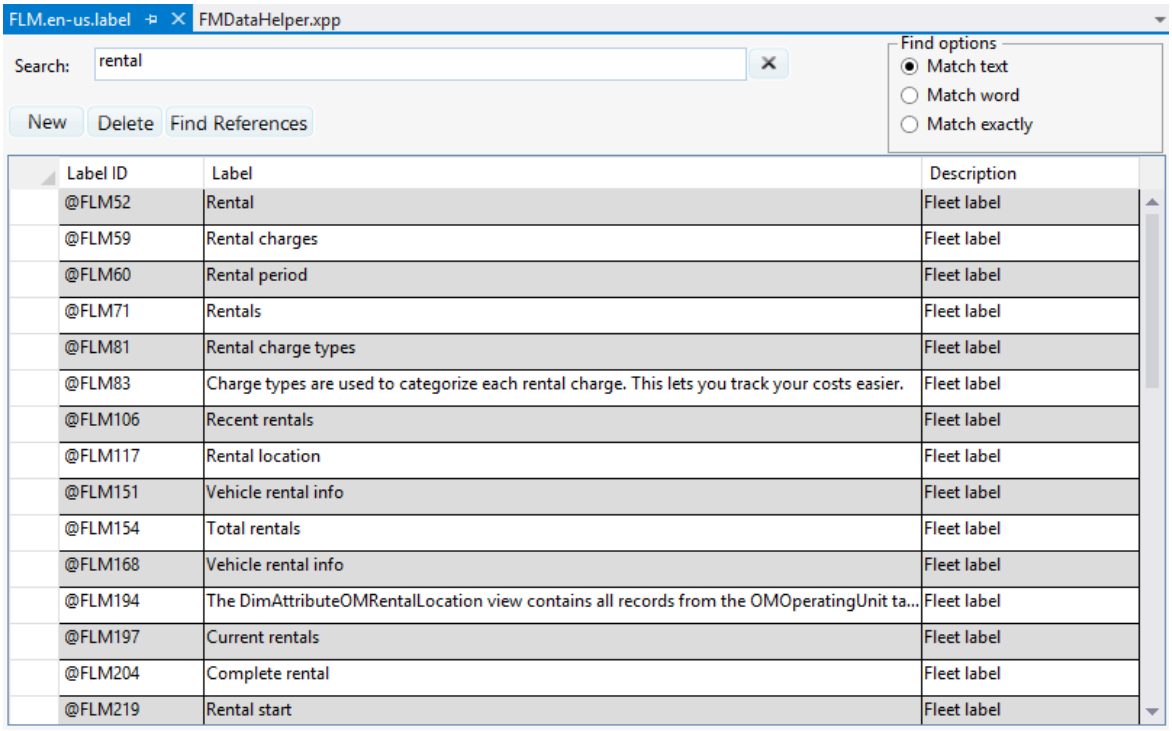
object server (AOS) while you develop your application. To avoid data loss, we recommend that you maintain your project files in a source code control system, such as Visual Studio Team Foundation Server. Although it's helpful to know how and where the source code files are stored, don't modify the XML files directly. **Always use Visual Studio to modify the source code for your projects.**

7. Close the window that **FMDataHelper.xml** file.

8. In **Solution Explorer**, double-click **Label Files** in the **FleetManagement Migrated** project to open the folder to view the labels for the project, and then double-click on the **FLM_en-US** node to open the label editor.



In the label editor **Search** box, enter "rental." As you type, you'll see the list of labels for the Fleet Management sample that contain the word "rental." You can double-click in any cell that can be edited to change its contents, and then save the label file.

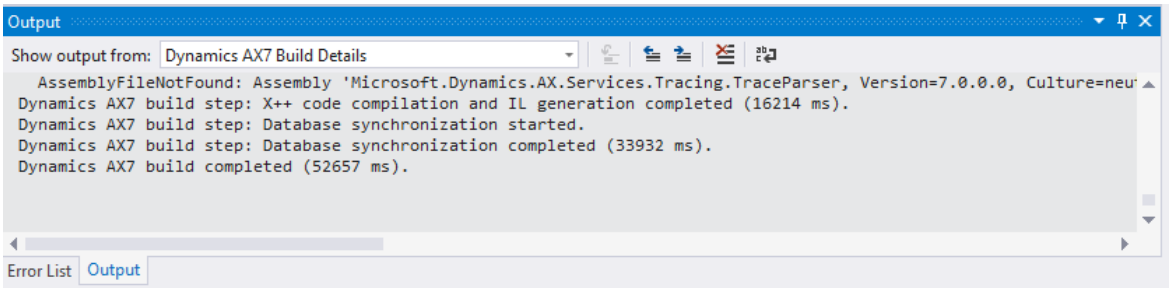


Label ID	Label	Description
@FLM52	Rental	Fleet label
@FLM59	Rental charges	Fleet label
@FLM60	Rental period	Fleet label
@FLM71	Rentals	Fleet label
@FLM81	Rental charge types	Fleet label
@FLM83	Charge types are used to categorize each rental charge. This lets you track your costs easier.	Fleet label
@FLM106	Recent rentals	Fleet label
@FLM117	Rental location	Fleet label
@FLM151	Vehicle rental info	Fleet label
@FLM154	Total rentals	Fleet label
@FLM168	Vehicle rental info	Fleet label
@FLM194	The DimAttributeOMRentalLocation view contains all records from the OMOperatingUnit ta...	Fleet label
@FLM197	Current rentals	Fleet label
@FLM204	Complete rental	Fleet label
@FLM219	Rental start	Fleet label

Build the FleetManagement migrated project

1. In the **Solution Explorer**, right-click **Fleet Management Migrated**, and then click **Rebuild**.

2. In the **Output** window, in the **Show output from** list, click **Build**. Verify that the build completed without compilation errors. Wait for the build to complete. The final build message in the **Output** window says, "... build completed." the final build message in the status bar (at the bottom left corner of Visual Studio) says "Ready."



3. On the **View** menu, click **Error List** to see the list of best-practice warnings. We've deliberately left some warnings in the build to demonstrate this feature.
4. Double-click any warning message to view the code or resource that caused the warning.
5. In the **Window** menu, click **Close All Documents** to close all open documents.

Application checker

2/18/2021 • 2 minutes to read • Edit Online

The application checker tool is a set of technologies that allow you to gain insight into your application code (both source and metadata) in ways that have not been possible before. The technology is based on representing both source code and metadata in XML and providing rich search facilities by using the XQuery language to express declarative queries over the source code. The current implementation runs inside a BaseX repository, which runs locally on the developer's box.

For information about how to install and use the application checker, see Dynamics365FO-AppChecker.

Application Explorer

2/18/2021 • 6 minutes to read • Edit Online

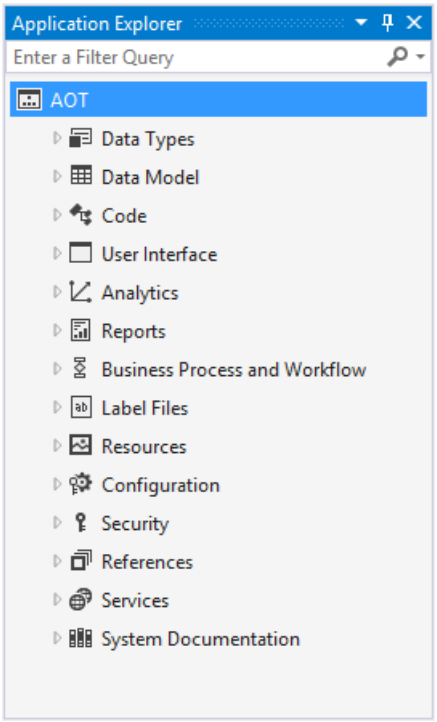
This topic reviews Application Explorer, and the various views and filtering methods in it. The topic also describes how to work with elements in Application Explorer.

Application Explorer

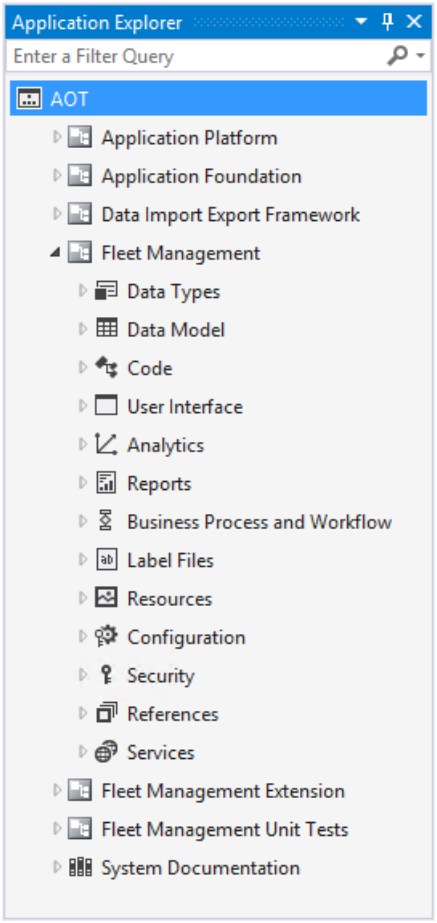
Application Explorer is the tool that you use to find the elements that you want to add to a project so that you can work with them. To access Application Explorer, on the **View** menu, click **Application Explorer**. An important difference between Application Explorer and the Application Object Tree (AOT) in the MorphX environment of Microsoft Dynamics AX 2012 is that you don't use Application Explorer to add or edit model elements. Instead, you use it to view elements, view code, find references to a selected element, and add elements to a project. To create, design, edit, and build model elements, you must use a Finance and Operations project.

Application Explorer views

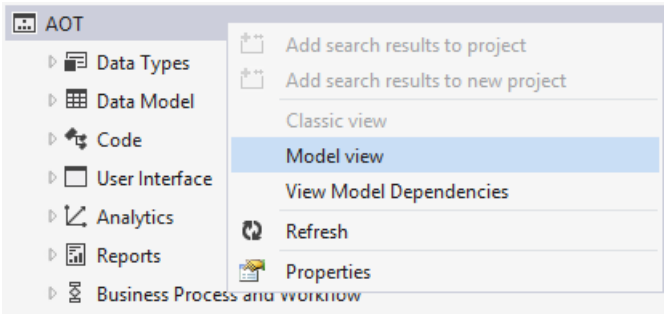
The content in Application Explorer can be organized in two ways. In the *classic view*, all the elements from every model are grouped according to type. This view resembles the way that the AOT was organized in Dynamics AX 2012. The following illustration shows the classic view.



The second view is called the *model view*. In this view, each model is listed separately. The elements within each model are grouped, as in the classic view. The following illustration shows the model view. Notice that the node for the Fleet Management model is expanded, and that the elements in the model are arranged as they would be in the classic view.



To switch to the model view, right-click the **AOT** node, and then click **Model view**. To switch back to the classic view, right-click the **AOT** node, and then click **Classic view**.

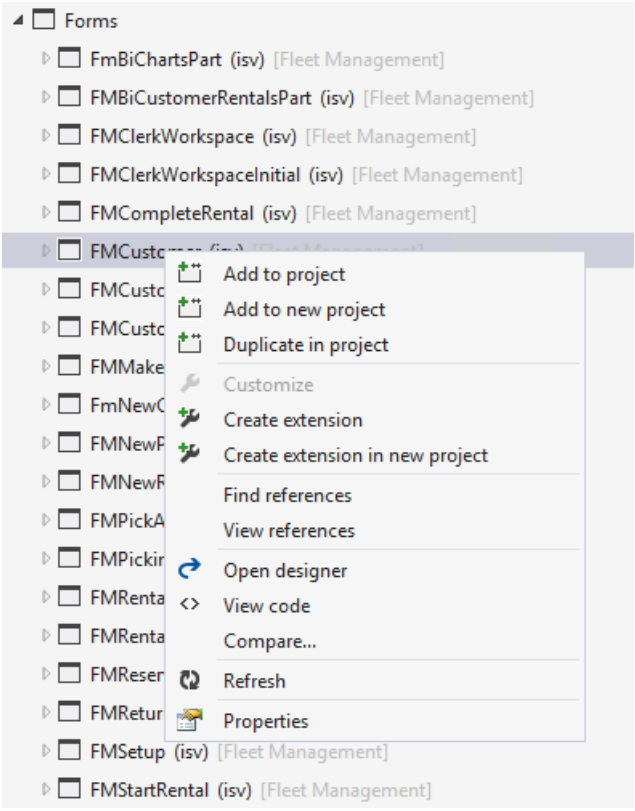


Working with elements

To work with elements, you must select them in the Application Explorer:

- To select one element, click it.
- To select a contiguous group of elements, hold down the Shift key while you click through the group of elements.
- To select noncontiguous elements, hold down the Ctrl key while you click the individual elements.

After you select the elements, right-click the selection to view the actions that you can perform. The following illustration shows the actions that are available for a form element.



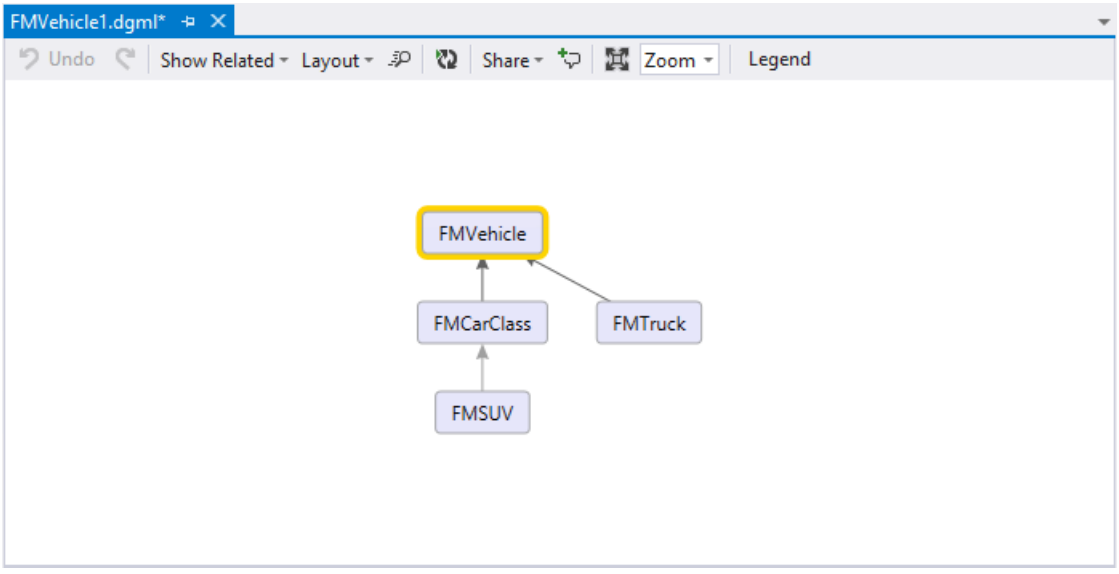
The actions that are available depend on the elements that you've selected. The following are some of the common actions that you can perform for elements in Application Explorer. **Note:** When the selected node is an element that exists in more than one model (in the case of overlayering customizations), the selected element will depend on the view that Application Explorer is currently in:

- If Application Explorer is in **model view**, the selected element is the element that belongs to the Application Explorer model that it appears under.
- If Application Explorer is in **classic view**, the selected element depends on the current context and the selected project in Solution Explorer. For example, if **Add to project** is the selected action, the selected element will be the element that belongs to the model of the current project, when applicable.

ACTION	DESCRIPTION
Add to project	Add the selected element or elements to the current project.
Add to new project	Add the selected element or elements to a new project.
Duplicate in project	Create a copy of the selected element or elements in the current project.
Customize	Create a customized version of the element. Click **Customize** when you want to do overlayering (customization) of an existing element. The model that you're using for your current project must be in the same package as the selected element, and it must belong to a higher layer than the element that you want to customize. When you click **Customize**, a new "customization" model element file is created and is added to your project.
Create extension	Create an extension for the element. A new extension model element (.Extension) is added to the current project in Solution Explorer. This is the preferred way to work with existing elements.

ACTION	DESCRIPTION
Create extension in new project	Create an extension for the element as part of a new project. You define the new project when the **New Project** dialog box opens.
Find References	Find all of the X++ code and other elements that reference the selected element.
View references	Create a diagram that shows the other elements that reference the selected element.
Open designer	Open the element designer for the element of that type, so that you can view the element. Although you can modify the element, this is typically done when the element is part of a project.
View code	Open the code editor, where you can view and edit the X++ code for the element.
Refresh	Update the metadata for the element.
Compare	Compare the element with the XML representation of the element that you select. Typically, you will compare an element with a different version of the element from a source code control repository.
Properties	Open the **Properties** dialog box in Visual Studio, so that you can see the property settings for the element.

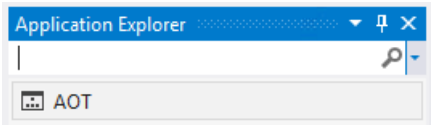
Some elements have unique commands that let you perform actions for that type of element. For example, table elements have two commands that provide additional information about the table. The first is the **View hierarchy** command. When you right-click a table element and click this command, you will see a graphical representation of the table hierarchy that the table is part of. For example, the following illustration shows the table hierarchy for the FMVehicle table.



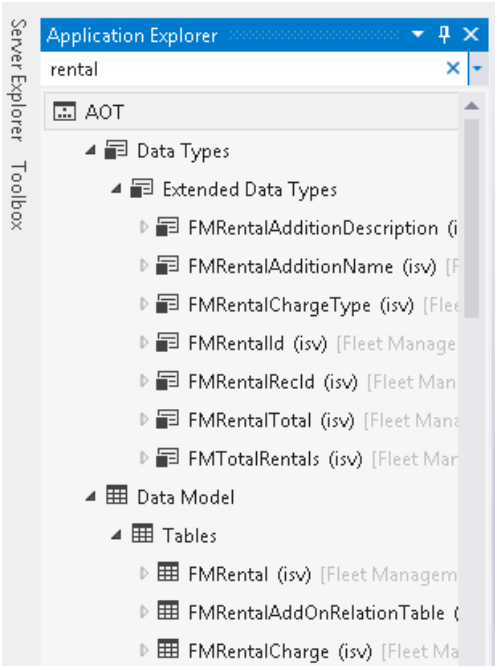
A similar hierarchy tool is available for classes. The second command is **Open table browser**. When you click this command, the data from the table is displayed as a list in the program.

Filtering Application Explorer

Application Explorer can contain a very large number of elements. This can make it difficult to find the specific element or elements that you want to work with. However, Application Explorer can filter the elements, based on a query that you supply. To filter the elements that are displayed, you can enter a query in the search bar at the top of Application Explorer.



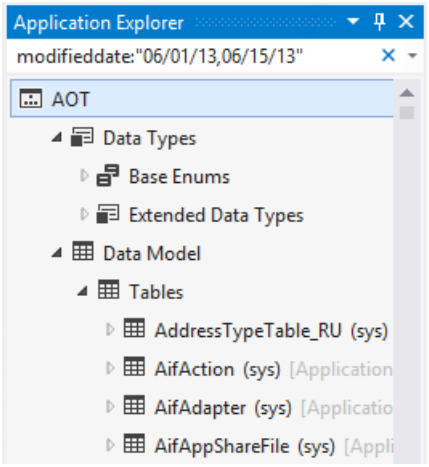
To apply a simple filter, just type the text that you want to filter by, and then click the **Filter** button at the end of the search bar. For example, if you want to find all of the elements that have a name that contains the word "rental," enter **rental** as the search term.



To clear the filter and return to the complete view, click the **Clear filter** button (X) in the search bar. Notice the drop-down arrow at the end of the search bar. If you click this arrow, you will see a list of filter options that you can use to refine the filter:

- Filter By Type
- Filter By Model
- Filter By Name
- Filter By Modified Date
- Filter By Extension Point

When you select one of these options, a predefined criterion is added to the search bar. You supply the specific values for the criteria. This feature can provide powerful search capabilities. For example, if you want to find the elements that were modified within a specific period, select **Filter By Modified Date**, and specify the start and end dates.



Previously used filters are listed in the drop-down list at the end of the search bar. You can also perform actions on the filtered results that are displayed in Application Explorer. Right-click the **AOT** node, and then select one of the following actions to perform on the results.

ACTION	DESCRIPTION
Add search results to project	Add the elements from the filter results to the current project.
Add search results to new project	Add the elements from the filter results to a new project.

These actions can be used only when the filter results are limited to a single model. Your query must contain the **model:** "*Model Name*" criterion to limit the filter to a specific model.

Additional resources

Develop and customize home page

Development tools in Visual Studio

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Build and debug projects

2/18/2021 • 11 minutes to read • Edit Online

In this tutorial, you'll learn about using the tools in Visual Studio to analyze and debug code in the Fleet Management application. You'll go through a simple developer scenario in which you will set breakpoints, modify some code, and build the result.

Prerequisites

Previous experience with code and Visual Studio is helpful to get the full benefit of this tutorial. This tutorial requires you to access the environment using Remote Desktop and that you be provisioned as an administrator on the instance.

Key concepts

- The debugger in Visual Studio is used to analyze and debug code for your projects.
- The standard features of the Visual Studio debugger are available to use when you're examining the running application. These features include modifying values of variables, setting breakpoints, and so on.
- IntelliSense and other features of Visual Studio are vital to efficient code editing and comprehension.
- Development is an iterative process. After making modifications to the code, the project will be built, and changes can be tested.

Scenario

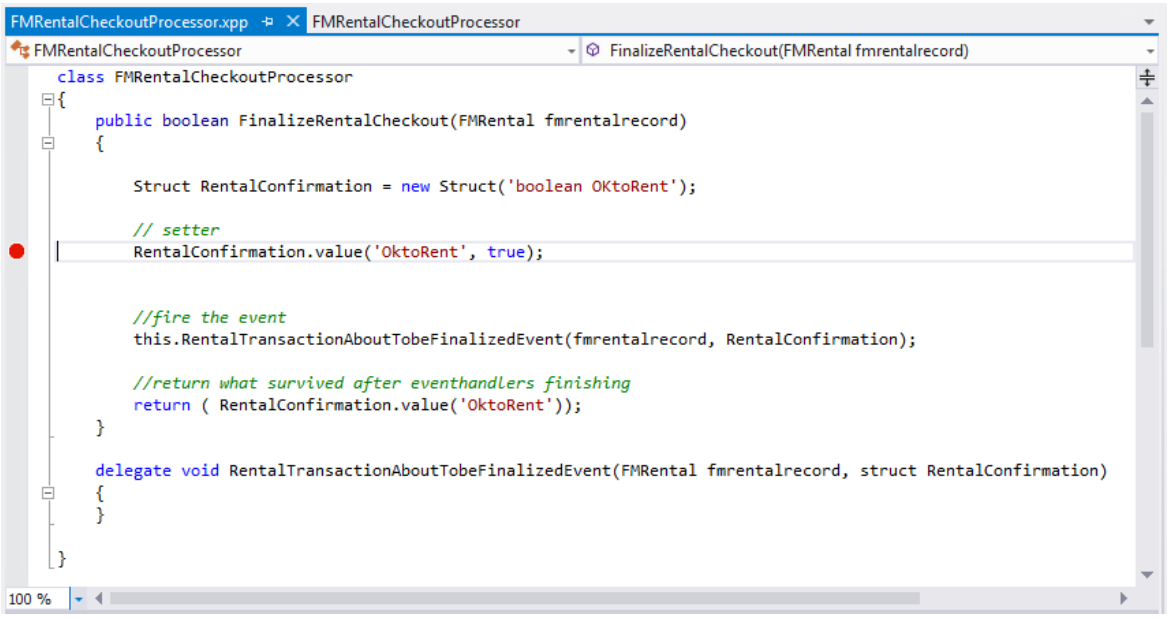
The rental company has had unfortunate events when customers rent cars using credit cards that are past the expiration dates. You, the developer, are tasked with revising the application to help prevent this situation. You'll identify the problem by using the debugger in Visual Studio. After the problem is identified, you'll edit some code to implement a fix. Finally, you'll build the project and validate that the fix was successful.

Run to a breakpoint

1. On the desktop, double-click the Visual Studio shortcut to open the development environment.
2. Open the FleetManagement solution. On the **File** menu, point to **Open**, and then click **Project/Solution**.
3. Browse to the desktop, and then open the **FleetManagement** folder. If the solution file is not on your computer, the steps to create it are listed in Tutorial: Create a Fleet Management solution file out of the Fleet Management models in the AOT.
4. Select the file named **FleetManagement**. The file type listed is SLN File.
5. Click **Open**. Loading the solution may take some time.
6. Make the **FleetManagement** project the startup project. In **Solution Explorer**, right-click the **Fleet Management** project, and choose **Set as StartUp Project** in the context menu.
7. In **Solution Explorer**, double-click the **Fleet Management** project to display its content.
8. Double-click the **Code** folder, and then double click the **Classes** folder of the Fleet Management project. Locate the **FMRentailCheckoutProcessor** class. Right-click this class, and then click **Open**. Alternatively, you can use the solution explorer search bar at the top of the solution explorer window. As you enter the name in the search bar, you'll see the corresponding artifacts selected in the solution

explorer. You can now see the X++ code for the class. This class has a method named **FinalizeRentalCheckout**.

9. Place a breakpoint in this method on the line following the first comment. To do this, click in the margin to the left of the line of code where you want the debugger to pause execution. You can also click anywhere in the line of code, and then press F9. The following illustration shows a breakpoint, which is displayed as a red-filled circle in the margin.



The FinalizeRentalCheckout method is called when a rental transaction is saved. This method calls the delegate named RentalTransactionAboutTobeFinalizedEvent. You can implement an event handler method, which is called by this delegate. The method that calls the delegate passes a parameter, named RentalConfirmation, which contains a value that indicates whether the rental should be allowed or blocked. If the rental is allowed, the value contains "true"; if it's blocked, the value contains "false". An event handler can change this value, based on any test the developer chooses to implement in code. In this case, we'll modify the code to test the expiration date of the credit card.

10. Press F5 to start the application for debugging, or, on the **Debug** menu, click **Start Debugging**. It's important that you start the application in one of these ways. If you don't, the Visual Studio debugger won't start, so you won't hit any of the breakpoints you've set. **Note**: The debugger needs to relate code position to source positions. It does this through consuming PDB files produced alongside the assemblies and net modules. The debugger will load symbols from the PDB files as described in the settings in the global tools settings. To open the options page containing the setting that controls which symbols load, go to the **Tools** menu and choose **Options**. In the **Microsoft Dynamics 365 for Finance and Operations** group, select the **Debugging** page. If this option is selected, the system will load symbols from only the PDB files related to the artifacts in the current solution. This reduces the startup time significantly, so be sure it's selected for this lab. Be aware that when this option is selected, it won't be possible to see source code from entities outside of the current solution. After a few moments, the browser will start and display the startup object that was selected in the project.

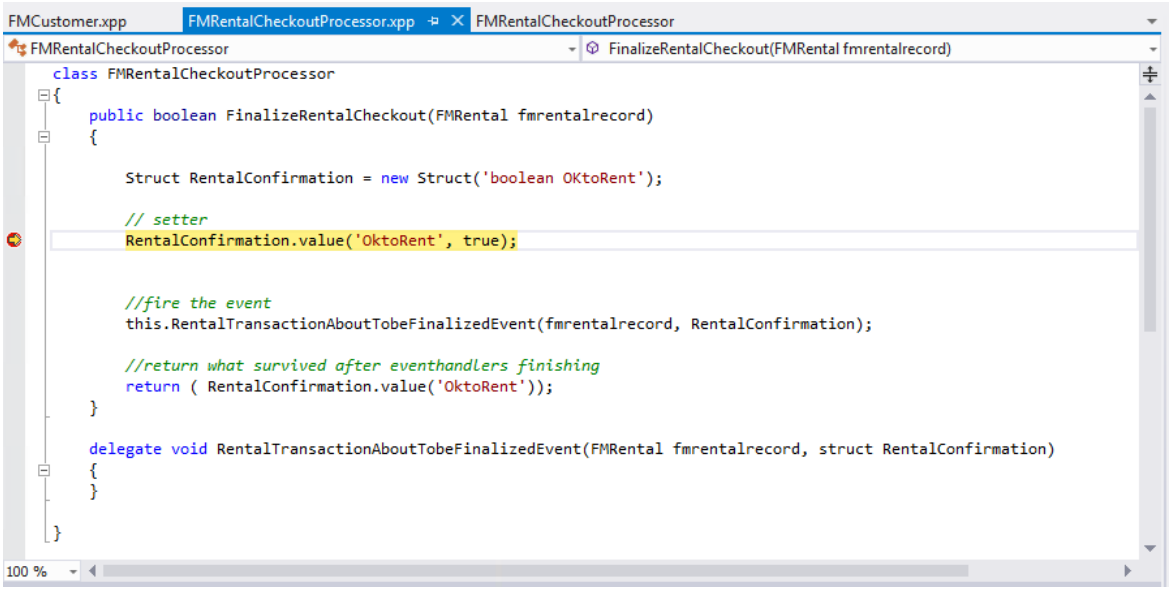
11. The **Current Rentals** page will open.

    a. In the Action Pane, click **Edit**.

    b. When the page is displayed, click **Show list** in the **Show/Hide** list (or press **Ctrl+F8**).

12. Make a change to any existing rental. For example, click **Edit**, and change the time that the rental period started.

13. Click **Save** to force a validation of the rental record. The method in which you placed a breakpoint is called. Execution pauses at the line of code that contains the breakpoint.



While the application is paused at a breakpoint, you can examine the application state. Use the same techniques that you typically would for any application developed with Visual Studio. For example, place the cursor over a variable or a parameter to see its value in a tooltip.

```
class FMRentalCheckoutProcessor
{
    public boolean FinalizeRentalCheckout(FMRental fmrentalrecord)
    {
        Struct RentalConfirmation = new Struct('boolean OKtoRent');

        // setter
        RentalConfirmation.value('OktoRent', true);


        //fire the event
        this.RentalTransactionAboutTobeFinalizedEvent(fmrentalrecord, RentalConfirmation);

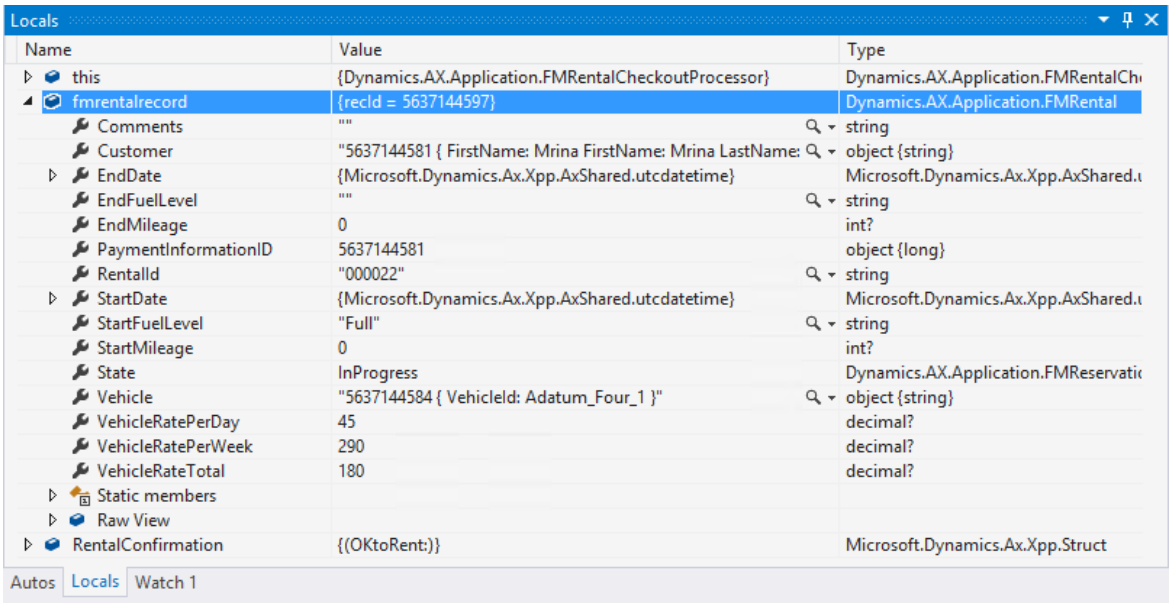
        //return what survived after eventhandlers finishing
        return ( RentalConfirmation.value('OktoRent'));
    }

    delegate void RentalTransactionAboutTobeFinalizedEvent(FMRental fmrentalrecord, struct RentalConfirmation)
    {
    }

}
```

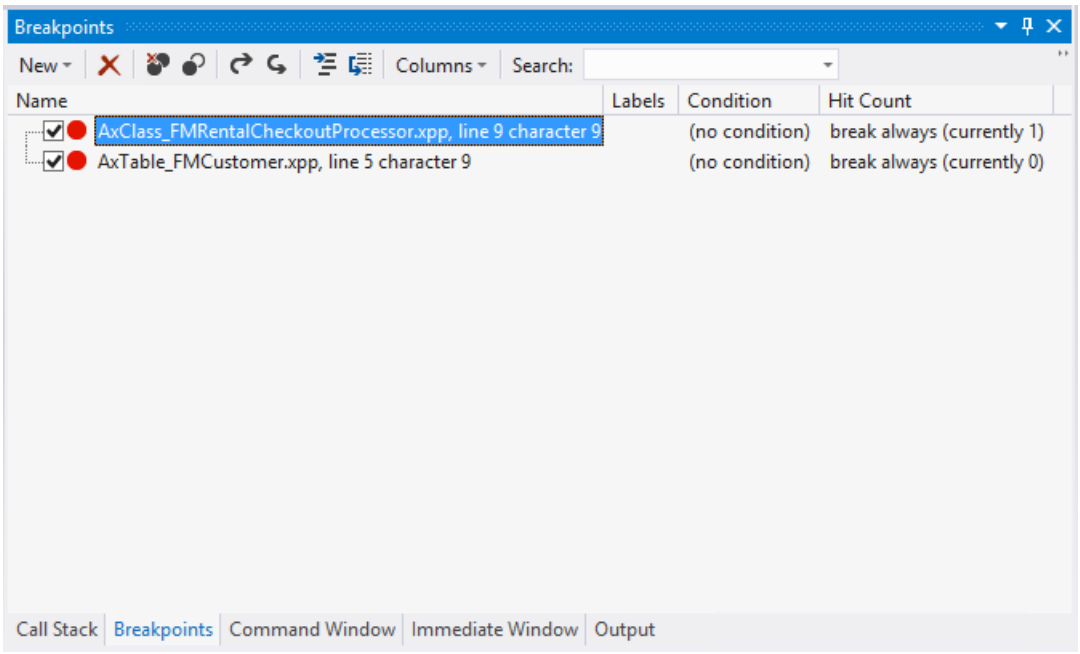
{recId = 5637144597}

14. The other debugging tools in Visual Studio are available as well. For example, the **Locals** window shows all of the local variables for the location where execution has stopped. Click the **Locals** tab at the bottom of Visual Studio, and expand the **fmrentalrecord** variable. You will see the internal state of the record, showing the values of all the fields in the record.



Notice the value of the **Vehicle** property of the **fmrentalrecord** variable. This property is a foreign key field in the **FMRental** table. The debugger allows us to peek into the related record in the FMVehicle table. It shows values that belong to the **AutoIdentification** field group.

15. The **Breakpoints** window lists all of the breakpoints that have been set. Click the **Breakpoints** tab to see its content.



16. Press F10 a few times to step through the code, line-by-line, and use the full complement of debugger features. Notice that the **Locals** window updates the values of variables immediately with each statement that's executed.

17. On the toolbar, click **Continue**, or press F5

18. Close Internet Explorer to close the **Fleet Management** application. Visual Studio will exit the debugging mode. An alternative is to choose **Stop Debugging** from the **Debug** menu. This will leave Internet Explorer open, allowing the next debugging session to start faster.

Add the validation code

In the **FinalizeRentalCheckout** method, you saw that the developer added code to call the delegate that's used to determine the validity of the rental. To solve the problem of expired credit cards, you'll add an event handler,

which you'll use to verify that the credit card isn't expired. To simplify the lab, the handler will be added in the same file that contains the delegate. Use the following code as inspiration. Rather than copying and pasting the code, type it in manually to see the IntelliSense features in action. These features add to the high level of productivity that Visual Studio users expect.

```
[SubscribesTo(classstr(FMRentalCheckoutProcessor),
    delegatestr(FMRentalCheckoutProcessor, RentalTransactionAboutTobeFinalizedEvent))]
public static void RentalFinalizedEventHandler(FMRental rentalrecord, Struct rentalConfirmation)
{
    FMPaymentInformation paymentInfo;
    date ccExpiryDate, lastDayOfExpiryMonth;
    str s;

    select firstonly * from paymentInfo where paymentinfo.RecId == rentalRecord.PaymentInformationId;

    if (paymentInfo)
    {
        // Check if the payment info is valid
        // For now, we will check if the credit card is expired
        // Credit cards expire on the last day of the month indicated
        ccExpiryDate = mkdate(1, str2int(paymentInfo.ExpirationMonth), paymentInfo.ExpirationYear);
        lastDayOfExpiryMonth = endmth(ccExpiryDate);

        if (lastDayOfExpiryMonth < today())
        {
            rentalConfirmation.value('OktoRent', false);
            s = "Credit card validation failed for rental ";
        }
        else
        {
            s = "Credit card validation succeeded for rental ";
        }

        info (s + rentalrecord.RentalId);
    }
    else
    {
        rentalConfirmation.value('OktoRent', false);
        info ("No Credit card available for " + rentalrecord.RentalId);
    }
}
```

The preceding code is straightforward. The method is marked as handler for the relevant delegate by using the **SubscribesTo** attribute, as shown. In the code, the customer record is retrieved, and then the credit card date is compared to today's date. If the expiration date of the credit card is in the past, the event handler sets a value in the **RentalConfirmation** structure to signal that the customer isn't eligible to rent a vehicle. The idea is that any number of handlers can subscribe to the delegate. If any handler determines that a rental should not proceed, it sets the **OkToRent** flag to false. A superior implementation might refrain from doing any analysis if it determines that the **OkToRent** flag has already been set to false.

1. Be sure that you're working in the FMRentalCheckoutProcessor.xpp file. Begin by adding the new event handler definition to the FMRentalCheckoutProcessor class. Add the following code on an empty line just above the brace (}) that marks the end of the class definition.

   ```
   public static void RentalFinalizedEventHandler(FMRental rentalrecord, Struct rentalConfirmation)
   {

   }
   ```

2. Add the attributes to the beginning of the event handler. These attributes indicate which delegate the event handler is subscribing to.

```
    [SubscribesTo(classstr(FMRentalCheckoutProcessor),
        delegatestr(FMRentalCheckoutProcessor, RentalTransactionAboutTobeFinalizedEvent))]
        public static void RentalFinalizedEventHandler(FMRental rentalrecord, Struct
                                                                        RentalConfirmation)
    {

    }
```

3. Now, add the code that checks the credit card expiration value. The completed method should look similar to the following code.

```
[SubscribesTo(classstr(FMRentalCheckoutProcessor),
    delegatestr(FMRentalCheckoutProcessor, RentalTransactionAboutTobeFinalizedEvent))]
public static void RentalFinalizedEventHandler(FMRental rentalrecord, Struct rentalConfirmation)
{
    FMPaymentInformation paymentInfo;
    date ccExpiryDate, lastDayOfExpiryMonth;
    str s;

    select firstonly * from PaymentInfo where paymentinfo.RecId == rentalRecord.PaymentInformationId;

    if (paymentInfo)
    {
        // Check if the payment info is valid
        // For now we will check if the credit card is expired
        // Credit cards expire on the last day of the month indicated
        ccExpiryDate = mkdate(1, str2int(paymentInfo.ExpirationMonth), paymentInfo.ExpirationYear);
        lastDayOfExpiryMonth = endmth(ccExpiryDate);

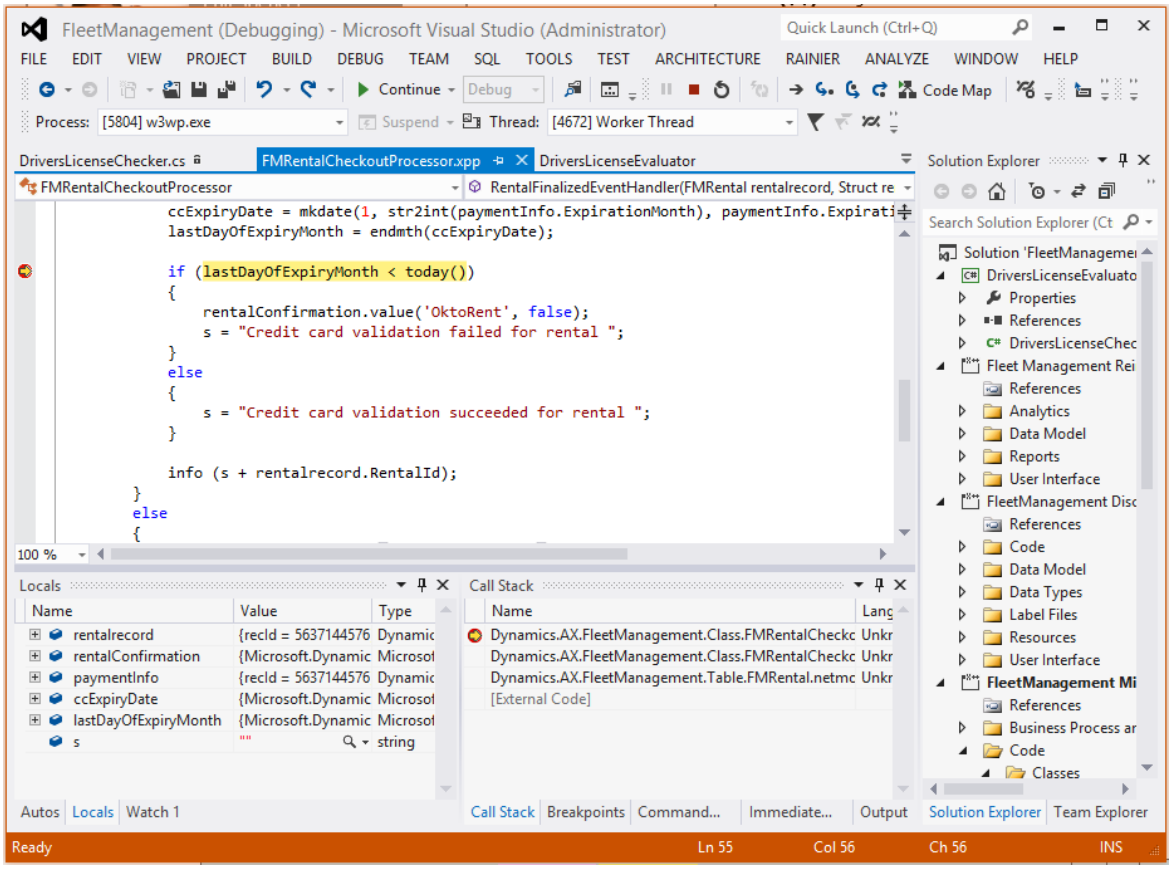
        if (lastDayOfExpiryMonth < today())
        {
            rentalConfirmation.value('OktoRent', false);
            s = "Credit card validation failed for rental ";
        }
        else
        {
            s = "Credit card validation succeeded for rental ";
        }

        info (s + rentalrecord.RentalId);
    }
    else
    {
        rentalConfirmation.value('OktoRent', false);
        info ("No Credit card available for " + rentalrecord.RentalId);
    }
}
```

4. Make sure the handler and the delegate are separated by exactly one blank line.
5. On the toolbar in Visual Studio, click **Save**.
6. After you're satisfied with the code, build the code for the Fleet Management project. To do this, in **Solution Explorer**, right-click the **FleetManagement** project name, and then click **Build**. You may see errors or warnings if the code isn't correct. If so, correct the code, and build again until all warnings and errors have been resolved. You're now ready to validate that the revision works as intended.
7. On the **Debug** menu, click **Delete All Breakpoints**.
8. Place a new breakpoint in the event-handler method at the line that contains the following statement:

```
if (lastDayOfExpiryMonth < today())
```

9. Start the Fleet Management sample with debugging active by pressing F5.
10. Browse to the **Current rentals** page, as described starting in step 11 of the previous section. Select one of the reservations, and click **Edit**.
11. In the **Customer** drop-down list, select Adrian Lannin from the list, and then click **Save**. Execution pauses at the breakpoint that you set in the event-handler method.
12. Press F10 three times to step through the code block.



13. To see the code map for the call stacks, right-click in the code editor, and then click **Show Call Stack on Code Map**. You may want to add a comment to the call stack, step through the code, and watch as the code map changes accordingly.
14. Press F5 to continue. You'll see that the customer has been disallowed.
15. In the same rental, change the customer name to Phil Spencer, and then click **Update**. This time, the transaction is allowed.
16. Close Internet Explorer.
17. Comment out the **SubscribesTo** attribute on the **RentalFinalizedEventHandler** method. This step ensures that the credit card test will no longer run as you work on the remaining tutorials.

Best practices

Earlier in this tutorial, you had the opportunity to add code to the project and build the solution with your changes. The build process may not have been successful, requiring you to refer to the error window. Using this window, you can get to the error by clicking on the line that describes the error. You may have noticed some diagnostic messages that don't represent compilation errors. These are diagnostics from the **Best Practice** checker. This tool will check for instances where the developer has violated a known best practice, and display a

warning when one is found. The Best Practice rules apply to both code constructs and metadata. Each software development organization is likely to have its own set of best practices that they enforce. They may want to disregard some of the existing best practice checks. To support this, the set of reported best practice diagnostics can be modified by the individual developer. To demonstrate this, complete the following steps:

1. On the **View** menu, click **Error List**. You should see a small number of best practice warnings.
2. On the **Dynamics 365** menu, click **Options**. In the **Dynamics 365** group, choose **Best Practices**.
3. In the **Model** drop-down list, make sure that the **Fleet Management** model is selected. The best practice rules apply to a particular model.
4. Mark the selections for some of the sets of Best Practice rules. For example, if you select the entry for **CodeStyleRules**, best practice guidelines for variables will be examined. After you've updated the selections, click **OK**.
5. Rebuild the Fleet Management project by right-clicking the project name and then clicking **Rebuild**. You'll notice that the violations of the best practice rules that you specified appear in the **Error list** window.

Build operations

2/18/2021 • 6 minutes to read • Edit Online

This topic reviews the process to build projects and full build of model packages.

The elements of a model must be built so that they can be used by the application. You can build the elements in a project. You can also build all the elements in a model. The following actions are performed during a build operation:

- Metadata validation
- X++ code validation
- Best practice checks
- Report RDL generation
- Compilation, IL generation, and creation of the .NET assemblies
- Label assembly generation and deployment of other resource files
- Database synchronization

Build a project

When you build a project, only those elements that are new or that have changed are built. To build a project, follow these steps.

1. In Solution Explorer, select the project.
2. On the **Build** menu, click **Build <project name>** to start the build process. Alternatively, right-click the project in Solution Explorer, and then click **Build**.

During the build process, you might notice that some elements that are built aren't part of the project. This behavior is required because of the way that assemblies are created. When you build an element, you're actually building the .NET module that the element is included in. A single .NET module contains multiple model elements, and a single assembly contains multiple .NET modules. The assembly can be created only if all the .NET modules in the assembly have been built and are up to date. If any elements in any of the .NET modules for an assembly haven't been built or aren't up to date, they will be built, even if they aren't included in the current project.

> **NOTE**
>
> If you delete an element from a project, you must rebuild the project or perform a full build on the model before the deletion takes effect.

Rebuild a project

If you want to build all the elements in a project, regardless of whether they have changed, you must perform a rebuild operation. To rebuild a project, follow these steps.

1. In Solution Explorer, select the project.
2. On the **Build** menu, click **Rebuild <project name>** to start the rebuild process. Alternatively, right-click the project in Solution Explorer, and then click **Rebuild**.

Synchronizing the database at each build

A project property lets you specify that the synchronize operation for the database should be performed every time that you build the project. This can be useful when you're making changes to the table structure for an application. Each time that you build, you will know that the database is synchronized with the tables as they are defined in the project. For information about how to set project properties, see Finance and Operations project type in Visual Studio. If your application has a large number of tables, and you aren't yet testing the application, you can set the **Synchronize database on build** property to **false**. This change will reduce the time that is required to build the project. Then, when you begin testing, be sure to set this property back to **true**. If you must manually synchronize the tables in a project, you can right-click the project in Solution Explorer and then click **Synchronize <project name> with database**. To synchronize the entire database, which can be a long process, on the **Dynamics 365** menu, click **Synchronize database**.

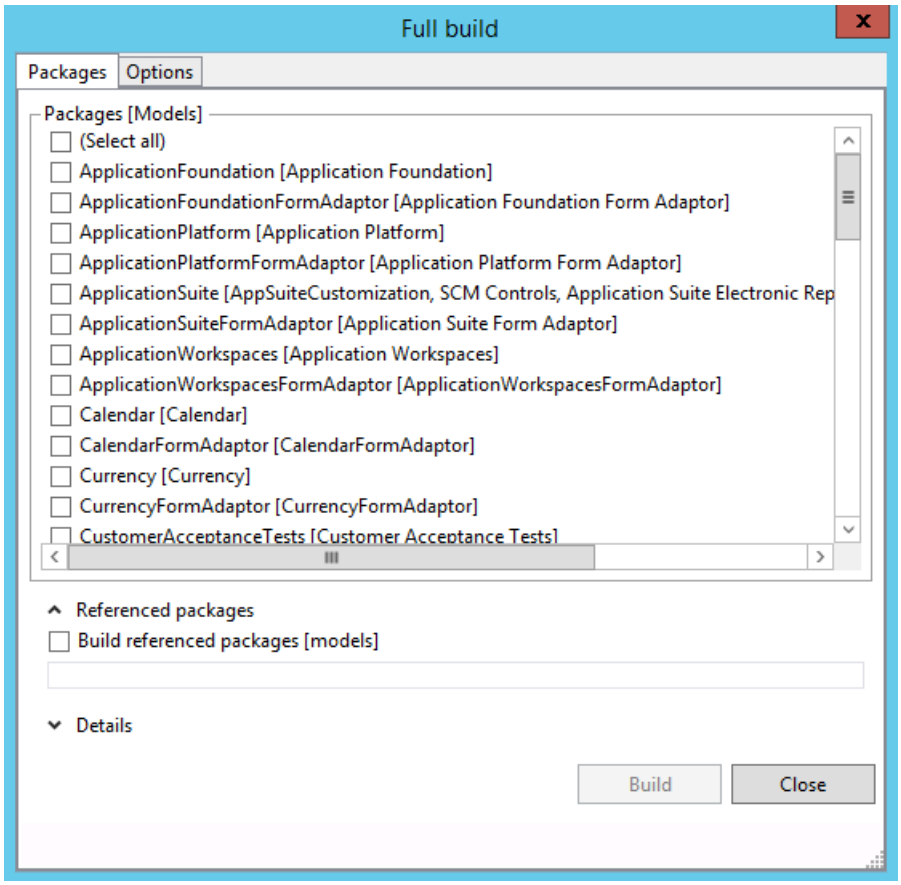
> **NOTE**
> If you try to synchronize the database before you have fully compiled assemblies, the Visual Studio database synchronization tool will display a message that synchronization has completed successfully, when in fact, the synchronization was not successful.

Tables and views cannot be synchronized against the database until they are fully compiled. After you complete a full build of the Application Platform, Application Foundation, and Application Suite, you can complete a Database Synchronization from the Dynamics 365 menu in Visual Studio.

Build a model's package

You might want to build all the elements in a specific model. To do this, you must perform a full build on the package that the model belongs to. Follow these steps.

1. On the **Dynamics 365** menu, click **Build models**.
2. In the **Packages** list, select the package(s) to build.
   - Package names are listed alphabetically.
   - Models belonging to the package are shown in brackets.
3. If you want to build the dependent packages first, select **Build referenced packages**. Any dependent package that must be built will be listed.



4. On the **Options** tab, review the options for the build process. The following options are available.

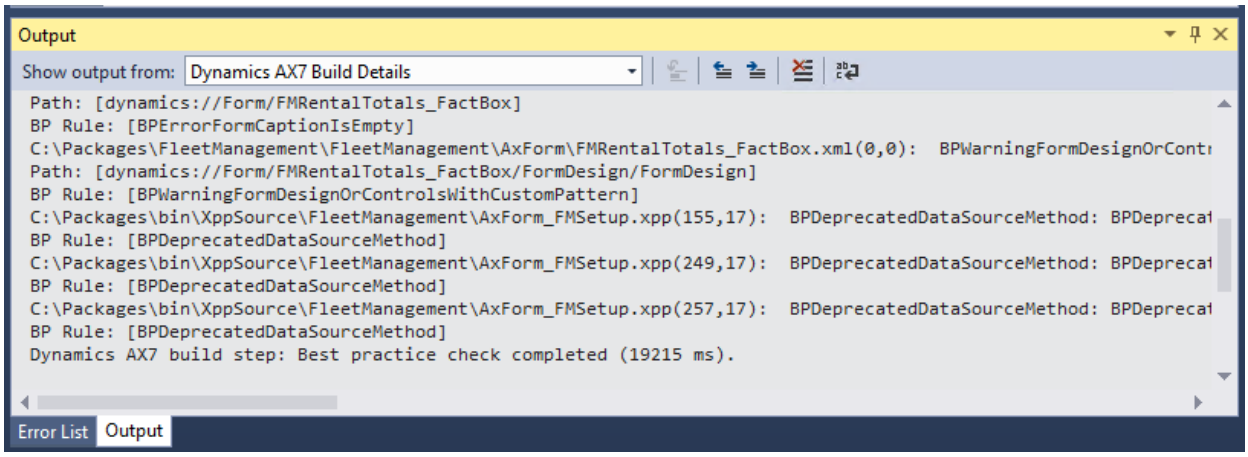
OPTION	DESCRIPTION
Build Pre-Compiled Forms	Static HTML is generated for each form during the build process. This allows faster rendering of forms at run time.
Build Reports	Reports are built.
Build Aggregate Measurements	Aggregate measurements are build.
Run Best Practice Checks	Best practice checks are performed during the build process.
Synchronize Database	The schema of the SQL database is updated during the build process (after compilation of the metadata and source code), so that it matches the metadata.
Build cross reference data	The data for the cross-reference feature is updated during the build process. Cross reference data enables developers to find references to code and metadata during development.

5. Click **Build** to start the build process.

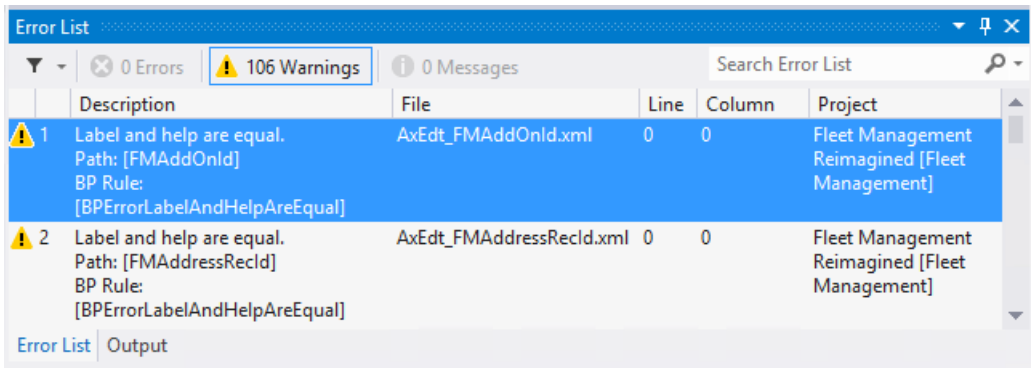
6. Expand the **Details** tab to follow details of the build process.

Build results

After a build operation is completed, you will see the results in Microsoft Visual Studio. The **Output** pane in Visual Studio shows the status of the build. You can use the **Show output from** field to switch between the standard build information and the build details.



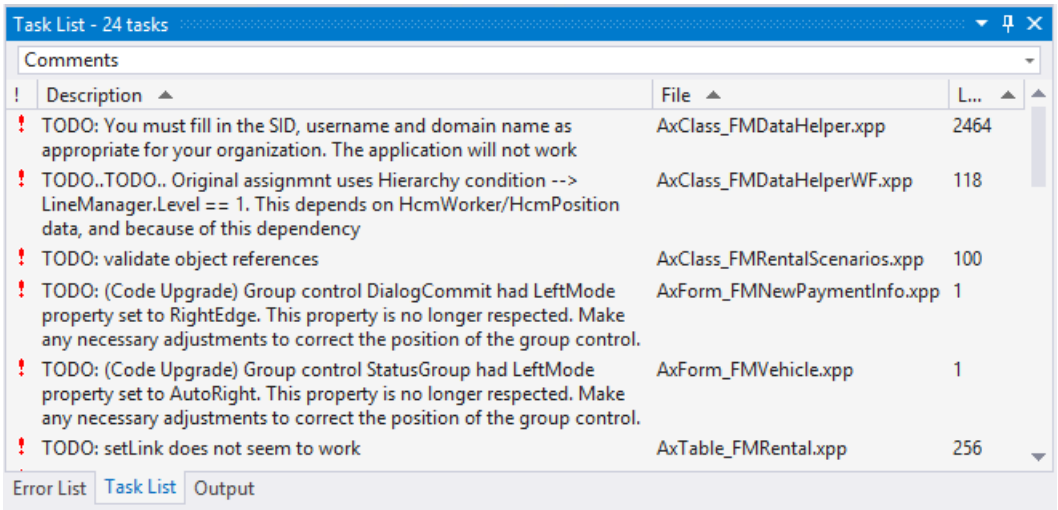
The **Error List** pane in Visual Studio shows the build errors and warning that occurred during the build process. If you see any build errors, you must fix them and then build again, so that valid assemblies can be created for the application. Many of the warnings that appear in the **Error List** pane are best practice checks that inform you of revisions that you should make to your application so that it conforms to the best practices for application development. Ideally, you should address all the best practice warnings for an application.



You can double-click most errors and warnings to see the source of the issue. The element designer or code editor will open, where you can see what property setting or code is causing the error or warning. The **Task List** pane in Visual Studio shows tasks that have been flagged with "TODO" comments in code. For example, the following comment indicates that some object references still require validation.

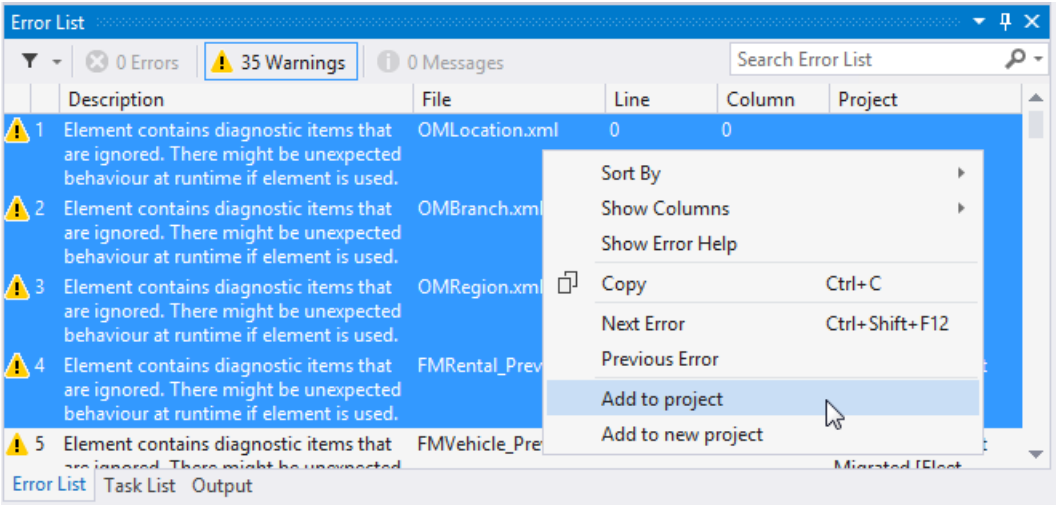
```
// TODO: validate object references
```

When the code is built, these "TODO" comments appear in the **Task List** pane. To view the **Task List** pane, on the **View** menu, click **Task List**.



To make resolution easier, you can add the elements that are affected by the error or task to the current project

or to a new project. In the **Error List** pane or the **Task List** pane, select the rows for the errors or tasks that you want to fix, right-click, and then click **Add to project** or **Add to new project**. This saves you the effort of finding the affected elements in the application.



Additional resources

Development tools in Visual Studio

Develop and customize home page

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

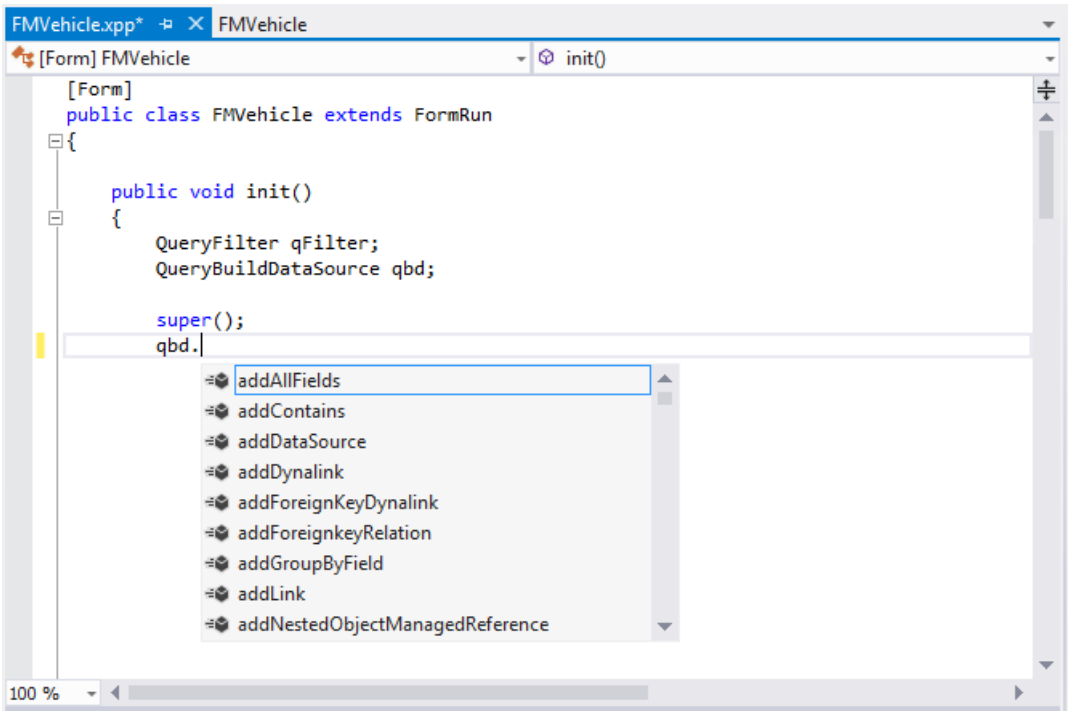
Code editor features

2/18/2021 • 2 minutes to read • Edit Online

This topic describes the code editor for Visual Studio.

Code editor

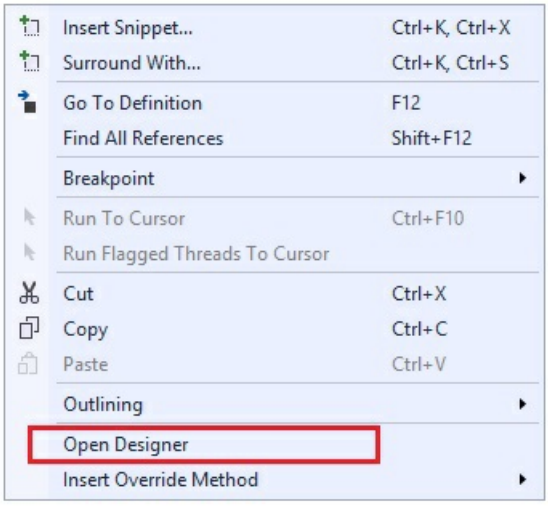
You use the code editor in Microsoft Visual Studio to write the X++ code for your applications. The X++ language is fully integrated into the Visual Studio environment. As you write your X++ code, you will see the familiar features of the Visual Studio code editor. For example, IntelliSense is displayed to help you write the code. You can also navigate to methods and classes in the code editor by using the navigation drop-down menus at the top of the code editor window.



Other features, such as collapsible sections, are also available.

Opening the element designer

You can open the element designer that corresponds to the current X++ source code by right-clicking in the code editor and then selecting **Open Designer**.



Additional resources

Develop and customize home page

Development tools in Visual Studio

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Create models and data model elements overview

2/18/2021 • 5 minutes to read • Edit Online

In this tutorial, you'll use Visual Studio's Dynamics 365 menu to create a new model named **Fleet Management tutorial**. You'll also create and edit new model elements.

Prerequisites

This tutorial requires that you have access to an environment, and that you be provisioned as an administrator

Keywords

- **Model** - You configure your model to refer to two other models. This enables your model to reference metadata and code elements that are in other packages.
- **Project** - You create a project and then associate your project to your new model. You add elements to your project, which are also added to your model. Specifically, you add an extended data type (EDT). You also add a table that you populate with fields and a method.
- **Designer** - Each time you add an item to your project, a designer is displayed that is tailored to the item type you selected. The **Properties** window adjusts each time a different node of the designer is highlighted. You make updates in the designers and in the **Properties** window.
- **EDT** - Extended data type.

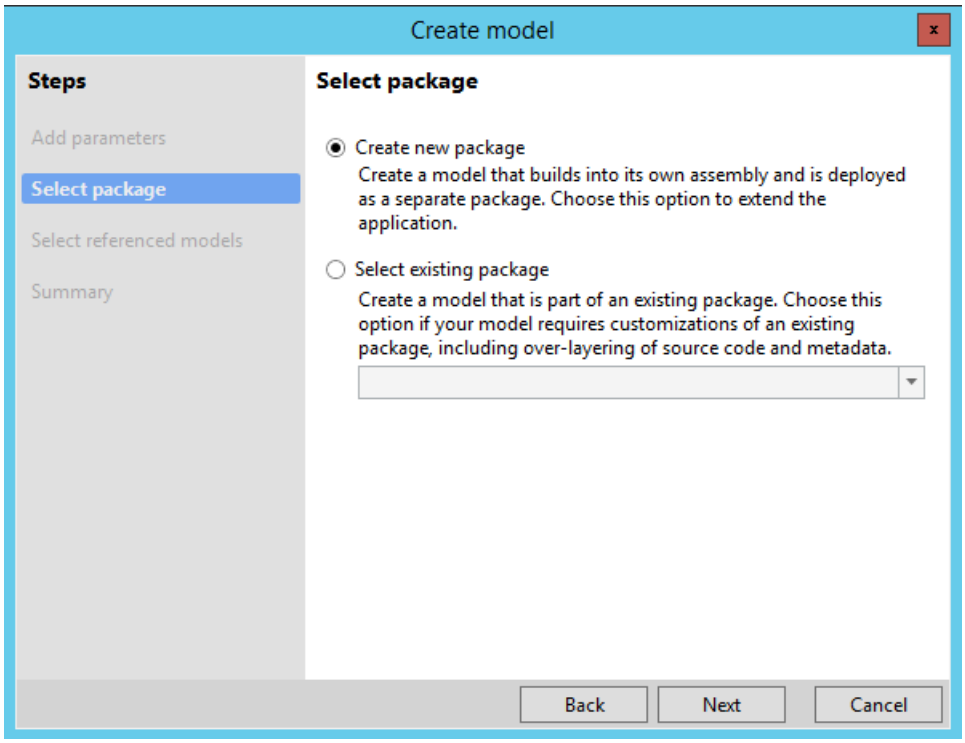
Create the Fleet Management tutorial model

1. Start Visual Studio using **Run as administrator**.
2. From the **Dynamics 365** window, select **Model Management > Create model** to open the **Create model** wizard.
3. Enter the following values for model parameters.

PROPERTY	VALUE
Model name	FleetMgmntTutorial
Model publisher	Microsoft Corp
Layer	isv
Model description	This tutorial shows how to build the Fleet Management application by using the Microsoft Dynamics AX development tools.
Model display name	Fleet Management Tutorial

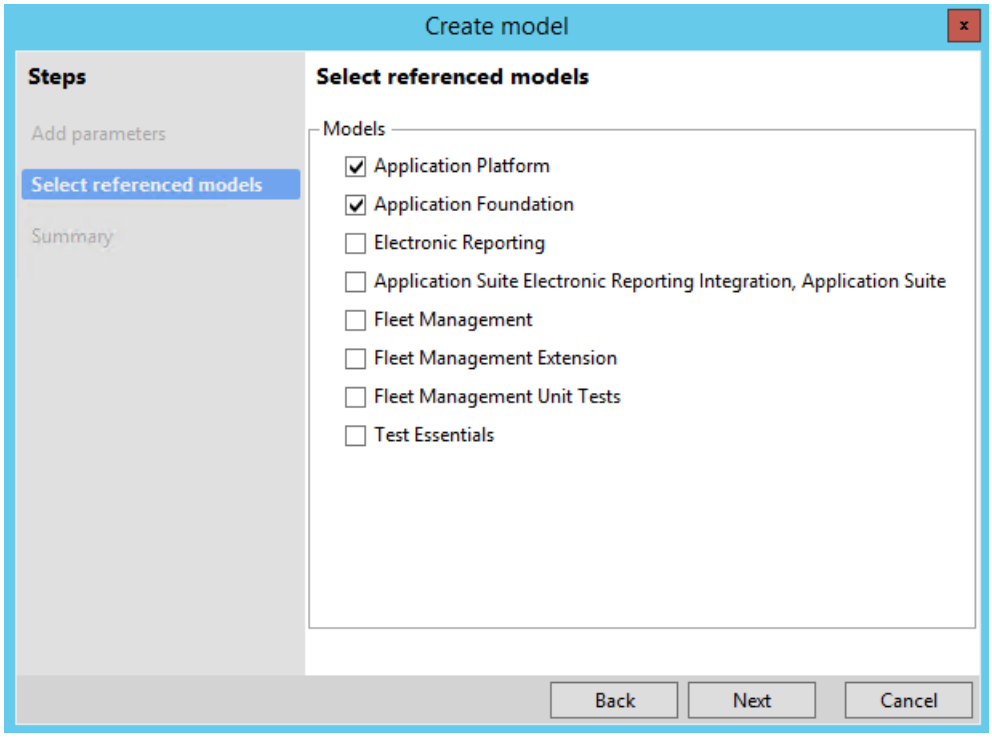
> **NOTE**
>
> Your model name must be **FleetMgmntTutorial**. Don't use any other name. In other tutorials, you'll overwrite model elements in this model by importing a project. If the model you create in this tutorial isn't named **FleetMgmntTutorial**, you may not be able to correctly import the project in other tutorials.

4. Click **Next** to advance to the next page, and then select **Create New Package**. The model you're creating will have its own package and build its own .NET assembly.



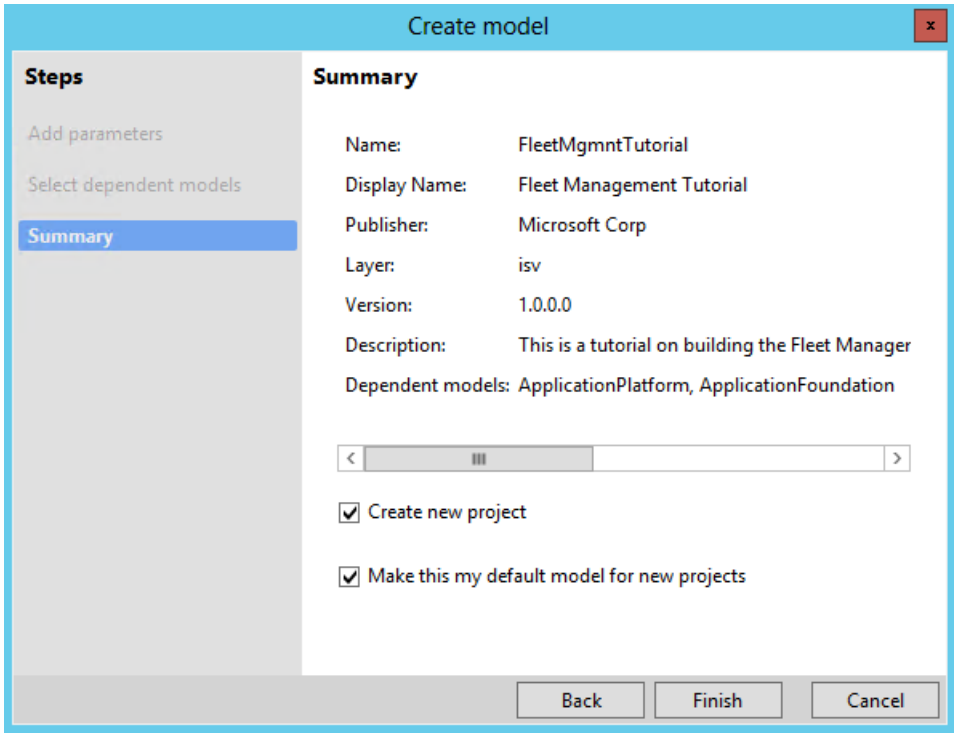
5. Click **Next** to advance to the **Select referenced models** step.

6. Select **Application Platform** and **Application Foundation** as referenced models.



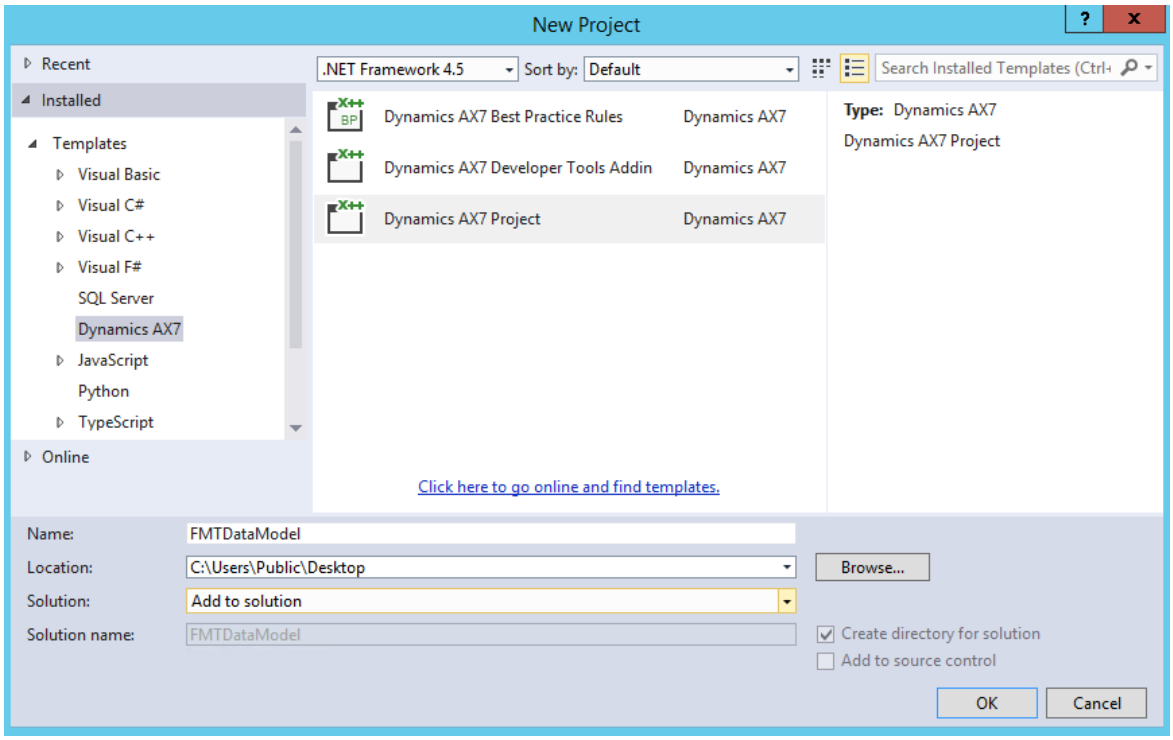
> **IMPORTANT**
>
> Verify that you've selected the correct referenced models.

7. Click **Next** to advance to the **Summary** step.
8. Verify the information on the summary page, and then select the **Create new project** and **Make this my default model for new projects** check boxes.



9. Click **Finish**. The **New Project** dialog box opens.
10. Under **Templates**, select **Dynamics 365**.
11. Select the **Unified Operations** template.
12. Enter the following values in the fields in the dialog box.

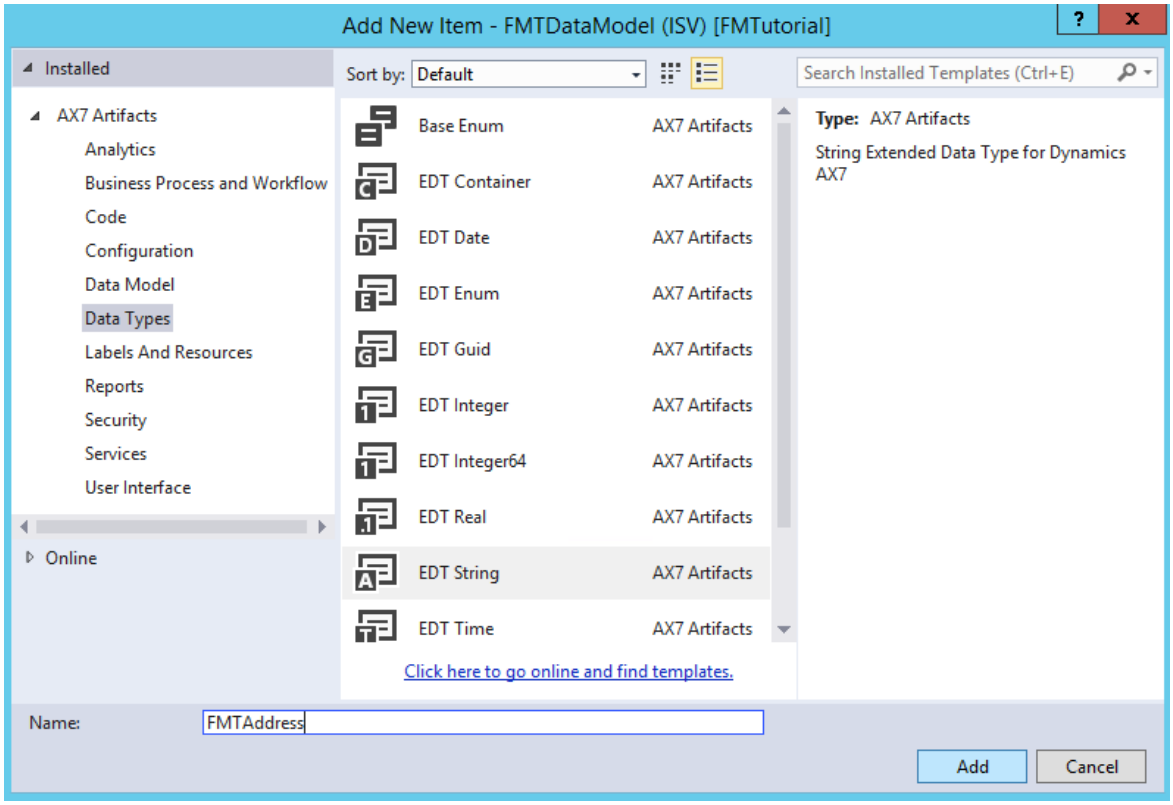
PROPERTY	VALUE
Name	FMTDataModel
Location	C:\FMLab
Solution	Add to solution



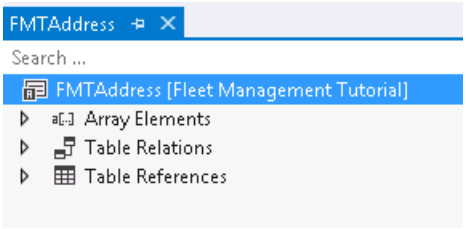
13. Click **OK** to create the project.

Create the FMTAddress extended data type

1. In **Solution Explorer**, right-click **FMTDataModel**, point to **Add**, and then click **New Item**.
2. Under **Dynamics 365 Items**, select **Data Types**.
3. Click **EDT String** to select the new item type.
4. In the **Name** field, enter **FMTAddress**, and then click **Add**.

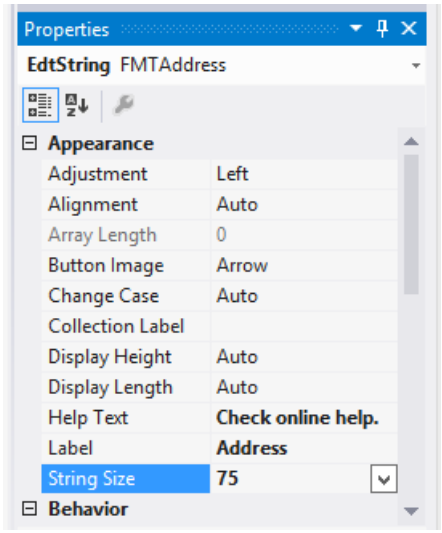


This adds a new EDT model element to the project, and opens the EDT designer for the new element, as shown in the following illustration.



5. Select the root node of **FMTAddress** in the designer.
6. In the **Properties** window, in the **Appearance section**, set the following properties.

PROPERTY	VALUE
Help Text	Check online help.
Label	Address
String Size	75

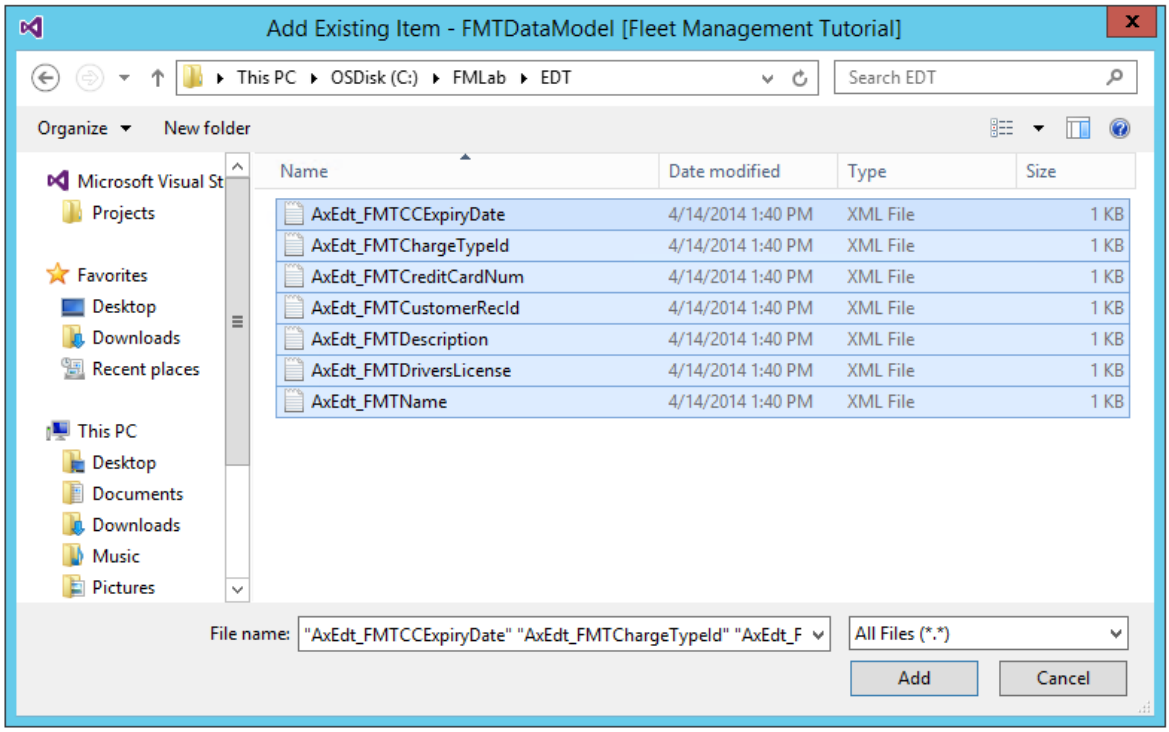


7. Press **Ctrl+S** to save the EDT.

Add existing model

Add the other required model element files to the current model and project. You can do this quickly by using the **Add existing item** feature.

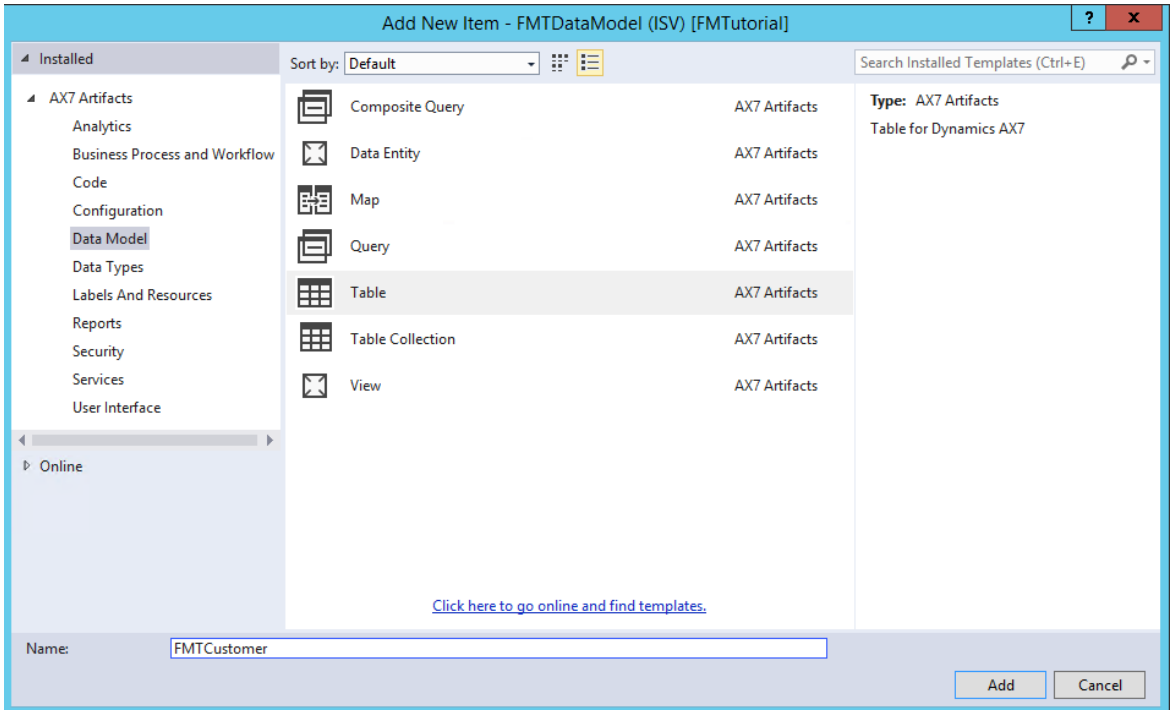
1. In the **Solution Explorer**, right-click **FMTDataModel**, point to **Add**, and then click **Existing Item**.
2. Browse to C:\FMLab\EDT\.



3. Press **Ctrl+A** to select all of the files, and then click **Add**.

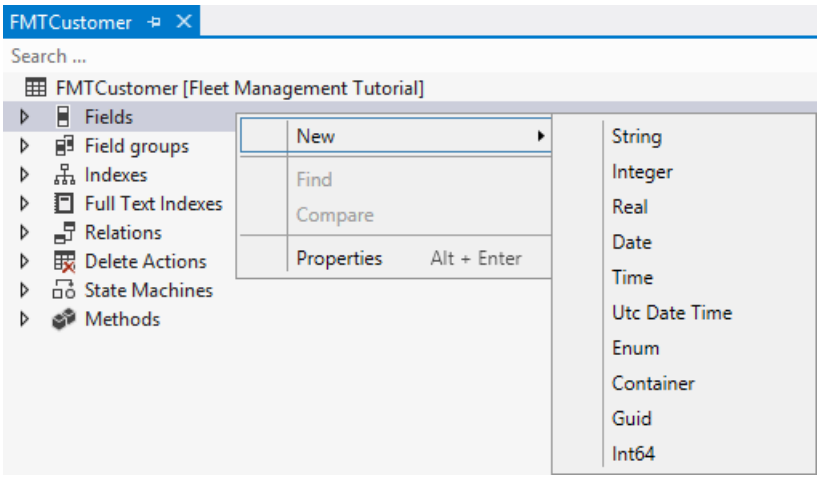
Create the FMTCustomer table

1. In **Solution Explorer**, right-click **FMTDataModel**, and then click **Add > New Item**.
2. In the left pane, expand **Installed**, expand **Dynamics 365 Items** and then click **Data Model**.
3. In the list of artifacts, select **Table**.
4. In the **Name** field, enter **FMTCustomer**, and then click **Add**. The table designer opens.



Add fields to the FMTCustomer table

In the table designer for FMTCustomer, you now add several fields to the table.

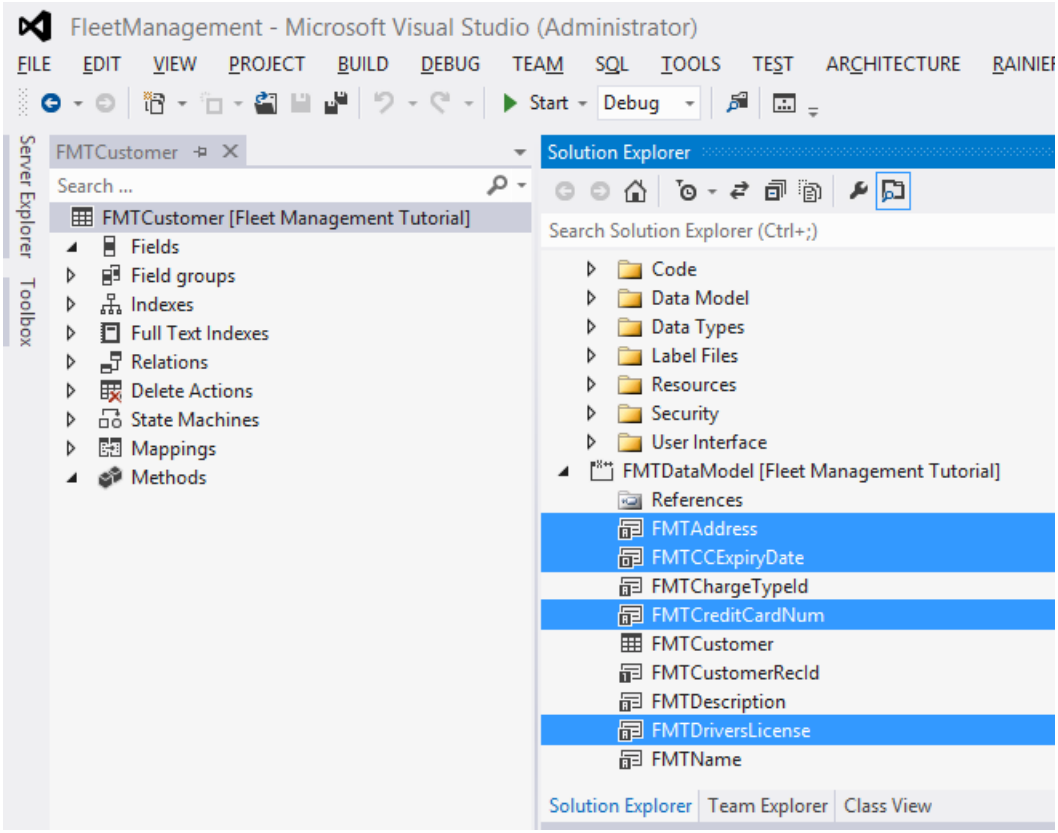


1. To add each field, right-click **Fields**, click **New**, and then select a type. As you add each field, you must specify the field name and certain other values in the **Properties** window, as described in the following table.

TYPE	FIELD NAME	PROPERTY VALUES
Date	CCExpiryDate	Extended Data Type = FMTCCExpiryDate
String	Address	Extended Data Type = FMTAddressHelp Text = Help text for the address field.
String	CellPhone	Extended Data Type = Phone
String	CreditCardNum	Extended Data Type = FMTCreditCardNum
String	DriversLicense	Extended Data Type = FMTDriversLicense
String	Email	String Size = 80Label = Email
String	FirstName	Extended Data Type = FirstName
String	LastName	Extended Data Type = LastName
String	License	String Size = 100Label = License
String	Thumbnail	String Size = 100Label = Thumbnail

TIP

For all new fields in the table that reference an EDT, you can create the field by simply dragging the EDT element from **Solution Explorer** or **Application Explorer** and dropping it on the **Fields** node of the **FMTCustomer** table in the designer.



2. Press **Ctrl+S** to save the new fields on the table.

Add fields to field groups

1. Prepare to add some of the fields to the **AutoSummary** field group by selecting the fields in the following list. To select multiple fields, hold down the Ctrl key while you click each field:

   - **Address**
   - **CCExpiryDate**
   - **CellPhone**
   - **CreditCardNum**
   - **DriversLicense**
   - **Email**
   - **FirstName**
   - **LastName**

2. Expand the **Field Groups** node.

3. Drag the selected fields to the **AutoSummary** node

4. Use the same technique to add the fields **FirstName**, **LastName**, and **CellPhone** to the **AutoReport** field group.

5. Save the table.

Add a method

1. Add the X++ method named **fullName** to the **FMTCustomer** table by right-clicking the **Methods** node, and then clicking **New Method**.

2. In the code editor, replace the default method code with the following code.

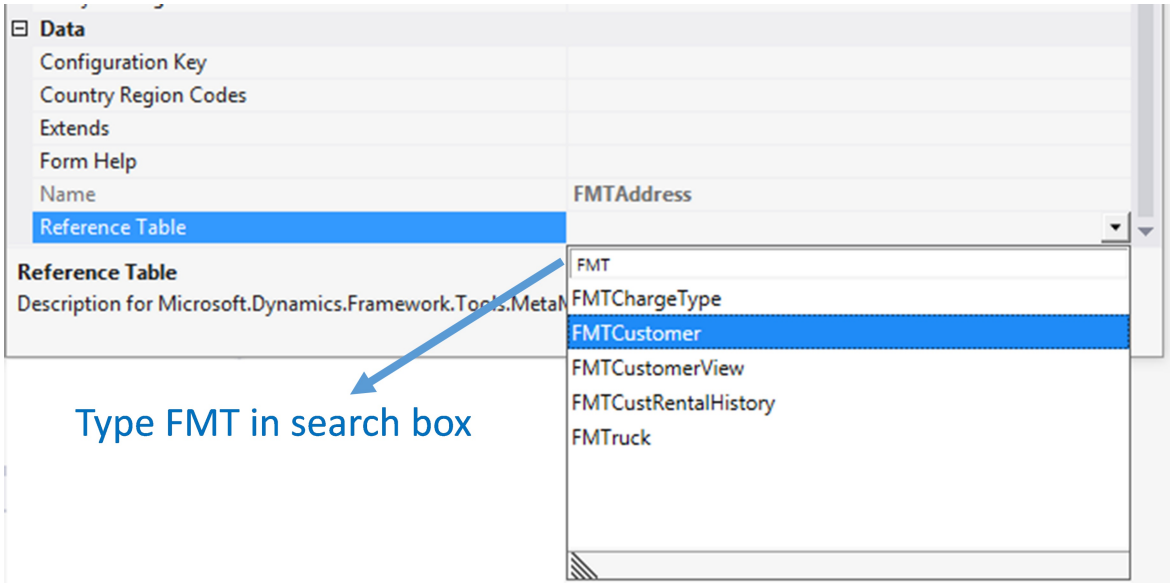
> **TIP**
>
> When you type “this.”, choose the field from the IntelliSense list.

```
public display FMTName fullName()
{
    return this.FirstName + ' ' + this.LastName;
}
```

3. Save the code.

Update the FMTAddress EDT

1. In **Solution Explorer**, expand the **FMTDataModel** project.
2. Right-click **FMTAddress**, and then click **Open**. The **EDT designer** opens.
3. In the **EDT designer**, select **FMTAddress**.
4. In the **Properties** window, in the **Reference Table** field, select **FMTCustomer**. **Tip:** Click the drop-down list, and then type the prefix "FMT" in the search box. This filters the drop-down list to only show tables that contain "FMT" in their name. Select the **FMTCustomer** table from the list of filtered entries.



5. Save the EDT.

Build the FMTDataModel project and the Fleet Management tutorial model

1. In **Solution Explorer**, right-click **FMTDataModel**, and then click **Rebuild**.
2. To do a full build of the entire model, on the **Dynamics 365** menu, click **Build models**.
3. Clear the check box for all models except for **Fleet Management Tutorial**.
4. On the **Options** tab, select the **Run Best practice checks** check box. Note that other options available.
5. On the **Models** tab, click **Build**.
6. Click **Close** in the dialog box.
7. On the **Window** menu, click **Close All Documents**, to close all open documents.

Turn off model customization and deprecate functionality

2/18/2021 • 6 minutes to read • Edit Online

This article describes the process of disabling customization of a model. By following this process, you make it ineligible for over-layering. Developers will still be able to extend that model. This article also describes how you can deprecate obsolete functionality.

Typically, you build an application in models that depend on other models. At the very least, your models will depend on the Application Platform model that is provided. Finance and Operations applications run on the Microsoft Azure cloud platform. In other words, it runs off-premises, in data centers that are managed by Microsoft. Because we can't support changes from a large number of vendors in the fundamental models, your applications can no longer over-layer artifacts in those models. Therefore, to build your applications, you must use extensions instead of over-layering. Even though all the artifacts in the models that you depend on are available for documentation purposes, you can't compile someone else's intellectual property and run it in the cloud.

Locking customizations

Transitioning a model from over-layering to extensions involves three steps:

1. You set a property that causes instances of over-layering to be flagged as warnings. This step is sometimes known as "soft-locking."
2. You have a period when you can burn down the warnings by using extensions instead of over-layering.
3. When you have resolved all the warnings, it's time to make over-layering a hard error that causes compilation to fail. This step is known as "hard-locking." When a model is hard-locked, the tooling that is required for over-layering can't be used for that model. Additionally, you can't have more than one model in a given package.

Currently, there is no tooling that you can use to manipulate the property that disables customizations. Instead, you must add the **Customization** XML element to the model descriptor file, as shown in the following example. You can find the model descriptor file at ...\<packages>\<package name>\Descriptor\<model name>.xml.

```
<?xml version="1.0" encoding="utf-8"?>
<AxModelInfo xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
<Customization>DoNotAllow</Customization>
<Description>My cool locked application</Description>
...
<</AxModelInfo>
```

There are three levels of customization settings:

- **Allow customizations** – If the **Customization** element is omitted, or if it's provided but the text **Allow** is used inside it, there are no restrictions.
- **Soft-locking** – If you require soft-locking, use the text **AllowAndWarn** inside the **Customization** element.
- **Hard-locking** – To disable customization of the model, use the text **DoNotAllow** inside the **Customization** element, as shown in the preceding example.

> **NOTE**
>
> Elements in the descriptor file must be listed in alphabetical order.

Backward compatibility of the platform

There is a requirement that your application must continue to run even if a newer version of a dependent platform model is installed. Therefore, we enforce strict requirements for backward compatibility on all the changes that we make in those platform models. We can clarify this point through an example. For this example, you have a base model M that has the following class.

```
public void DoSomething(int arg)
{
    // Do great stuff
}
```

In your model that depends on model M, you want to take advantage of all the great functionality that the **DoSomething** class offers. Therefore, you have the following code.

```
{
    var c = new SomeClass();
    c.DoSomething(42);
}
```

Later, your vendor (perhaps Microsoft) releases a new, updated version of the model that you depend on (model M). When this model is released, we must support the public interface, because things must run without recompilation. Suppose that we changed the method signature by adding another parameter, as shown here.

```
public void DoSomething(int arg, str anotherArg)
{
    // Do greater stuff
}
```

In this case, we break the app. The common language runtime (CLR) can't call the method, because the parameter profile that is assumed in the call isn't the same as the parameter profile of the callee (the calling code provided one parameter, but two are now required). Therefore, it's a good idea to limit what is publicly consumable. If something is private, nobody can use it from outside your model. However, another option is to make the artifact internal. If you apply the **internal** modifier to a class or method, the artifact is visible only inside your model and can't be reached from the outside. Essentially, you must assume that anything that isn't internal or private will be used from outside your model. Therefore, you must support it forever. Never make anything public or protected unless it absolutely must be public or protected, and use interfaces to hide implementation details that aren't relevant outside the boundary of your model. There is no way to mark metadata (as opposed to code) with the internal visibility specifier.

Testing internal functionality

By limiting the surface area of your model, you help guarantee that you will be able to fix issues in your models that do not allow customizations, and that the application will run smoothly. However, you might argue that, by making things internal, you limit the ability to test your code. To remedy this situation, you can inform your test packages about the packages that they should test. In other words, the test packages can access internal things. To implement this solution, you edit the descriptor file, as shown in the following example.

```
<?xml version="1.0" encoding="utf-8"?>
<AxModelInfo xmlns:i=http://www.w3.org/2001/XMLSchema-instance>
<InternalsVisibleTo xmlns:d2p1="http://schemas.microsoft.com/2003/10/Serialization/Arrays">
<d2p1:string>ExternalPackageName</d2p1:string>
</InternalsVisibleTo>
...
</AxModelInfo>
```

You can provide any number of external packages by including them in the **InternalsVisibleTo** element. This information is provided for the package that contains the code to test, not for the package that contains the testing code. In other words, the package determines who it shares information with. **Note:** The elements under **AxModelInfo** must be in alphabetical order.

Deprecating functionality

Deprecating methods

Sometimes, you no longer want to support source code that is public. As we mentioned earlier, you should not remove an artifact from the code unless that artifact is private or internal, because there might be users who rely on it. Instead, you can use the **SysObsoleteAttribute** attribute to specify that consumers should no longer use that artifact. We recommend that you mark things as obsolete in two phases:

1. Specify that using the artifact is flagged as a warning.
2. After one or more release cycles, make using the artifact an error.

In the following example, the **DoSomething** method is defined in the first release of a model.

```
class SomeApi
{
    void DoSomething()
    {
        // Do great stuff
    }
}
```

In the second release, you've determined that there is a better way to accomplish something. Therefore, you add the new implementation and make the old implementation obsolete.

```
class SomeApi
{
    [SysObsolete("Use DoSomethingNew instead", false)]
    void DoSomething()
    {
        // Do great stuff
    }

    void DoSomethingNew()
    {
    }
}
```

All your existing customers will continue to call the old version. This situation is fine, but the customers won't be able to take advantage of the benefits of the new version. Anyone who codes against your class will receive a build warning together with the message that you provided in the **SysObsolete** attribute argument. Presumably, these clients will update their code so that it uses the new method. As time passes, more clients will move to the new version. Therefore, at some point, it will make sense for you to make coding against the method a hard error.

```
class SomeApi
{
    [SysObsolete("Use DoSomethingNew instead", true)]
    void DoSomething()
    {
        // Do great stuff
    }

    void DoSomethingNew()
    {
    }
}
```

Again, for the reasons that we mentioned earlier, you may never be able to get rid of the old method completely, because it was not made private or internal.

Deprecating metadata

For deprecating model elements (tables, data entities, EDTs, Enums, ...etc.), use the property **IsObsolete** that is available on all model element types. **IsObsolete** is also available on table, view, and data entity fields. When you set **IsObsolete** to Yes, references to that element or field will cause compilation warnings.

Customization Analysis Report (CAR)

2/18/2021 • 11 minutes to read • Edit Online

This article describes how to generate a Customization Analysis Report for your model. It also describes some best practice rules that are included in the report, and provides suggestions for fixing errors and warnings that are associated with these rules.

What is the Customization Analysis Report?

The Customization Analysis Report is a tool that analyzes your customization and extension models, and runs a predefined set of best practice rules. The report is one of the requirements of the solution certification process. The report is in the form of a Microsoft Excel workbook.

How to generate the report

To generate the Customization Analysis Report, run the following command in a development environment.

```
xppbp.exe -metadata=<local packages folder> -all -model=<ModelName> -xmlLog=C:\BPCheckLogcd.xml -module=
<PackageName> -car=<reportlocation>
```

Example

```
xppbp.exe -metadata=C:\Packages -all -model="MyAppSuiteCustomizations" -xmlLog=C:\temp\BPCheckLogcd.xml -
module="ApplicationSuite" -car=c:\temp\CAReport.xlsx
```

The xppbp.exe tool is located in c:\packages\bin or I:\AosService\PackagesLocalDirectory\bin.

Issues List

This section describes all best practice rules (errors, warnings, or informational messages) that can appear on the **Issues List** page of the report and provides suggestions for fixing the issues. Issues are of the **metadata** or **code** type. For all **code** issues, keep in mind that, if the warning or error occurs in a method that you've overlaid, the lines of code that violate the rule might belong to a model in a lower layer. In that case, you're not responsible for fixing warnings and errors in code that isn't yours.

BPCheckPackReturnsConnull

DESCRIPTION	THIS RULE APPLIES TO ALL CLASSES THAT IMPLEMENT THE SYSPACKABLE INTERFACE. IT MAKES SURE THAT THE PACK METHOD DOESN'T RETURN CONNULL().
Error message	Container pack method returns connull in a Runbase derived class
Issue type/severity	Code/Warning
How to fix it	Update the return value of the **Pack** method, or return **Super()** when applicable.

BPCheckParametersModified

DESCRIPTION	THIS RULE FAILS IF A PARAMETER OF A METHOD IS MODIFIED INSIDE A METHOD.
Error message	Parameter 1% in method %1 are modified inside the method
Issue type/severity	Code/Warning
How to fix it	Refactor your code. Parameters must be modified by the caller, not within the called method.

BPCheckSQLCode

DESCRIPTION	THIS RULE FAILS IF YOUR X++ CODE CONTAINS DIRECT SQL CODE.
Error message	SQL code found in method %1
Issue type/severity	Code/Warning
How to fix it	Refactor your code to use X++ to access the database.

BPCheckNestedLoopInCode

DESCRIPTION	THIS RULE FAILS IF IT FINDS A WHILE SELECT LOOP NESTED WITHIN A LOOP.
Error message	Nested data access loop found in %1 method
Issue type/severity	Code/Warning
How to fix it	Refactor your code to use joins instead of nested data access loops. If you can't refactor the code without altering the business logic of your method, document an exception when you submit your Customization Analysis Report to Microsoft.

BPCheckInsertMethodInLoop

DESCRIPTION	THIS RULES FAILS IF IT FINDS A CALL TO THE METHOD INSERT NESTED INSIDE A LOOP. WE RECOMMEND THAT YOU USE INSERTRECORDLIST. THIS RULE DOESN'T APPLY TO INMEMORY TABLES.
Error message	Insert method can be replaced with RecordInsertList in method %1
Issue type/severity	Code/Warning
How to fix it	Refactor your code to use set-based operations, such as **InsertRecordList**.

BPCheckSelectwithJoin

DESCRIPTION	THIS RULE FAILS IF IT FINDS NESTED SELECT STATEMENTS THAT AREN'T JOINED.
Error message	Nested select statement in %1 method can be joined

DESCRIPTION	THIS RULE FAILS IF IT FINDS NESTED SELECT STATEMENTS THAT AREN'T JOINED.
Issue type/severity	Code/Warning
How to fix it	Refactor your code to use joins instead of a nested **select** statement. If you can't refactor the code without altering the business logic of your method, document an exception when you submit your Customization Analysis Report to Microsoft.

BPFunctionCallwithSelect

DESCRIPTION	THIS RULE FAILS IF A FUNCTION CALL IS FOUND WITHIN A SELECT STATEMENT.
Error message	Function call found in select statement in method %1
Issue type/severity	Code/Warning
How to fix it	Assign the function's return value to a local variable before you call the **select** statement, and use the local variable in the **select** statement.

BPCheckinvalidExecuteQuery

DESCRIPTION	THIS RULE FAILS IF A CALL TO ADDRANGE IS FOUND AFTER A CALL TO SUPER() IN THE EXECUTEQUERY METHOD.
Error message	Add range after super() should not be added in ExecuteQuery method for form %1
Issue type/severity	Code/Warning
How to fix it	Refactor your code to avoid this pattern.

BPCheckInvalidInitFormMethod

DESCRIPTION	THIS RULE MAKES SURE THAT YOU DON'T INITIALIZE FORM CONTROLS AND DATA SOURCES IN A FORM'S INIT METHOD BEFORE YOU CALL SUPER(). IT FAILS IF IT FINDS A STATEMENT THAT USES ELEMENT OR THIS BEFORE THE CALL TO SUPER() (THIS.AMETHOD() OR ELEMENT.AMETHOD()). AN INFORMATIONAL MESSAGE IS SHOWN ONLY FOR SOME ALLOWED PATTERNS, SUCH AS INITIALIZING NUMBER SEQUENCES (NUMBERSEQPREINIT).
Error message	Form element statements should not be used before super() in init method of form %1
Issue type/severity	Code/Info or Warning
How to fix it	Refactor your code to access form controls and data after the call to **super()**. If your code doesn't initialize any form control and doesn't access any form data sources before **super()**, document an exception when you submit your Customization Analysis Report to Microsoft.

BPCheckEmptyLoop

DESCRIPTION	THIS RULE FAILS IF IT FINDS LOOPS THAT HAVE NO STATEMENTS.
Error message	Empty loop found in method %1 in class %2
Issue type/severity	Code/Warning
How to fix it	Remove the loop from your code.

BPCheckLockQueryRange

DESCRIPTION	THIS RULE FAILS IF THE CODE CALLS ADDRANGE IN THE INIT METHOD OF THE FORM, AND DOESN'T SET THE RANGE AS LOCKED OR HIDDEN.
Error message	Range should be locked or hidden in form %1
Issue type/severity	Code/Warning
How to fix it	Add **QueryBuildRange.status(RangeStatus::Locked)** or **QueryBuildRange.status(RangeStatus::Hidden)** after the call to **AddRange**.

BPCheckSkipStatementValidation

Description	This rule reports an informational message if it finds a set-based operation that doesn't have **Skip** statements. • When **update_recordset** is found, the rule checks for **skipDataMethods(true)**. The rule applies only when the **update** method on the table is overridden. • When **insert_recordset** is found, the rule checks for **skipDataMethods(true)**. The rule applies only when the **insert** method on the table is overridden. • When **delete_from** is found, the rule checks for **skipDeleteActions(true)**. The rule applies only when the **insert** method on the table is overridden. This is an informational message that highlights the need to call **skip** methods to take full advantage of the performance gains of set-based operations. If **skip** methods are not used, the execution falls back to a row-by-row operation.
Error message	Set-based operation must invoke Skip statements in method %1 in class %2; otherwise, execution will fall back to a row by row operation.
Issue type/severity	Code/Information
How to fix it	Use **skipDataMethods(true)** or **skipDeleteActions(true)** when applicable.

BPCheckNoTTSTryBlock

DESCRIPTION	THIS RULE FAILS IF IT FINDS A TRY STATEMENT WITHIN A TTS BLOCK THAT ISN'T HANDLED CORRECTLY.
Error message	Tts block with Try statement does not explicitly catch exceptions in method %1

DESCRIPTION	THIS RULE FAILS IF IT FINDS A TRY STATEMENT WITHIN A TTS BLOCK THAT ISN'T HANDLED CORRECTLY.
Issue type/severity	Code/Warning
How to fix it	Use the code example that follows this table.

The following examples show when the rule will fail or pass. Use these examples as guidelines to refactor your code.

```
ttsbegin;
try {
}
// fail
catch {
}
try {
}
// pass
catch(Exception::UpdateConflict) {
}
try {
}
// pass
catch(Exception::UpdateConflictNotRecovered) {}
```

BPCheckEmptyTableMethod

DESCRIPTION	THIS RULE CHECKS FOR TABLE METHODS THAT HAVE NO SOURCE CODE. EMPTY TABLE METHODS CAUSE RECORD-SET OPERATIONS ON THE TABLE TO FALL BACK TO A ROW-BY-ROW OPERATION.
Error message	%1 table has an empty %2 method
Issue type/severity	Code/Warning
How to fix it	Remove this method from your code.

BPCheckBatchJobsEnabled

DESCRIPTION	THIS RULE MAKES SURE THAT ALL CLASSES THAT EXTEND RUNBASEBATCH HAVE A CANGOBATCH METHOD THAT RETURNS TRUE.
Error message	Custom job is not batch-enabled in class %1
Issue type/severity	Code/Warning
How to fix it	The **canGoBatch** method must return **true** for all batch classes.

BPCheckDisplayMethodCached

Description	For each control that is bound to a data method, this rules fails if one of these conditions is met: • The **Cache Data Method** property is set to **Auto**, the corresponding table **display/edit** method doesn't have the **SysClientCacheDataMethodAttribute**, and the **init** method of the data source doesn't use **CacheAddMethod**. • The **Cache Data Method** property is set to **No**, and the **init** method of the data source doesn't use **CacheAddMethod**.
Error message	Display method %1 in form %2 not cached
Issue type/severity	Code/Warning
How to fix it	Use the note that follows this table.

You can fix this warning by using one of the following patterns:

- Set the **Cache Data Method** property to **Yes**.
- Set the **Cache Data Method** property to **Auto**, and mark the data method of the table with the **SysClientCacheDataMethodAttribute** attribute. Here is an example.

```
[SysClientCacheDataMethodAttribute(true)]
Display TransDate myDateMethod()
{
    //...
}
```

- Use **CacheAddMethod** in the **init** method of the form to mark the method as cached.

BPCheckSQLQueryInInit

DESCRIPTION	THIS RULE FAILS IF A DATA ACCESS QUERY THAT HAS A WHILE LOOP IS FOUND IN THE INIT METHOD OF A FORM. THIS PATTERN COULD CAUSE PERFORMANCE ISSUES WHEN THE FORM IS INITIALIZED.
Error message	SQL query with while loop was found in init method of form %1
Issue type/severity	Code/Warning
How to fix it	Refactor your code to avoid this pattern.

BPCheckNewQueryWithForm

DESCRIPTION	THIS RULE FAILS IF IT FINDS THE NEW QUERY() STATEMENT IN THE INIT OR EXECUTEQUERY METHOD OF A FORM.
Error message	Data source query overridden in form method %1
Issue type/severity	Code/Warning
How to fix it	Refactor your code to avoid this pattern.

DESCRIPTION	THIS RULE FAILS IF IT FINDS THE NEW QUERY() STATEMENT IN THE INIT OR EXECUTEQUERY METHOD OF A FORM.

BPCheckSelectForUpdateAbsent

DESCRIPTION	THIS RULE FAILS IF SELECT FORUPDATE IS USED TO SELECT RECORDS, BUT AN UPDATE ISN'T BEING PERFORMED. THIS RULES DOESN'T APPLY TO TABLES THAT ARE ENABLED FOR OPTIMISTIC CONCURRENCY CONTROL (OCC).
Error message	Select ForUpdate not implemented in method %1
Issue type/severity	Code/Warning
How to fix it	Use **Select** instead of **Select ForUpdate**, or set the **OCC Enabled** property to **Yes** on the table.

BPCheckTablePropertyMismatch

Description	This rule fails when the table belongs to a table group but doesn't have the appropriate **Cache Lookup** value. The rule fails if one of these conditions is met: • The **Table Group** property is set to **Main**, and the **Cache Lookup** property is set to **NotinTTS** or **EntireTable**. • The **Table Group** property is set to **Group** or **Parameter**, and the **Cache Lookup** property is set to **NotinTTS**. • The **Table Group** property is set to **WorksheetHeader**, **WorksheetLine**, or **Transaction**, and the **Cache Lookup** property is set to **Found**, **FoundAndEmpty**, or **EntireTable**.
Error message	%1 table has an invalid combination of table group and cache lookup
Issue type/severity	MetaData/Warning
How to fix it	Set an appropriate **Cache Lookup** value on the table.

BPCheckMissingDeleteActions

DESCRIPTION	THIS RULE VALIDATES THAT THE VALUE OF EITHER A DELETE ACTION OR THE ON DELETE PROPERTY OF A TABLE RELATION ISN'T EQUAL TO NONE. THE RULE ISN'T TRIGGERED WHEN THE RELATED OR CURRENT TABLE IS A TEMPDB, IN MEMORY TABLE, REFERENCE TABLE, STAGING TABLE, OR PARAMETER TABLE.
Error message	Delete actions missing in table %1 which is related to table %2 with relation name %3
Issue type/severity	MetaData/Warning
How to fix it	Set a **delete** action value that isn't equal to **None**.

BPCheckAddressModel

DESCRIPTION	THIS RULE FAILS IF A TABLE FIELD IS OF THE ADDRESSZIPCODEID OR ADDRESSSTATEID TYPE. THESE TYPES INDICATE THAT THE CODE DIDN'T UPTAKE THE NEWER ADDRESS FRAMEWORK.
Error message	Field %1 of table %2 is not part of address location model
Issue type/severity	MetaData/Warning
How to fix it	Replace these types with any other appropriate EDT type in the Directory model.

BPCheckDimensionModel

DESCRIPTION	THIS RULE FAILS IF A TABLE FIELD IS OF THE DIMENSION OR LEDGERACCOUNT TYPE. THESE TYPES INDICATE THAT THE CODE DIDN'T UPTAKE THE NEWER FINANCIAL DIMENSION FRAMEWORK.
Error message	Field %1 of table %2 is not part of financial dimension framework
Issue type/severity	MetaData/Warning
How to fix it	Replace these types with any other appropriate EDT type in the Dimensions model.

BPCheckNumberofNewFields

DESCRIPTION	THIS RULE VERIFIES THAT NO MORE THAN 10 FIELDS WERE ADDED TO A TABLE DURING THE PROCESS OF CUSTOMIZING OR EXTENDING THAT TABLE. THIS RULE DOESN'T APPLY TO STAGING TABLES.
Error message	Number of new fields in table %1 is greater than
Issue type/severity	Metadata/Warning
How to fix it	Refactor your schema and create related tables instead of adding too many fields to an existing table.

BPCheckEnumUpgradeIssue

DESCRIPTION	THIS RULE FAILS WHEN THE FOLLOWING CONDITION IS MET: WHEN YOU CUSTOMIZE AN ENUM (OVERLAYER), NEW ENUM VALUES ARE LESS THAN THE MAXIMUM EXISTING VALUE + 10. THIS RULES DOESN'T APPLY TO EXTENSIBLE ENUMS.
Error message	Enum %1 will have upgrade issues
Issue type/severity	MetaData/Warning
How to fix it	Use larger enum values, or extend the enum.

BPCheckPassiveJoinUse

Description	This rule validates that, when a form allows for pre-loading of data, the link type of tab page data sources is passive. The rule fails if all the following conditions are met: • A form has the **AllowPreLoading** property set to **Yes**. • A tab page on the form is bound to a top-level data source. • The data source's **Join Source** property is set. • The data source's **Link Type** property isn't set to **Passive**.
Error message	New message suggest: "Use passive joins on data sources %1 bound to a tab page control in form %2"
Issue type/severity	MetaData/Warning
How to fix it	Set the **AllowPreLoading** property to **No** on the form, or use passive joins on the data source. Passive joins require that you explicitly program when the query of a tab page is run.

BPCheckAlternateKeyAbsent

DESCRIPTION	THIS RULE VERIFIES THAT TABLES THAT HAVE A UNIQUE INDEX HAVE AT LEAST ONE ALTERNATE KEY.
Error message	Table %1 does not have an alternate key
Issue type/severity	MetaData/Warning
How to fix it	Set the **Alternate Key** property to **Yes** on a unique index of the table.

BPCheckBaseTableModified

DESCRIPTION	THIS RULE DISCOURAGES THE CUSTOMIZATION (OVERLAYERING) OF A LIST OF SPECIFIC BASE TABLES THAT ARE SHIPPED BY MICROSOFT. EXAMPLES ARE THE SOURCEDOCUMENTHEADER AND SOURCEDOCUMENTLINE TABLES.
Error message	%1 is a base table and must not be modified
Issue type/severity	MetaData/Warning
How to fix it	Don't customize the table.

Element designers

2/18/2021 • 3 minutes to read • Edit Online

This topic reviews the element designers and explains how to use them. The tools contain designers for each kind of element in the program. You use these designers when you create or modify elements.

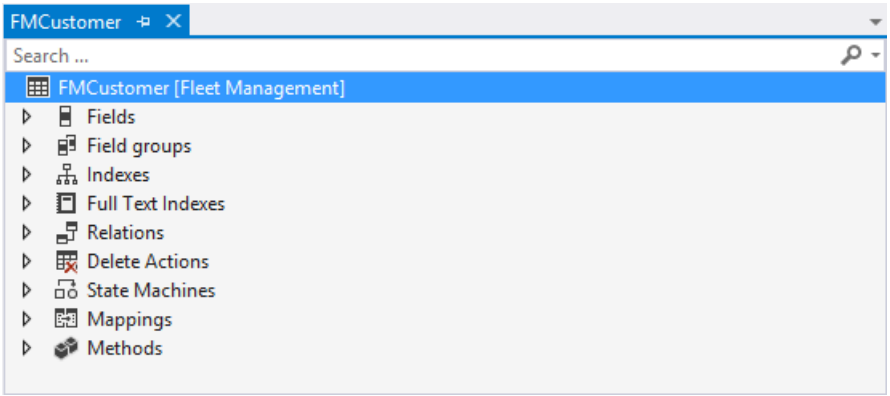
Open an element designer

To open an element designer, follow these steps.

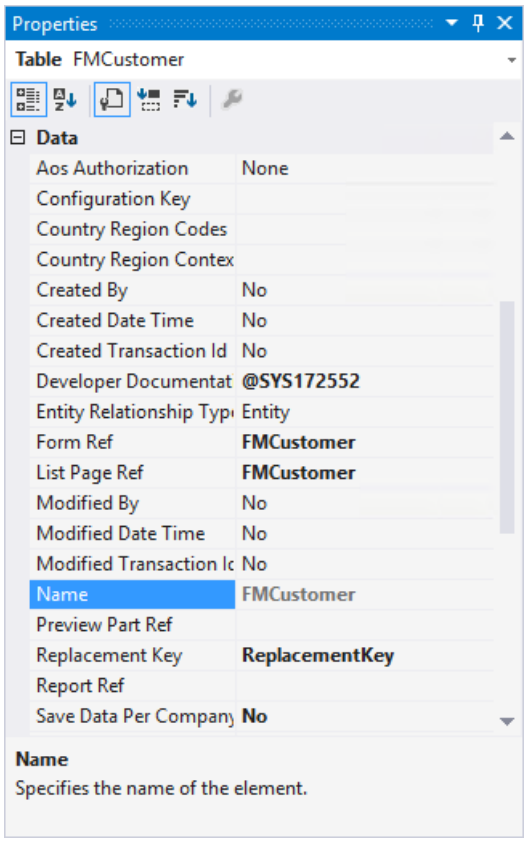
1. Find the element in the project or in Application Explorer.
2. Right-click the element. In Application Explorer, click **Open designer**. In the project, click **Open**.
3. Expand the nodes in the element designer to see the details about the element.

Node properties

When you select the individual nodes in the element designer, the **Properties** pane in Visual Studio shows the various properties for that node. Most of the characteristics of an element are controlled by these properties. For example, the following illustration shows the element designer for the FMCustomer table. Notice that the top-level node is selected.



The following illustration shows the set of properties for the table, which corresponds to the top-level node that is selected.

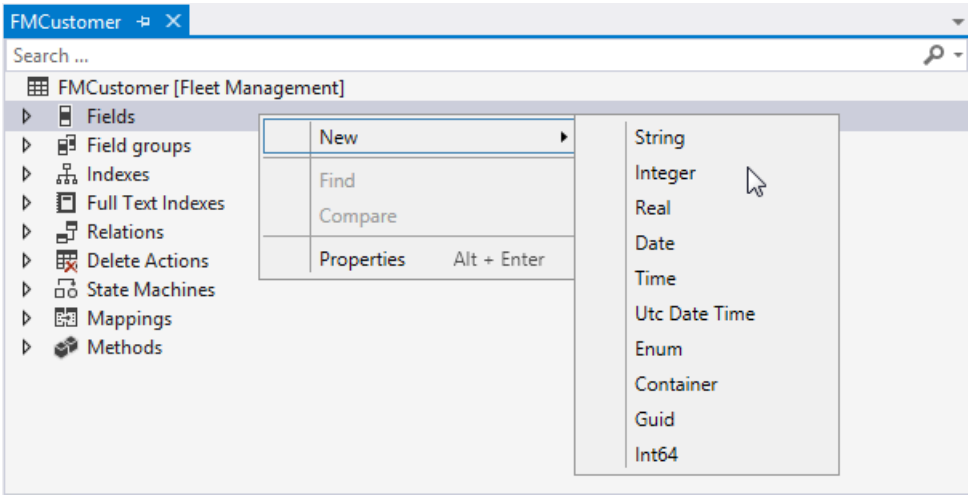


Each node for the element will have a set of properties that applies to it. To make it easier to find the properties that you want to work with, use the buttons at the top of the **Properties** pane to control how they are displayed. The properties can be arranged in the following ways.

ORGANIZATION	DESCRIPTION
Alphabetical	Arrange the properties in alphabetical order.
Categorized	Arrange the properties into standard categories for the node type.
Changed	Divide the properties into those that have been changed and those that use the default values.
Frequency	Divide the properties into categories, based on whether a property is often, occasionally, or rarely changed.

Working with nodes

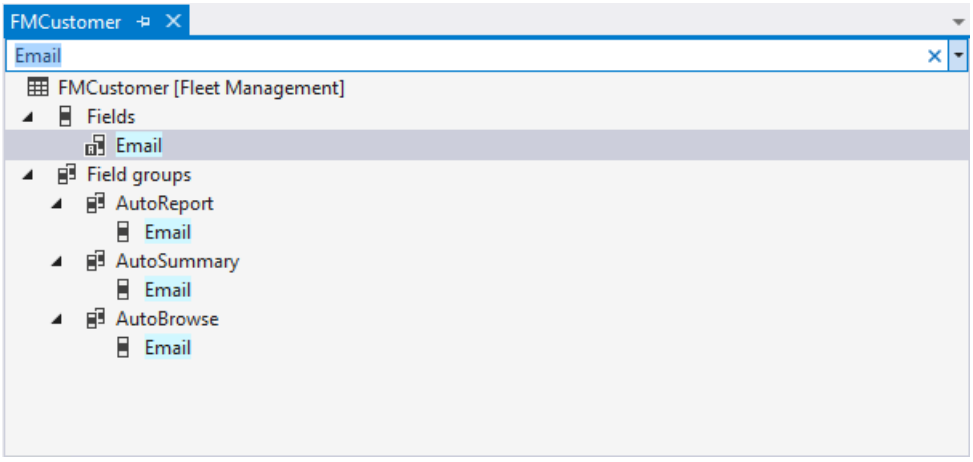
When you create or modify elements, you will often find that you must add or remove nodes for the element. For example, to add a field to a table, right-click the **Fields** node for the table element, point to **New**, and then click the type of field to add.



To remove a node, right-click the node, and then click **Delete**. You can also perform other actions for a node. You rename a node, duplicate a node, or move the node up or down in the node list.

Searching element nodes

Sometimes, the node list for an element can be long, so that it's difficult to find the specific node that you're looking for. Notice that there is a search bar at the top of each element designer. You can enter a string to search for, and the node list will be filtered to include only the nodes that match the search string. For example, the following illustration shows the element designer for the FMCustomer table after the “Email” search term was applied. Only nodes that have names that match that search term appear in the element designer.

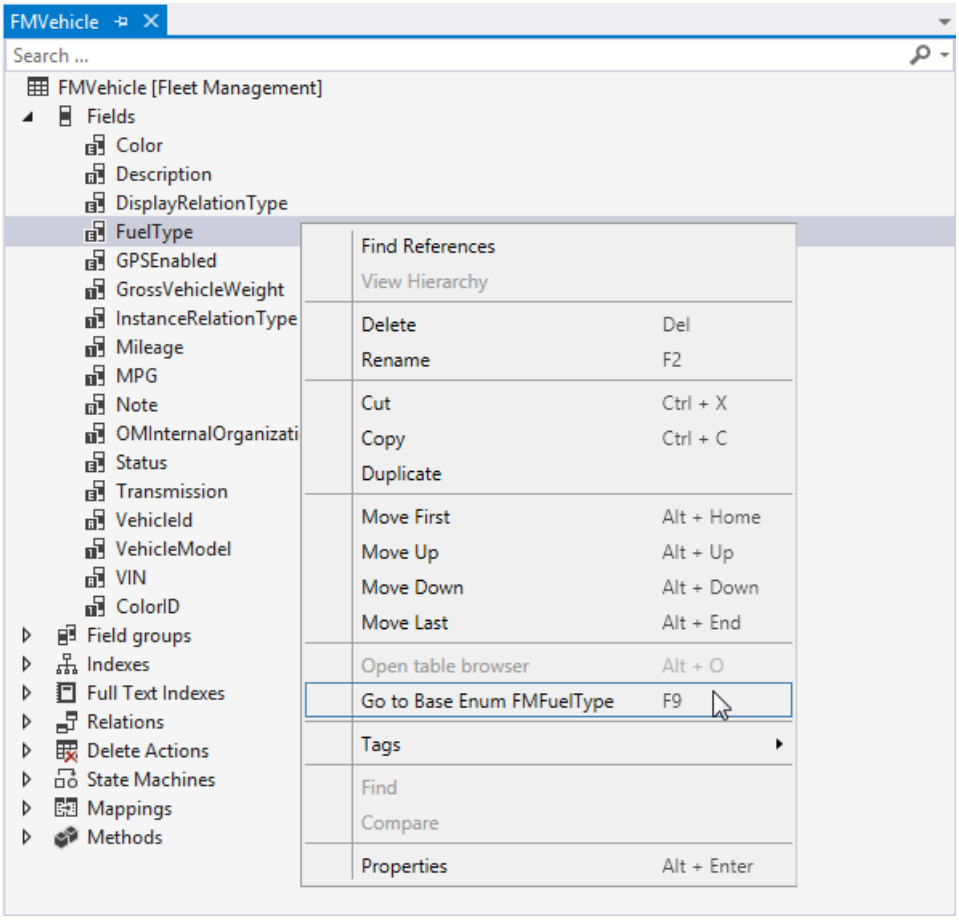


If you're working with a customization element or an extension element, you can prefix your search string with **c:** (for "customization elements") or **e:** (for "extension elements") to return only customizations or extensions, respectively. **Examples**

- **e:** returns all extensions that belong to the current element.
- **c:** returns all customizations that belong to the current element.
- **e:Email** returns all extension nodes that have the string "Email" in their name.
- **c:Email** returns all customization nodes that have the string "Email" in their name.

Navigating to related elements

The value of a node in the element designer is often a reference to another element. For example, a field node in a table element is typically based on an extended data type (EDT) element. When you right-click a node in the element designer, you can click the **Go to <element>** command to navigate to that related element. For example, when you right-click the **FuelType** node in the list of fields for the FMVehicle table, you can click **Go to Base Enum FMFuelType** to show the base enumeration that is used to define the field.



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Commands for determining how elements are used

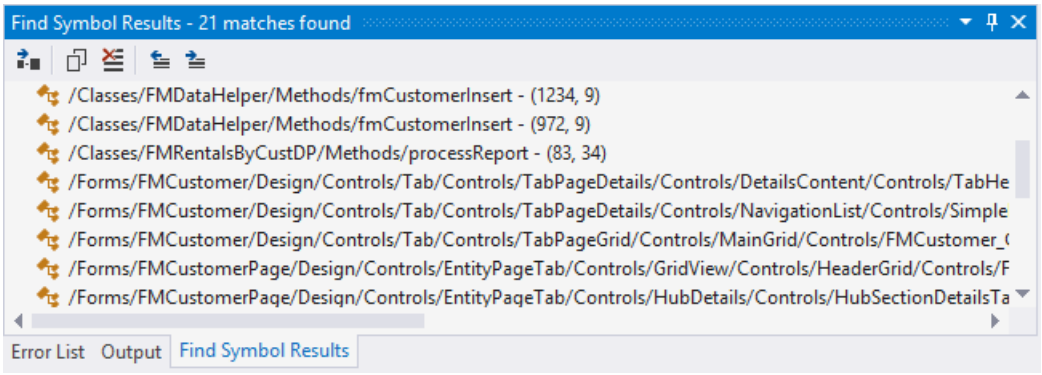
2/18/2021 • 2 minutes to read • Edit Online

This article reviews the commands that have been added to the Microsoft Visual studio Tools to help you determine how elements are used in an application.

Because of the large number of elements in a typical application, commands have been added to the Microsoft Visual Studio Tools to make it easier to determine how an element is used.

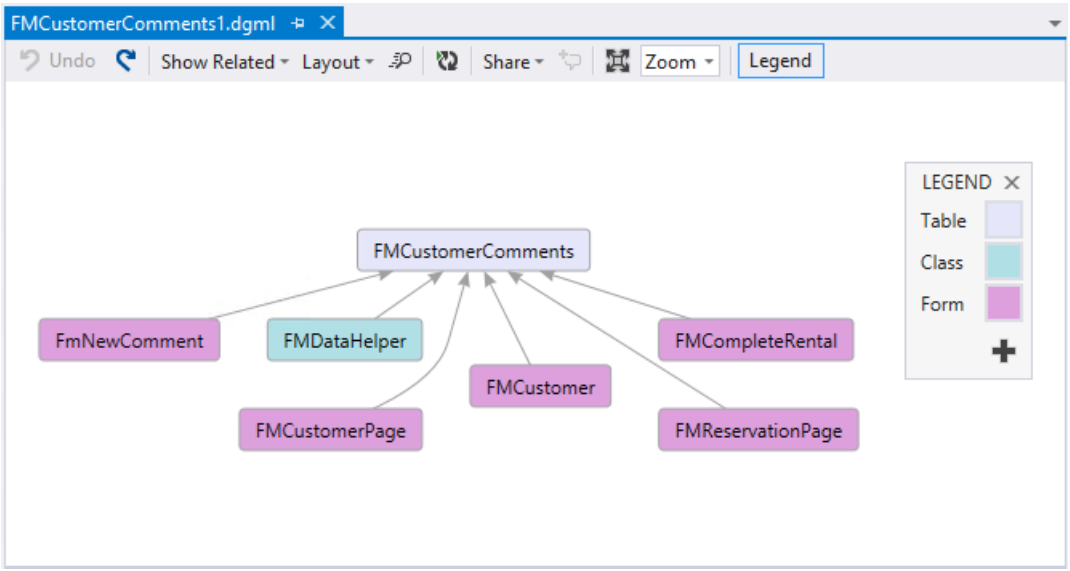
Finding where elements are used

During build operations, cross-reference information is generated that can be used to show how elements are used. You can right-click an element and then click **Find References** to display a list of the locations where that element is used. When you click one of the items in the list, the designer for the element opens.



Viewing a reference diagram

When you right-click some higher-level elements, such as tables, the **View Reference** command is available. This command produces a graphic that shows the elements that are related to the current element. You can right-click the items in the graphic and then click **Go To Definition** to navigate to those elements.



Additional resources

Development tools in Visual Studio

Element designers

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Export and import models

2/18/2021 • 2 minutes to read • Edit Online

Model files let you distribute models to customers and partners, and can be installed in development environments. They are key components of a Lifecycle Services (LCS) solution. Model files contain a model descriptor file, metadata, source code, and referenced .NET assemblies (when applicable). This article describes how to export a model into a model file, install a model file, and delete a model in a development environment.

Export a model into a model file for distribution

To export an existing model into a model file, use the ModelUtil.exe tool and the **-export** directive. This tool is located in the packages bin folder (typically, c:\packages\bin or i:\AosService\PackagesLocalDirectory\bin).

```
ModelUtil.exe -export -metadatastorepath=[path of the metadata store] -modelname=[name of the model to
export] -outputpath=[path of the folder where the model file should be saved]
```

Example

```
ModelUtil.exe -export -metadatastorepath=c:\packages -modelname="FleetManagement" -outputpath=c:\temp
```

The preceding example creates an .axmodel file under c:\temp. Typically, you then upload the model file to the Asset Library of the customer project or the Microsoft Dynamics Lifecycle Services (LCS) solution project.

Install a model in a development environment

To install a model file in a development environment, use the ModelUtil.exe tool and the **-import** directive.

```
ModelUtil.exe -import -metadatastorepath=[path of the metadata store where model should be imported] -file=
[full path of the file to import]
```

If the model already exists in your development environment, you must first delete it by using the **-delete** directive.

```
ModelUtil.exe -delete -metadatastorepath=[path of the metadata store] -modelname=[name of the model to
delete]
```

> **NOTE**
> If you're using an older version, you can use the -replace parameter to replace standard models (like Foundation) for overlayering.

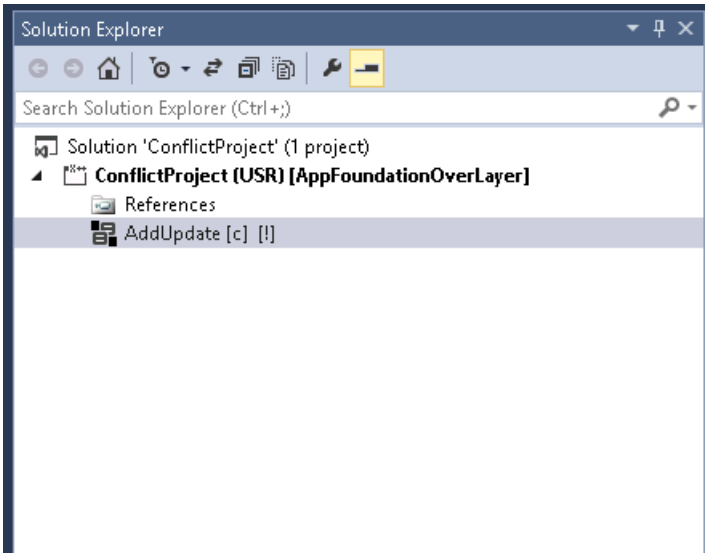
Resolve conflicts

If you install a model on a development environment that contains customizations to that model (in a higher-layer), you may have to resolve code or metadata conflicts. You can use the development tools to create a project that groups all items that have conflicts.

1. Under **Dynamics 365 > AddIns**, click **Create Project from Conflicts**.

2. In the dialog box, select the model to check for conflicts. This is the model that contains customizations to elements in the newly installed baseline model.

3. Click **Create project**. A project is generated that contains only the elements in that model that have conflicts.



4. Open the designer for the conflicting element to view and resolve conflicts by using the tools that are provided.

Metadata search in Visual Studio

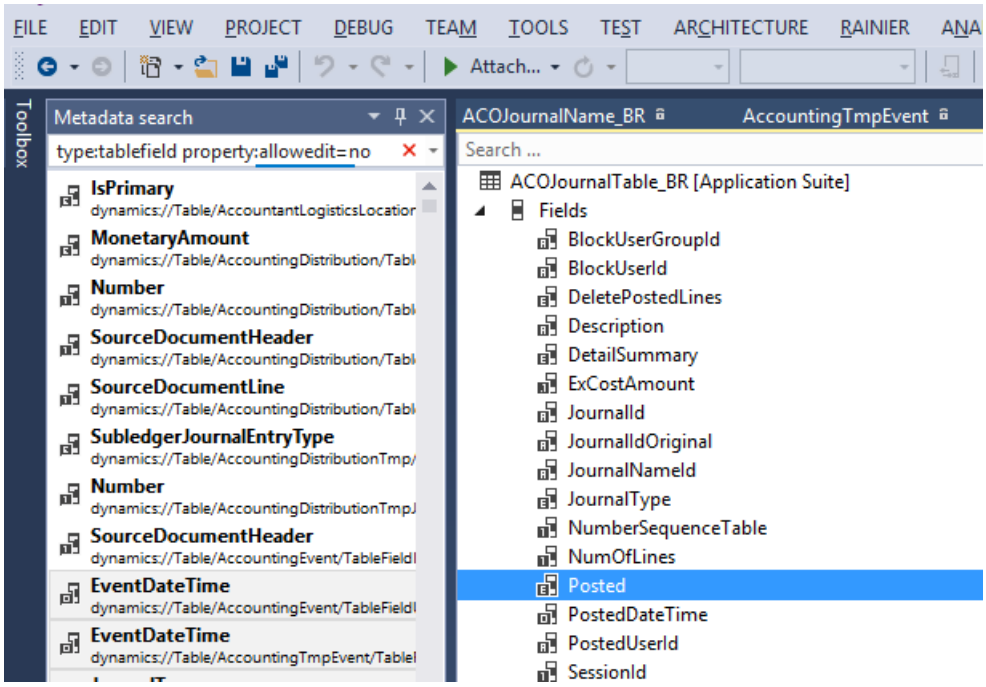
2/18/2021 • 3 minutes to read • Edit Online

This article describes how to use metadata search to search your code and metadata for arbitrary patterns and content.

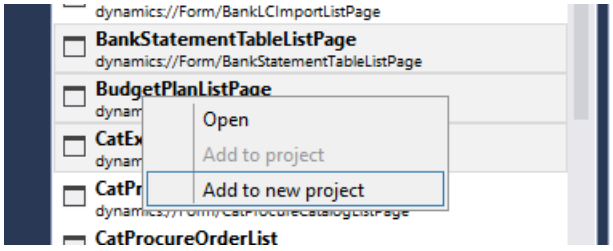
Given the large volume of the code base and metadata, it is often necessary to find things in the code that meet a certain criteria. For example, you might not know the name of the metadata element that contains the pattern or meets the criteria. Metadata search is exposed in Visual Studio through two user interfaces: the Metadata Search tool window and the Navigate To window.

Metadata search tool window

You can access the Metadata search tool window from the **Dynamics 365 > Metadata Search** menu command. Enter your search query to start the search. Results will start populating in the window asynchronously as you type. You can double-click any result line to navigate to the corresponding X++ code or metadata that matches your search query.



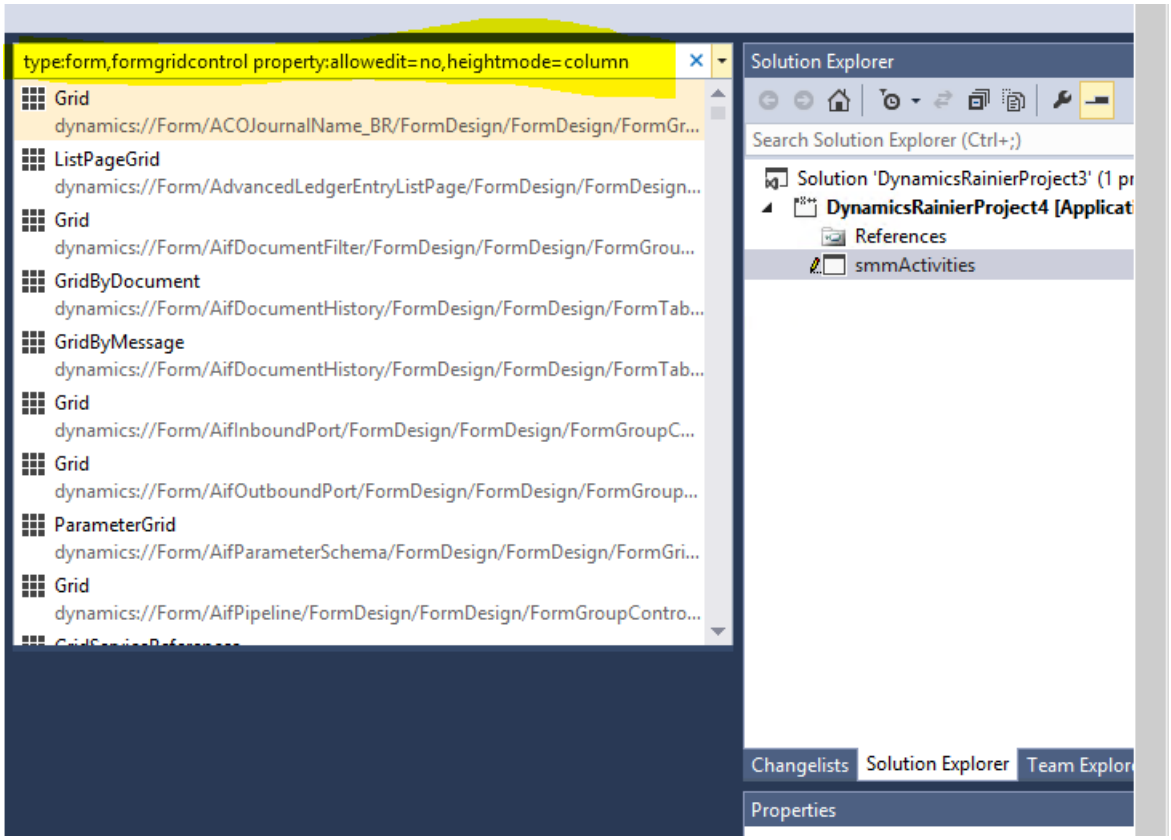
You can also select one or more results, right-click, and then add these elements to a project. You don't need to wait for the search to complete before you start interacting with the search results.



Navigate To window

The **Navigate To** window is invoked using the **Ctrl+',** (the comma character) shortcut keys. Pressing **Ctrl+',** displays the query entry box in top-right corner of the Visual Studio main document window. You can also

access the **Navigate To** window from the Visual Studio **Edit** menu. Enter you search query and see the results appear as you type. A progress indicator will stop when the search is complete. You don't need to wait for the search to complete to start interacting with the results.



Search query syntax

This section describes the search query syntax and provides example queries.

Syntax

The search query is a search string that consists of a set of filters in this general form:

```
<filter_1>:<filter_1_value> [<filter_2>:<filter_2_value> … [ <filter_N>:<filter_N\_value>]]
```

Where `<filter_i>` is one of the acceptable filter names, and < `filter_i_value>` are comma separated (and possibly quoted) filtering values.

Filter names

- **Name**: Filter by element name. This is the default filter, meaning if you just type a filter value, it is assumed to be an element name. Each comma-separated value is an acceptable element name.
- **Type**: Filter by element type. Each comma-separated value should be the name of an element or subelement type (root type or subtype) (that is, table, class, field). Logic of filtering is:

  `(roottype_1 OR roottype_2 OR … OR roottype_N) AND (subtype_1 OR subtype_2 OR … OR subtype_N)`
- **Model**: Filter by model name. Each comma-separated value should be the name of a model in your application.
- **Property**: Apply property filters. Each comma-separated value should be in the form `property_name=property_value`.
- **Code**: Filter using code snippets, use quotes around code snippets. The matching source code is the elements that contain the specified code snippet.

You can get help about using filter and filter syntax by opening the drop-down menu available in the search box.

[

]

Examples

QUERY STRING	WHAT IT DOES
`TrvExpTable`	If the token is by itself, it is assumed to be the name. So this will find everything in the application that has "TrvExpTable" in the name.
`type:form ccount`	Finds all forms that have "ccount" in their names.
`type:form property:formtemplate=listpage`	Finds all forms that contain the property "FormTemplate" equal to 'ListPage'.
`type:table,formDesign property:"WorkflowDataSource=TrvExpTable"`	Finds formDesign nodes under tables, nothing would be found.
`type:form,formmenufunctionbuttoncontrol property:Text=@SYS311998`	Finds all menu function button controls with the Text property equal to (a label) '@SYS311998'.
`type:table,method name:insert`	Finds tables with a method containing "insert" in the method name.
`type:table,tableindex name:Export`	Finds tables with an index name containing the word "Export".
`type:table,tableindexfield name:xpNum`	Finds table indexes with "xpNum" in the index field name.
`type:table,tablefieldgroup name:EPNew`	Finds FieldGroups (in tables) containing 'EPNew' in their names.
`type:form,formgridcontrol property:allowedit=no,heightmode=column`	Finds form grid controls, with properties **allowedit** equal to "no" and heightmode equal to "column".
`type:form,formtabcontrol property:arrangeMethod=Vertical,ViewEditMode=view,WidthMode=Auto`	Finds form tab controls, with properties arrangeMethod equal to "Vertical" and ViewEditMode equal to "view" and WidthMode equal to "Auto".
`type:form,formDesign property:"WorkflowDataSource=TrvExpTable"`	Finds all forms with the "WorkflowDataSource" property in the FormDesign node set to the value "TrvExpTable".
`model:"Application Suite" type:formdesign property:style=simplelistdetail`	Find all forms in Application Suite model that has the style property set to simpleListDetail in the FormDesign node.
`code:"return null"`	Finds all places in the source code that contains "return null".
`code:"element.lock()" type:form`	Finds all places in the forms source code that contain the snippet "element.lock()".

QUERY STRING	WHAT IT DOES
`code:"insert" type:table,form`	Finds all places in the source code of either forms or tables that contain "insert".
`code:"public display" type:form,method`	Finds all form methods that contain the code "public display".
`type:formbuttoncontrol property:text=`	Finds all form Button Controls that have **empty** text properties.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Models and packages

2/18/2021 • 4 minutes to read • Edit Online

This topic describes the concept of models and packages. It also explains how to use the development tools in Microsoft Visual Studio to create new models, how to update the parameters of existing models, and how to visualize dependencies between models.

To work with models in the model store, you use tools in Microsoft Visual Studio. You can create new models and change parameters for existing models.

Conceptual overview

A model is a group of elements, such as metadata and source files, that typically constitute a distributable software solution and includes customizations of an existing solution. A model is a design-time concept, for example a warehouse management model or a project accounting model. A model always belongs to a package. A package is a deployment and compilation unit of one or more models. It includes model metadata, binaries, and other associated resources. One or more packages can be packaged into a deployable package, which is the vehicle used for deployment on runtime environments.

Creating a new model

You use the **Create model** wizard to create new models. You can access this wizard from **Model Management** on the **Dynamics 365** menu. You can create two types of models:

- **A model that is deployed in its own package** – You can use this type of model to create new model elements, and extend the metadata and business logic of referenced models. The wizard lets you select the referenced models. This type of model is compiled into its own assembly and binaries, and will simplify and reduce the cost of upgrades, deployment, and application lifecycle management in general.
- **A model that is a part of an existing package** – You can use this type of model to perform advanced customizations, such as overlayering source code and metadata.

In the **Create model** wizard, select **usr** for the layer. This layer will store user customizations. If needed, you can patch your customizations using the **usp** layer. If there are multiple versions of the same object in different layers, then the top layer will take precedence and will be used.

When the **Create model** wizard is completed, if you chose to create a new project, you will be prompted to specify a name and location for it.

Updating model parameters

If you must change the parameters for a model, you can use the **Update model parameters** dialog box.

1. On the **Dynamics 365** menu, point to **Model Management**, and then click **Update model parameters**.
2. In the **Model name** field, select the model to update parameters for.
3. Update the parameters as you require.
4. Click **Next**.
5. Update the dependency information for the current model, if changes are required.
6. Click **Next**. The summary information for the model is displayed.
7. Click **Finish**.

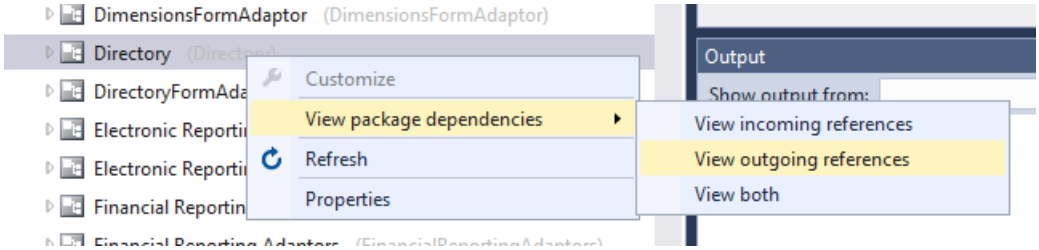
The updated model parameters become effective only after you restart Visual Studio.

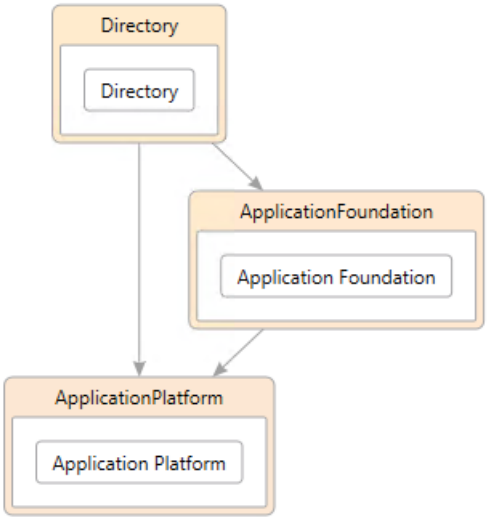
Viewing package dependencies

You can create a graphical representation that shows which packages and their models have dependencies on other packages. On the **Dynamics 365** menu, point to **Model Management**, and then click **View package dependencies**. A Directed Graph Markup Language (DGML) diagram will be generated for the current packages and their models. This diagram is a collection of interdependent nodes, each of which represents a package. Each node lists all the models that belong to that package. Additional tools let you enhance or simplify the diagram. For example, you can add comments, move nodes around, or remove nodes. You can also view package dependencies of a single model by following these steps:

1. Make sure the Application Explorer is in Model view: Right-click the AOT node and select **Model view**.
2. Right-click on any model and select **View package dependencies** > **View outgoing references.**

This will generate a graph of all packages that the selected model depends on.





Deleting a model

In a development or test environment, follow these steps to delete a model.

The following steps assume the local model store folder is C:\AOSService\PackagesLocalDirectory and your model is named MyModel1.

If your model belongs to its own package. For example an extension package with no other models in the package.

1. Stop the following services: AOS web service and Batch Management service.
2. Delete the package folder C:\AOSService\PackagesLocalDirectory\MyModel1.
3. Restart the services that were stopped in step 1.
4. If Visual Studio is running, refresh your models (**Visual Studio > Dynamics 365 > Model management > Refresh models**)
5. In Visual Studio, perform a full database synchronization (**Visual Studio > Dynamics 365 > Synchronize**

database).

If your model belongs to a package with multiple models. For example, the MyModel1 overlays Application Suite.

1. Stop the following services: AOS web service and Batch Management service.
2. Delete the model folder C:\AOSService\PackagesLocalDirectory<PackageName>\MyModel1. In this example. PackageName=ApplicationSuite.
3. Restart the services that were stopped in step 1.
4. In Visual Studio, refresh your models (**Visual Studio > Dynamics 365 > Model management > Refresh models**).
5. In Visual Studio, build the package that the deleted models belonged to (**Visual Studio > Dynamics 365 > Build models**).
6. In Visual Studio, perform a full database synchronization (**Visual Studio > Dynamics 365 > Synchronize database**).

Additional resources

Development tools in Visual Studio

Develop and customize home page

Export and import models

Finance and Operations project type in Visual Studio

2/18/2021 • 5 minutes to read • Edit Online

The Finance and Operations project type is part of the development tools. This project type resembles other projects in Visual Studio. It helps you organize and manage the elements that you're working with for a model. For example, the project can have folders that help you group the elements. A Visual Studio solution can contain multiple projects. There is one important constraint for a project: it can contain elements from only one model. If you must work with elements from different models, you must use multiple projects in your Visual Studio solution.

Create a new project

To create a new, empty project, follow these steps.

1. On the **File** menu, point to **New**, and then click **Project**.
2. In the list of template types, expand the **Installed** node.
3. Expand the **Templates** node.
4. Select the **Finance and Operations** category.
5. Select the **Operations Project** template.
6. Enter the name and location for the new project.
7. Specify whether you want to create a new solution or add the project to the current solution.
8. Click **OK**.

Every project has several important properties. To set the properties for a project, right-click the project in Solution Explorer, and then click **Properties**. The following table describes these properties.

PROPERTY	DESCRIPTION
Startup Object type	The type of object that will be used as the Startup Object when the project is run. The following types are available: Form Class Output menu item
Startup Object	The object that will be invoked when the project is run.
Company	The default company that will be used when the project is run.
Partition	The partition that will be used when the project is run.
Project File	The name of the file that contains information about the project.
Project Folder	The location of the project.
Model	The model that the project is associated with. All elements in the project must be in the selected model.

PROPERTY	DESCRIPTION
Model Publisher	A read-only value that indicates the publisher of the model.
Layer	A read-only value that indicates the application layer that the model is located in.
Synchronize database on build	A value that indicates whether the synchronize operation for tables will be performed when the build action is performed for the project.

Of these properties, the **Model** property is particularly important. You must specify which model the project is associated with. All the elements that you create or add to the project must be part of this model. The **Startup Object type** and **Startup Object** properties are useful when you test and debug your application. When you start your project (by pressing F5 for debugging or Ctrl+F5 for no debugging), the specified form will be loaded, or the **main()** method from the specified class will be run. The method must have the following signature:
public static void main(Args _args)

Add elements to a project

There are several ways to add existing elements to a project. Here are the most typical:

- After you select the project in Solution Explorer, you can find the element in Application Explorer, right-click it, and then click **Add to Project**. This is the simplest method.
- You can use drag-and-drop operations to add an element from Application Explorer to a project.
- If you're limiting your search results to a single model, you can add the results of a filter in Application Explorer to a project.

After you've added the elements, you might want to use Solution Explorer to group them into folders, so that they are easier to find. The location of the project file and the folders that you create in the project don't affect the location of the XML files that represent the model elements. The model elements are always stored in the appropriate folder in the model store. To organize elements into folders, select the **Organize projects by element type** option. On the **Dynamics 365** menu, click **Options**. Select the **Projects** category to see this option. When this option is selected, elements that are added to a new or existing project (such as when search results in Application Explorer are added to a project) are grouped into folders, based on the element type name. To create a new element for a project, follow these steps.

1. In Solution Explorer, right-click the project, point to **Add**, and then click **New Item**.
2. In the **Operations Artifacts** list, select the category of element to create.
3. Select the specific element type.
4. Enter a name for the element.
5. Click **Add**. The element will be added to the project. It will also be added to the model in the model store that the project is associated with.

After you've added the new element, you might want to use Solution Explorer to move it into a folder in the project, so that it's easier to find.

Export a projects as an .axpp file

To transfer elements to a different installation, you can use a project package file. Project package files have the .axpp file name extension. A project package contains all the elements from the project. To export a project, follow these steps.

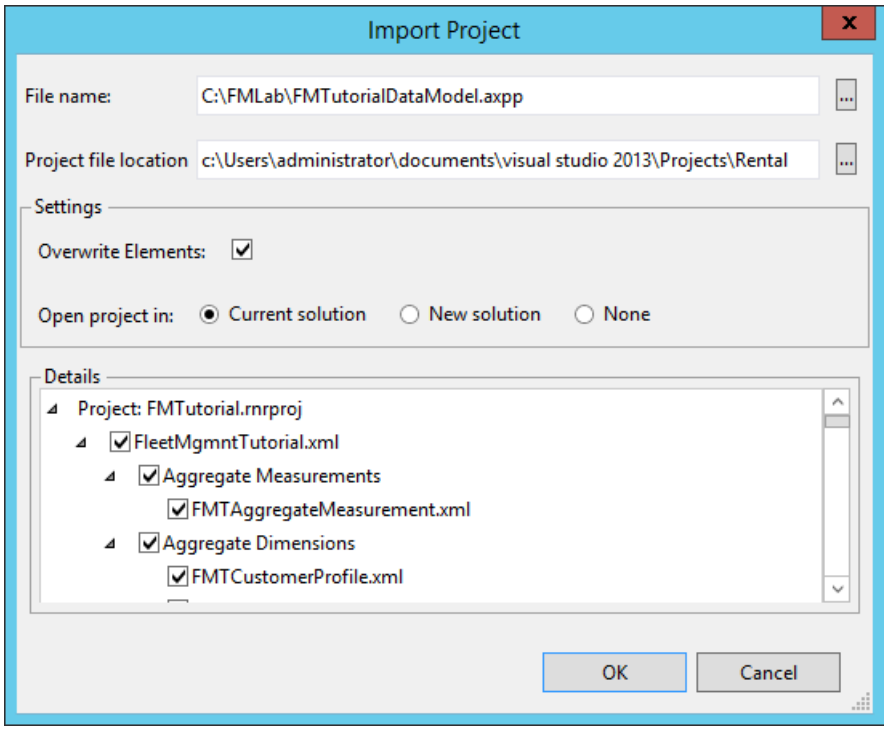
1. In Solution Explorer, select the project to export.

2. On the **Project** menu, click **Export Project**. (The command on the menu will contain the name of the selected project.)
3. Enter a name for the project package file, and select a location.
4. Click **Save**.

Import an .axpp file

To use the contents of a project package file, you must import the .axpp file into an installation. The elements from the project package file will be imported into the same model that they were exported from. If that model doesn't exist in the installation, it will be created during the import process. To import a project package file, follow these steps.

1. On the **Dynamics 365** menu, click **Import Project**.
2. In the **Import Project** dialog box, specify the location of the project package (.axpp) file to import.
3. If you want elements from the project package file to overwrite any existing elements, select **Overwrite Elements**.
4. Specify whether you want to open the project in the current selection, in a new solution, or not at all.
5. In the **Details** field, review the elements that will be imported. You can clear the check box next to any elements that you don't want to import.



6. Click **OK** to complete the import process.

Tools add-ins for Visual Studio

2/18/2021 • 2 minutes to read • Edit Online

This topic reviews the Add-ins infrastructure that has been added to Microsoft Visual Studio, so that developers can more easily add tools for development.

A lot of great tools have been added to Microsoft Visual Studio to support development. However, there will always be additional tools to meet specific requirements. To make it easier to add these additional tools, an **Add-ins** infrastructure has been provided for developers. The additional tools are available in two places:

- The **Add-ins** submenu on the **Dynamics 365** menu
- The **Add-ins** submenu on the shortcut menu in the element designer

To make it easier to create your own add-ins, you can select the **Dynamics Developer Tools Add-in** project type when you create a new project in Visual Studio. This project type has the infrastructure that is required to implement an add-in.

For more information on add-ins, see:

- Visual Studio add-ins that support form patterns

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Update the Visual Studio development tools

2/18/2021 • 2 minutes to read • Edit Online

This topic explains how to update the development tools.

Use this tutorial to update your Visual Studio development tools with a new version. It explains how to uninstall your existing Visual Studio development tools and install the new extension. The new extension is in the form of an installable VSIX file. This file is a part of the binary hotfix available on the Dynamics Lifecycle Services (LCS) site. The VSIX file is located in the **DevToolsService\Scripts** folder of the binary hotfix package.

> **NOTE**
> You do not need to follow the instructions in this article if you are upgrading your Finance and Operations platform to Platform update 4 or newer. It is an automatic step that is part of the platform upgrade process.

Uninstall the existing Visual Studio extension

In order to install a new version of the development tools, you'll need to uninstall the existing version first. Verify the version of the development tools that you have installed. If you don't have it installed, you can skip this section.

Verify your current version of the Visual Studio extension

1. Open the Visual Studio **Help > About Microsoft Visual Studio** dialog and find **Finance and Operations Developer Tools**.
2. Select it and click **OK**.

Uninstall the extension

1. Open the Visual Studio **Tools > Extensions and Updates** dialog.
2. Select **Finance and Operations Visual Studio Tools** and click **Uninstall**.
3. When the extension is uninstalled, exit Visual Studio.

Install a new version of the extension

1. Make sure Visual Studio is not running.
2. Double-click (or right-click and **Open**) the VSIX file of the new version.
3. Follow the installation instructions.
4. When installation is complete, you can start Visual Studio and start developing your application.

> **NOTE**
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

X++ and debugger features

2/18/2021 • 28 minutes to read • Edit Online

This tutorial is for developers to use advanced constructs of the X++ language and take advantage of productive debugger features. This is a walkthrough of the new features with exercises included to practice using these features.

In previous versions, the X++ code was compiled into pseudo-code, or p-code, that was interpreted on the server or on the client application. This code was then subject to further compilation into CIL. Today the story is much simpler--only CIL is supported, and this code is generated from a new compiler. In this tutorial, we'll be discussing some of the new features that have been added to the X++ language. As we run through the scenarios, you'll also see some of the new features in the debugger.

Declare anywhere

Previously, all local variables had to be placed at the start of the method in which they're used. Now, you have fine-grained control over the scope of your variables. With this new feature, it's possible to provide smaller scopes for variables, outside of which the variables can't be referenced. The lifetime of the variable is the scope in which it's declared. Scopes can be started at the block level (inside compound statements), in for statements, and in using statements as we will see below. There are several advantages to making scopes small.

- Readability is enhanced.
- You can reduce the risk of reusing a variable inappropriately during long-term maintenance of the code.
- Refactoring becomes much easier. You can copy code in without having to worry about variables being reused in contexts they shouldn't.

Example - declare a loop counter

In this example, we declare the loop counter inside the 'for' statement in which it's used.

```
void MyMethod()
{
    for (int i = 0; i < 10; i++)
    {
        info(strfmt("i is %1", i));
    }
}
```

The scope of the variable is the for statement itself, including the condition expression and the loop update parts. The value can't be used outside this scope. If you attempt to do that, you will get the following.

```
void MyMethod()
{
    for (int i = 0; i < 10; i++)
    {
        if (i == 7)
        {
            break;
        }
    }
    info(strfmt("Found: %1", i));
}
```

The compiler will issue an error message in the info call: 'i' is not declared.

Example - declare in a using statement

There's another place where scopes can be established: the `using` statement, which is another newcomer to the X++ language.

```
static void AnotherMethod()
{
    str textFromFile;

    using (System.IO.StreamReader sr = new System.IO.StreamReader("c:\\test.txt"))
    {
        textFromFile = sr.ReadToEnd();
    }
}
```

As a rule, when you use an **IDisposable** object, you should declare and instantiate it in a using statement. The using statement calls the **Dispose** method on the object in the correct way, even if an exception occurs while you are calling methods on the object. You can achieve the same result by putting the object inside a try block, and then explicitly calling Dispose in a finally block; in fact, this is how the using statement is translated by the compiler. Declarations can now be provided anywhere statements can be provided-- a declaration is syntactically a statement, a declaration statement. You can, therefore, provide declarations immediately prior to the usage. You don't have to declare the variables all in one place.

Example with new features

The following sample shows some of the features described above.

```
// loop variable declared within the loop: It will
// not be misused outside the loop
for(int i = 1; i < 10; i++)
{
    // Because this value is not used from outside the loop,
    // its declaration belongs in this smaller scope.
    str s = int2str(i);
    info(s);
}
```

To avoid confusion, the X++ compiler will issue an error if you attempt to introduce a variable that would hide another variable in an enclosing scope or even in the same scope. For instance, the following code will cause the compiler to issue the following diagnostic message: A local variable named 'i' cannot be declared in this scope because it would give a different meaning to 'i', which is already used in a parent or current scope to denote something else.

```
{
    int i;
    {
        int i;
    }
}
```

This aligns well with the rules that are known from C#, but is different from the rule in C++ where shadowing is not diagnosed.

Adapt code to use a smaller scope

Adapt the code in **FMVehicleInventoryServiceClass** to use smaller scopes.

Static constructors and static fields

Static constructors and static fields are new features in the X++ language. Static constructors are guaranteed to run before any static or instance calls are made to the class. In C#, the concept of static relates to the whole executing application domain. The execution of the static constructor is relative to the user's session. The static constructor has the following profile.

```
static void TypeNew()
```

You'll never call the static constructor explicitly; the compiler will generate code to ensure that the constructor is called exactly once prior to any other method on the class. A static constructor is used to initialize any static data, or to perform a particular action that needs to be performed only once. No parameters can be provided for the static constructor, and it must be marked as static. Static fields are fields that are declared using the static keyword. Conceptually they apply to the class, not instances of the class.

Implement a singleton

We'll show how a singleton, called instance in the example below, can be created by using the static constructor.

```
public class Singleton
{
    private static Singleton instance;

    private void new()
    {
    }

    static void TypeNew()
    {
        instance = new Singleton();
    }

    public static Singleton instance()
    {
        return Singleton::instance;
    }
}
```

The singleton will guarantee that only one instance of the class will ever be called, which is consumed by the following.

```
{
    // Your code here.
    Singleton i = Singleton::instance();
}
```

Assignment of field members inline

It's now possible to assign a value to a field inline, i.e. along with the declaration of the field itself. This applies to both static and instance fields. In the following code, the values of field1 and field2 are defined in this fashion.

```
public class MyClass2
{
    int field1 = 1;
    str field2 = "Banana";

    void new()
    {
        // Your code here.
    }
}
```

The code above has the same semantic meaning as:

```
public class MyClass2
{
    int field1;
    str field2;

    void new()
    {
        this.field1 = 1;
        this.field2 = "Banana";
        // Your code here.
    }
}
```

The inline assignments work for both static and instance members.

Consts and Readonly

The concept of macros continues to be fully supported in X++. However, using constants instead of #defines has a number of benefits.

- You can add a documentation comment to the const, not to the value of the macro. Ultimately, the language service will pick this up and provide good information to the user.
- The const is known by IntelliSense.
- The const is cross referenced, so you can find all references of a particular constant. This is not the case for a macro.
- The const is subject to access modifiers, either private, protected, or public. The accessibility of macros is not well understood or even rigorously defined.
- Consts have scope, while macros do not.
- You can see the value of consts and readonly variables in the debugger.

Macros that are defined in class scopes (in class declarations) are effectively available in all methods of all derived classes. This was originally a bug in the legacy compiler macro implementation, but this loophole is now massively exploited by application programmers. The new X++ compiler still honors this, but no new code that uses this should be written. This particular feature also considerably impacts compiler performance. Constants can be declared at the class level as suggested below.

```
private const str MyConstant = 'SomeValue';
```

The constants can then be referenced by using the double-colon syntax.

```
str value = MyClass::MyConstant;
```

If you're in the scope of the class where the const is defined, you can omit the type name prefix (MyClass in the example above). You can easily implement the concept of a macro library this way. The list of macro symbols becomes a class with public const definitions.

Example - macro definitions

The fleet application contains the **FMDataHelper** class that contains the following macro definitions.

```
public class FMDataHelper
{
    #define.FMSvcTechUserId('FMSvcTec')
    #define.FMClerkUserId('FMClerk')
    #define.FMManagerUserId('FMMgr')
    #define.FMSvcTechUserGrpId('FMSvcTech')
    #define.FMClerkUserGrpId('FMClerk')
    #define.FMManagerUserGrpId('FMManager')
}
```

Change these to const definitions and update the places where the macros are used accordingly.

You can also define consts solely as variables. The compiler will maintain the invariant that the value can't be modified.

```
{
    const int Blue = 0x0000FF;
    const int Green = 0x00FF00;
    const int Red = 0xFF0000;
}
```

Read-only fields can only be assigned a value once, and that value never changes; the field can be assigned its value either inline, at the place where the field is declared, or in the constructor. Currently, that's the only difference between const and read-only.

Var

You can now declare a variable without explicitly providing the type of the variable, if the compiler can determine the type from the initialization expression. Note that the variable is still strongly-typed into one, unambiguous type. It's only possible to use var on declarations where initialization expressions are provided (from which the compiler will infer the type). There are situations where this can make code easier to read, but this feature shouldn't be misused. You should consider the following rules:

- Use var to declare local variables when the type of the variable is obvious from the right side of the assignment, or when the precise type is not important.

  ```
  // When the type of a variable is clear from the context, use var
  // in the declaration.
  var var1 = "This is clearly a string.";
  var var2 = 27; // This is an integer (not a real).
  var i = System.Convert::ToInt32(3.4);
  ```

- Don't use var when the type isn't apparent from the initialization expression.

  ```
  // When the type of a variable is not clear from the context, use an
  // explicit type.
  int var4 = myObject.ResultSoFar();
  ```

- Use var for the declarations of for loop counters.

- Use var for disposable objects inside using statements.

Private and protected member variables

Previously, all member variables defined in a class were invariably protected. It's now possible to make the visibility of member variables explicit by adding the private, protected, and public keywords. The interpretation of these modifiers is obvious and aligns with the semantics for methods:

- A private member can only be used within the class where it's defined.
- A protected member can be used in the class where it's defined, and all subclasses thereof.
- A public member can be used anywhere: it's visible outside the confines of the class hierarchy in which it's defined.

The default for member variables that aren't adorned with an explicit modifier is still protected. You should make it a habit of explicitly specifying the visibility. As described, when a member variable is defined as public, it may be consumed outside of the class in which it's defined. In this case, a qualifier designating the object hosting the variable has to be specified, using the dot notation (as is the case for method calls). Reusing the code from above:

```
public class MyClass2
{
    int field1;
    str field2;

    void new()
    {
        this.field1 = 1;   // Explicit object designated
        field2 = "Banana";  // 'this' assumed, as usual
    }
}
```

In this case, field1 is accessed using the explicit 'this.' qualifier.

> **NOTE**
> Making a member variable public may not be a good idea since it exposes the internal workings of the class to its consumers, creating a strong dependency between the class implementation and its consumers. You should always strive to only depend on a contract, not an implementation.

Finally in try/catch statements

Try/catch statements can now include an optional finally clause. The semantics are the same as they are in C# and other managed languages. The statements in the finally clause are executed when control leaves the try block, either normally or through an exception.

```
try
{
    // ...
}
catch
{
    // Executes when any exception is thrown in the dynamic
    // scope in the try block.
}
finally
{
    // Executed irrespective of how the try block exits.
}
```

Event handlers and Pre/Post methods

In legacy X++, it was possible to prescribe in metadata that certain methods were to be executed prior to and after the execution of a method. The information about what subscribes call was recorded on the publisher, which isn't useful in the environment. It's now possible to provide Pre and Post handlers through code, by providing the SubscribesTo attribute on the subscribers.

Example of pre and post methods

```
[PreHandlerFor(classStr(MyClass2), methodstr(MyClass2, publisher))]
public static void PreHandler(XppPrePostArgs arguments)
{
    int arg = arguments.getArg("i");
}

[PostHandlerFor(classStr(MyClass2), methodstr(MyClass2, publisher))]
public static void PostHandler(XppPrePostArgs arguments)
{
    int arg = arguments.getArg("i");
    int retvalFromMethod = arguments.getReturnValue();
}

public int Publisher(int i)
{
    return 1;
}
```

This example shows a publishing method called Publisher. Two subscribers are enlisted with the PreHandlerFor and PostHandlerFor. The code shows how to access the variables, and the return values.

This feature is provided for backward compatibility and, because the application code doesn't have many delegates, to publish important application events. Pre and Post handlers can easily break as the result of added or removed parameters, changed parameter types, or because methods are no longer called, or called under different circumstances. Attributes are also used for binding event handlers to delegates:

```
[SubscribesTo(
    classstr(FMRentalCheckoutProcessor),
    delegatestr(FMRentalCheckoutProcessor, RentalTransactionAboutTobeFinalizedEvent))]
public static void RentalFinalizedEventHandler(
    FMRental rentalrecord, Struct rentalConfirmation)
{
}

    delegate void RentalTransactionAboutTobeFinalizedEvent(
        FMRental fmrentalrecord, struct RentalConfirmation)
{
}
```

In this case, the SubscribesTo attribute specifies that the method RentalFinalizedEventHandler should be called when the FmRentalCheckoutProcessor.RentalTransactionAboutToBeFinalizedEvent delegate is called. Since the binding between the publisher and subscribers is done through attributes, there's no way of specifying the sequence in which subscribers are called.

Extension methods

The extension method feature lets you add extension methods to a target class by writing the methods in a separate extension class. The following rules apply:

- The extension class must be static.
- The name of the extension class must end with the ten-character suffix _Extension. However, there's no restriction on the part of the name that precedes the suffix.
- Every extension method in the extension class must be declared as public static.
- The first parameter in every extension method is the type that the extension method extends. However, when the extension method is called, the caller must not pass in anything for the first parameter. Instead, the system automatically passes in the required object for the first parameter.

It's perfectly valid to have private or protected static methods in an extension class. These are typically used for implementation details and are not exposed as extensions. The example below illustrates an extension class holding a few extension methods:

```
public static class AtlInventLocation_Extension
{
    public static InventLocation refillEnabled(
        InventLocation _warehouse,
        boolean _isRefillEnabled = true)
    {
        _warehouse.ReqRefill = _isRefillEnabled;
        return _warehouse;
    }

    public static InventLocation save(InventLocation _warehouse)
    {
        _warehouse.write();
        return _warehouse;
    }
}
```

Reasons to use extension methods

The extension method technique doesn't affect the source code of the class it extends. Therefore, the addition to the class can be done without over-layering. Upgrades to the target class are never affected by any existing extension methods. However, if an upgrade to the target class adds a method that has the same name as your extension method, your extension method becomes unreachable through objects of the target class. Extension

methods are easy to use. The extension method technique uses the same dot-delimited syntax that you routinely use the call regular instance methods. Extension methods can access all public artifacts of the target class, but they can't access things that are protected or private. In this way, extension methods can be seen as a kind of syntactic sugar.

Where can extension methods be applied

The target of an extension method must be one of the following application object types:

- Class
- Table
- View
- Map

Regardless of the target type, an extension *class* is used to add extension methods to the type. For example, an extension table is *not* used to add methods to a table, and there's no such thing as an extension table.

Using clauses

Previously, all references to managed artifacts that weren't authored in X++ was done using fully qualified names, including the namespace for each type. This is still possible, but you can now provide using clauses to make the use of such artifacts less onerous. As opposed to a using statement, each using clause precedes the class in which the clause is applied. It's also possible to provide aliases that introduce a short name for a fully qualified name. Aliases can denote namespaces and classes as shown below.

Example of using clause

Consider the following code:

```
using System;
using IONS=System.IO; // Namespace alias
using Alist=System.Collections.ArrayList; // Class alias

public class MyClass2
{
    public static void Main(Args a)
    {
        Int32 I; // Alternative to System.Int32
        Alist al; // Using a class alias

        al = new Alist();
        str s;

        al.Add(1);

        s = IONS.Path::ChangeExtension(@"c:\tmp\test.xml", ".txt");
    }
}
```

Differences between legacy X++ and new X++

In this section, we'll see some of the differences between legacy X++ and the new X++.

Reals are Decimals

The type used to represent real values has changed from interpreted X++. This won't require you to rewrite any code, because the new type can express all of the values that the old one could. We provide this material in the interest of full disclosure only. All instances of the real type are now implemented as instances of the .NET decimal type (System.Decimal). Just as the real type in previous versions, the decimal type in a binary coded decimal type that, unlike floating point type, is resilient to rounding errors. The range and resolution of the

decimal type are different from the original types. The original X++ real type supported 16 digits and an exponent that defined where the decimal point is placed. The new X++ real type can represent decimal numbers ranging from positive 79,228,162,514,264,337,593,543,950,335 (2^{96}-1) to negative 79,228,162,514,264,337,593,543,950,335 (-(2^{96}-1)). The new real type doesn't eliminate the need for rounding. For example, the following code produces a result of 0.9999999999999999999999999999 instead of 1. This is readily seen when using the debugger to show the value of the variables below.

```
public class MyClass2
{
    public static void Main(Args a)
    {
        real dividend = 1.0;
        real divisor = 3.0;
        str stringvalue;
        System.Decimal valueAsDecimal;

        real value = dividend/divisor * divisor;

        valueAsDecimal = value;
        info(valueAsDecimal.ToString("G28"));
    }
}
```

No number of decimals will suffice to represent the value of 1/3 accurately. The discrepancy obtained here is due to the fact that only a finite number of decimals are provided. You should use the Round function to round to the number of decimals required.

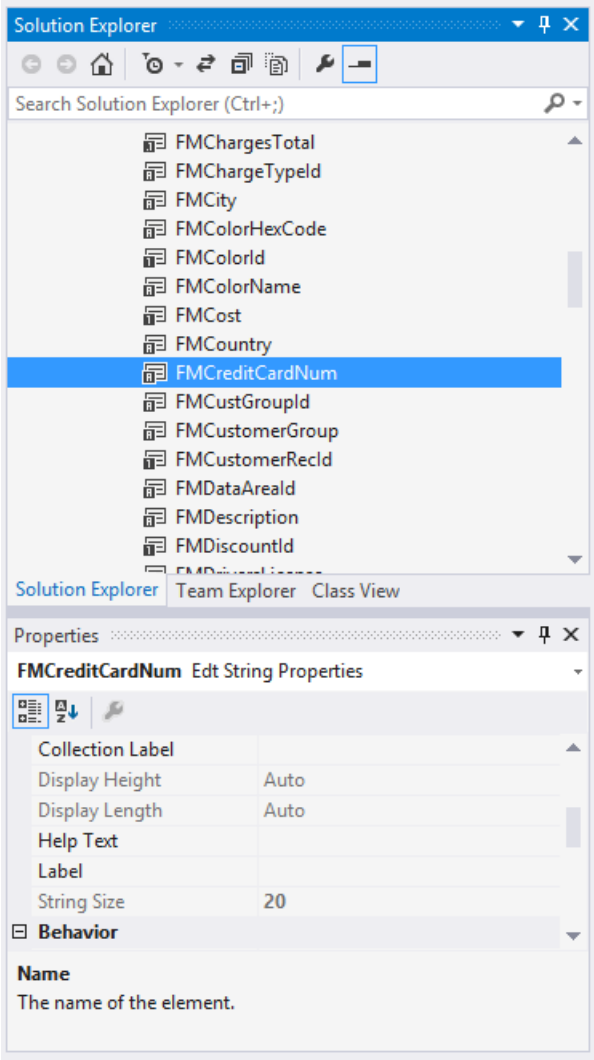
```
value = round(value, 2);
```

Internal representation

A decimal number is a floating-point value that consists of a sign, a numeric value where each digit in the value ranges from 0 to 9, and a scaling factor that indicates the position of a floating decimal point that separates the integral and fractional parts of the numeric value. The binary representation of a real value consists of a 1-bit sign, a 96-bit integer number, and a scaling factor used to divide the 96-bit integer and specify what portion of it is a decimal fraction. The scaling factor is implicitly the number 10, raised to an exponent ranging from 0 to 28. Therefore, the binary representation of a decimal value represents ((-2^{96} to 2^{96})/10(0\ to\ 28)), where -(2^{96}-1) is equal to the minimum value and 2^{96}-1 is equal to the maximum value that can be expressed.

String truncation

String truncation is not a new feature. However, when code is executed in IL in previous versions, the automatic string truncation described here doesn’t take place. String values can be declared in X++ to contain a maximum number of characters. Typically, this is achieved by encoding this information in an extended data type, as shown below: Credit card numbers cannot exceed 20 characters.



It's also possible to specify length constraints directly in the X++ syntax:

```
str 20 creditCardNumber;
```

All assignments to these values are implicitly truncated to this maximum length.

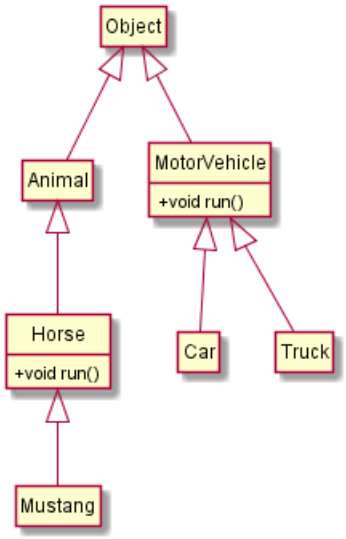
Exercise

Run the following code in the debugger by including it in a static main method:

```
creditCardNumber = "12345678901234567890Excess string";
```

Casting

The previous version of X++ was very permissive in its treatment of type casting. Both up-casting and down-casting were allowed without intervention from the programmer. Some of the casting permitted in legacy X++ can't be implemented in the confines of the .NET runtime environment. In object oriented programming languages, including X++, casting refers to assignments between variables whose declared types are both in the same inheritance chain. A cast is either a down-cast or an up-cast. To set the stage for this discussion, we introduce a few self-explanatory class hierarchies:



As you can see, the MotorVehicle class isn't related to the Animal class. An **up-cast** happens when assigning an expression of a derived type to a base type:

```
Animal a = new Horse();
```

A **down-cast** happens when assigning an expression of a base type to a derived variable.

```
Horse h = new Animal();
```

Both up-casts and down-casts are supported in X++. However, down-casts are dangerous and should be avoided whenever possible. The example above will fail with an InvalidCastException at runtime, since the assignment doesn't make sense. X++ supports late binding on a few types, like object and formrun. This means that the compiler won't diagnose any errors at compile-time when it sees a method being called on those types, if that method isn't declared explicitly on the type. It's assumed that the developer knows what they're doing. For instance, the following code may be found in a form.

```
Object o = element.args().caller();
    o.MyMethod(3.14, “Banana”);
```

The compiler can't check the parameters, return values, etc. for the MyMethod method, since this method isn't declared on the object class. At runtime, the call will be made using reflection, which is orders of magnitude slower than normal calls. Note that calls to methods that are actually defined on the late binding types will be naturally checked. For example, the call to ToString():

```
o.ToString(45);
```

will cause a compilation error:

```
'Object.toString' expects 0 argument(s), but 1 specified.
```

because the ToString method is defined on the object class. There's one difference from the implementation of previous version of X++, related to the fact that methods could be called on unrelated objects, as long as the name of the method was correct, even if the parameter profiles weren't entirely correct. This isn't supported in CIL.

Example - casting

```
public class MyClass2
{
    public static void Main(Args a)
    {
        Object obj = new Car();
        Horse horse = obj; // exception now thrown
        horse.run();    // Used to call car.run()!
    }
}
```

You should use the IS and AS operators liberally in your code. The IS operator can be used if the expression provided is of a particular type (including derived types); the AS operator will perform casting into the given type and return null if a cast isn't possible.

Compiler diagnoses attempts to store objects in containers

In previous incarnations of the X++ compiler, it was possible to store object references into containers, even though this would fail at runtime. This is no longer possible. When the compiler sees an attempt to store an object reference into a container:

```
container c = [new Query()];
```

It will issue the error message:

```
Instances of type 'Query' cannot be added to a container.
```

If the type of the element that is added to the container is any type that the compiler can't make the determination of whether the value is a reference type. The compiler will allow this under the assumption that the user knows what they're doing. The compiler won't diagnose the following code as erroneous but an error will be thrown at runtime.:

```
anytype a = new Query();
    container c = [a];
```

Cross company clause can contain arbitrary expressions

The cross company clause can be used on select statements to indicate the companies that the search statement should take into account. The syntax has been enhanced to allow arbitrary expressions (of type container) instead of a single identifier, which is a variable of type container.

```
private void SampleMethod()
{
    MyTable t;
    container mycompanies = ['dat', 'dmo'];
    select crosscompany: mycompanies t;
}
```

Now, it's possible to provide the expression without having to use a variable for this purpose.

```
private void SampleMethod()
{
    MyTable t;
    container mycompanies = ['dat', 'dmo'];
    select crosscompany: (['dat'] + ['dmo']) t;
}
```

mkDate predefined function no longer accepts shorthand values

In legacy systems, it was possible to use "shorthand" values for the year argument of the mkDate function. The effect can be seen in the following code sample.

```
static void Job16(Args _args)
{
    int y;
    date d;

    for (y = 0; y < 150; y++)
    {
        d = mkDate(1,1,y);
        info(strFmt("%1 - %2", y, year(d)));
    }
}
```

Running this code in the legacy system will produce the following values: 0 – 2000 1 – 2001 2 – 2002 ... 27 – 2027 28 – 2028 29 – 2029 **30 – 2030 31 – 1931** 32 – 1932 33 – 1933 ... 97 – 1997 98 – 1998 **99 – 1999 100 – 1900** We no longer support these values. Attempts to use such values will cause the mkDate function to return the null date (1/1/1900).

Obsolete statement types

The following statements are no longer supported.

Pause and Window statements

The X++ pause statement is no longer supported because the pop-up **Print Window** that it affected has been removed. The pause and window statements were mainly used for debugging within the MorphX development environment, which was the same as the execution environment. Since the two are now separated, with Visual Studio taking the place of the MorphX environment, these statements are no longer relevant.

Print statement

The X++ print statement is another statement that existed only for debugging purposes. It still exists, and its basic idea is unchanged. But print now outputs through System.Diagnostics.WriteLine. The product configuration determines the detail of the written information is sent. You may find that using the Infolog is more compelling, since its output appears in the debugger and the running form.

The Ignore list

Since the legacy environment was all interpreted, it was possible to have some parts not compile, and use the rest. As long as you only called methods that compiled correctly, you were fine; however, you would run into trouble if you tried to call methods that weren't successfully compiled. This way of working doesn't work in CIL. Assemblies are generated from successful compilations and the runtime system can't load incomplete assemblies. However, there are legitimate scenarios when porting legacy applications into the new environment where it's beneficial to get things running in a staged fashion and where parts of the application need to be tested before everything is ported. While this is useful for this very limited scenario, it shouldn't be used once the application is ready for production, since you would be hiding problems that will occur at runtime, after the

system has been deployed. This is how it currently works: You can specify a method in an XML by selecting, "Edit Best Practice Suppressions," from the context menu on the project. This will open an XML document where the exclusions are maintained.

New Debugger features

This section provides information about the new features that we've added to the debugging experience in Visual Studio.

Adding ToString methods to your classes

It's often a benefit to add ToString() methods to your classes. The effort spent doing this comes back many times and it's easy to do. This advice also holds true for legacy X++.

> **NOTE**
> Since ToString methods can be called at unpredictable times, it isn't a good idea to change the state of the object here.

Identifying unselected fields

It's a common source of bugs to use fields from a table when these fields don't appear in the field list in a select statement. Such fields will have a default value according to their type. It's now possible in the debugger to see if a value has been selected or not.

Using breakpoints

Consider the following code:

```
class MyClass
{
    public static void Main(Args a)
    {
        FMRental rental;

        select EndMileage, RentalId from rental;

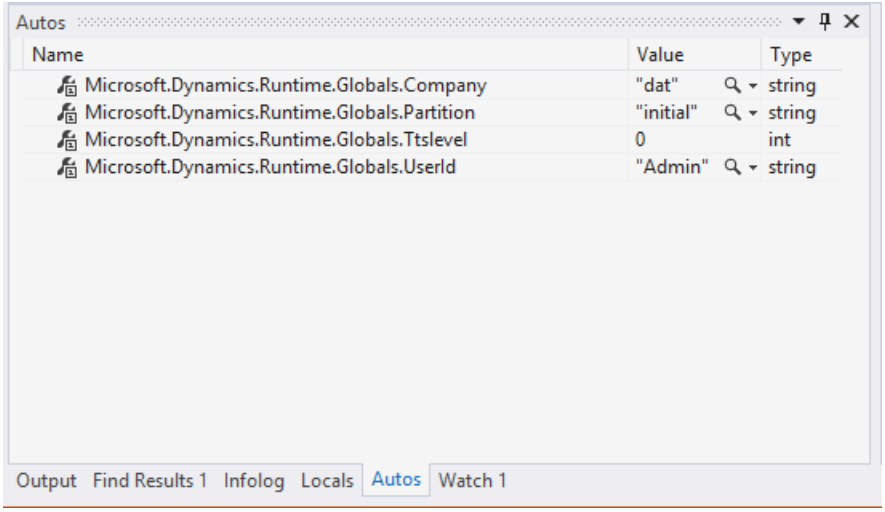
        rental.Comments = "Something";
    }
}
```

Set a breakpoint on the assignment statement. Make your class the startup object in your project, and start by pressing F5. When the breakpoint is encountered, view the rental variable by expanding it in the locals window. You can see that the fields that have been selected (EndMileage and RentalId) appear with their selected values, while the unselected fields appear as null. This signifies their value wasn't fetched from the database. Obviously, this is a debugging artifact. The values of the unselected fields will be the default value for the type of the field. Step over this and notice how the debugger changes the rendering to the actual value.

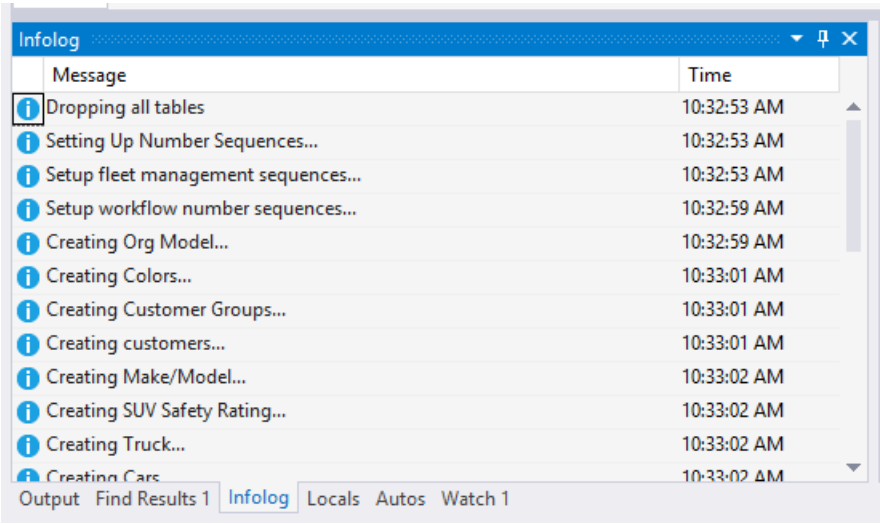
> **NOTE**
> If the table is set to Cache, the system will always fetch all fields from the entire table, irrespective of the field list provided in the code.

The Auto and Infolog Windows

The debugger will allow you to easily access certain parts of the state of the application. This information is available in the autos window, where the current company, the partition, the transaction level, and the current user ID are listed.



There is also a window showing the data that is written to the Infolog.



New breakpoint features

The Visual Studio debugger supports conditional breakpoints and breakpoints that are triggered by hit count. You can also have the system perform specific actions for you as you hit the breakpoint. None of these features were available in the legacy debugger. These are explained below:

- Hit count enables you to determine how many times the breakpoint is hit before the debugger breaks execution. By default, the debugger breaks execution every time that the breakpoint is hit. You can set a hit count to tell the debugger to break every 2 times the breakpoint is hit, or every 10 times, or every 512 times, or any other number you choose. Hit counts can be useful because some bugs don't appear the first time your program executes a loop, calls a function, or accesses a variable. Sometimes, the bug might not appear until the 100th or the 1000th iteration. To debug such a problem, you can set a breakpoint with a hit count of 100 or 1000.
- Condition is an expression that determines whether the breakpoint is hit or skipped. When the debugger reaches the breakpoint, it'll evaluate the condition. The breakpoint will be hit only if the condition is satisfied. You can use a condition with a location breakpoint to stop at a specified location only when a certain condition is true. For example, suppose you're debugging a banking program where the account balance is never allowed to go below zero. You might set breakpoints at certain locations in the code and attach a condition such as balance < 0 to each one. When you run the program, execution will break at those locations only when the balance is less than zero. You can examine variables and program state at the first breakpoint location, and then continue execution to the second breakpoint location, and so on.
- Action specifies something that should occur when the breakpoint is hit. By default, the debugger breaks execution, but you can choose to print a message or run a Visual Studio macro instead. If you decide to print a message instead of breaking, the breakpoint has an effect very similar to a Trace statement. This method of

using breakpoints is called trace points.

Using breakpoints with conditions

Consider the following code:

```
class PVsClass
{
    public static void Main(Args a)
    {
        int i;
        for (i = 0; i < 10; i++)
        {
            print i;
        }
    }
}
```

Put a breakpoint on the print statements by pressing F9 while that statement is selected. This will create a normal, unconditional breakpoint. Now, use the mouse to open the context menu for the breakpoint and select **Condition**. Put in a condition that indicates that the breakpoint should happen when the value of the 'i' variable exceeds 5. Set the class as a startup project, and the class containing the code as the startup item in the project. Run the code. Feel free to modify the value of 'i' using the debugger. Now, remove this breakpoint, and use the Hit count feature to accomplish the same thing.

> **NOTE**
>
> A breakpoint can have several conditions. It's often helpful to hover the cursor over the breakpoint, causing an informative tooltip to appear. Trace points are often useful tot race execution. Insert a trace point on the line in question and log the value of the variable. The trace output will appear in the output window in the debugger.

The immediate window

The immediate window is a useful feature in the VS debugger that allows the user to enter expression and statements to evaluate at any given time. This feature isn't currently implemented in the X++ stack, as is the case for many other languages, notably F#. However, that doesn't mean that the savvy user can't benefit from the immediate window. It just means that snippets must be expressed in C#, not in X++. There's a separate document that describes the details of how this can be done to great effect.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Debug X++ code by using the debugger in Visual Studio

2/18/2021 • 2 minutes to read • Edit Online

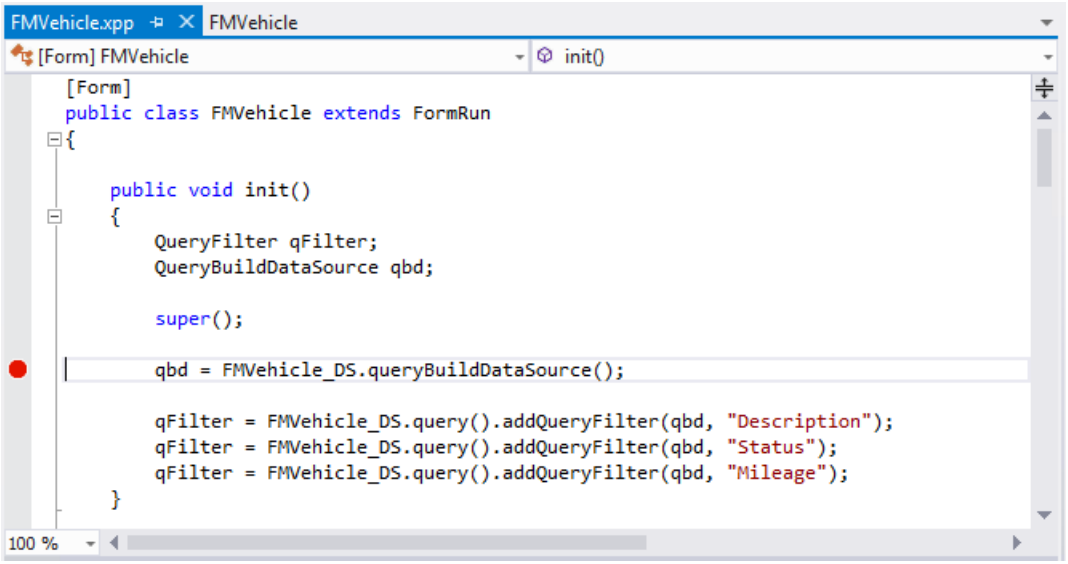
This topic reviews how you can debug X++ code by using the debugging feature in Microsoft Visual Studio.

To debug X++ code, you use the debugger in Microsoft Visual Studio. The process is similar to the process that is used for any other application that is created in Visual Studio. For example, the standard tools for examining the application are available when your code is stopped at a breakpoint.

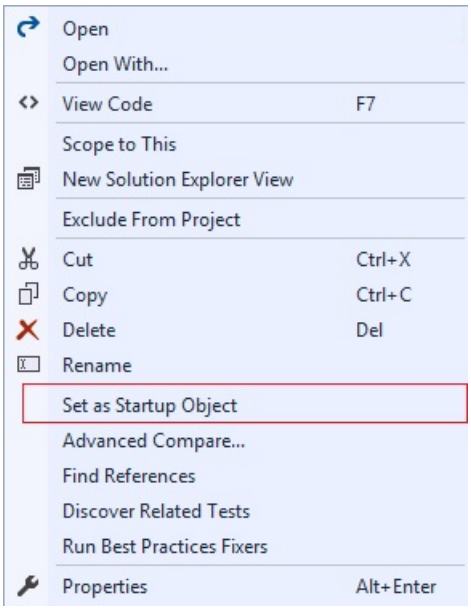
Debug your code

To debug X++ code, follow these steps.

1. In Visual Studio, open the X++ code to debug.
2. Find the line or lines where you want execution to stop, and set breakpoints in those lines. To set a breakpoint in a line, click in the left column of the code editor or press F9 while the cursor is on that line. A red dot indicates that a breakpoint has been set.

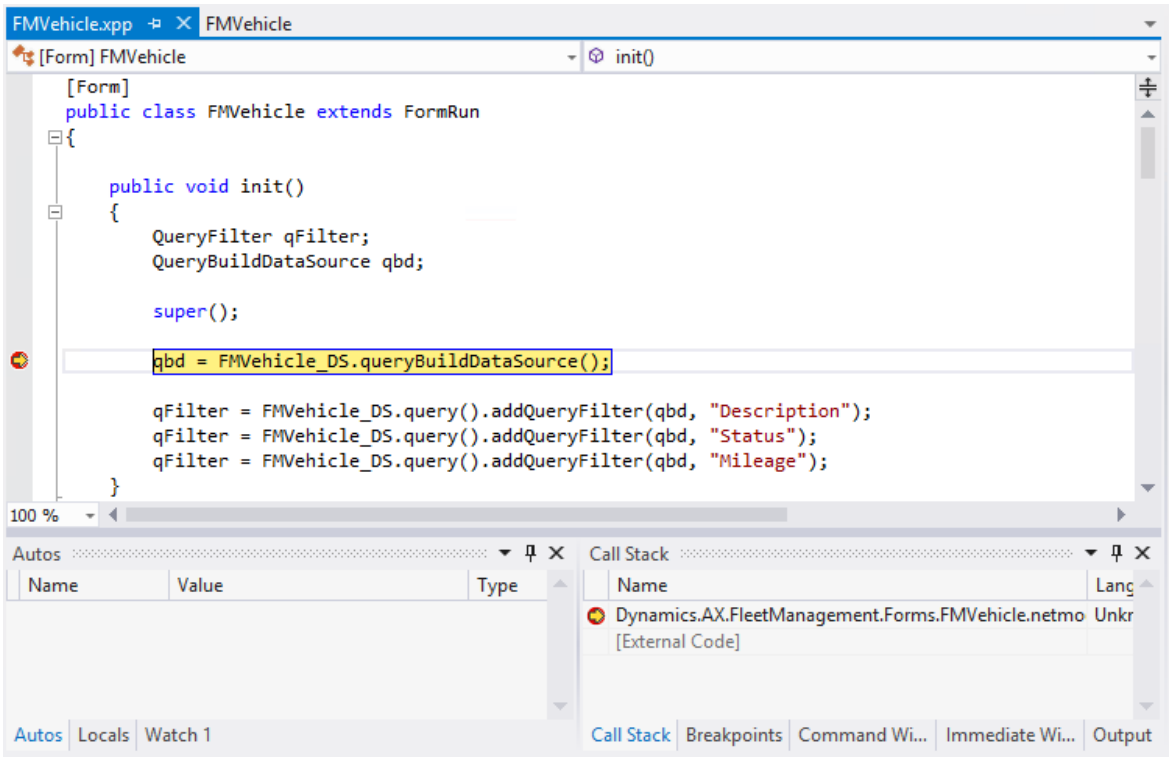


3. Set a startup project and a startup object. Startup objects can be any form, any class that has the **main** method, or any menu item. You can set the startup object in the **Properties** pane for the project. Alternatively, right-click the element in Solution Explorer, and then click **Set as Startup Object**.

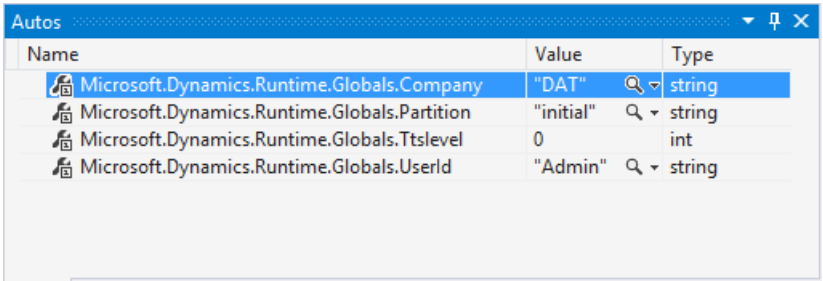


4. On the **Debug** menu, click **Start Debugging**.

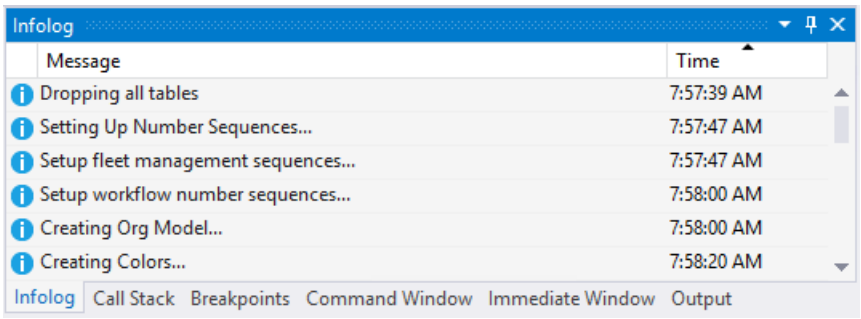
5. In the application, perform the action that causes the code that you're interested in to run. Typical actions include opening a form. Processing stops at the breakpoints that you set.



6. Use the tools in Visual Studio to examine the application. For example, you can hover over variables in the X++ code to see their values. You can also use commands on the **Debug** menu to step through the code. Additionally, tools such as the **Autos** pane in Visual Studio will show important information about the state of the application.



Another tool that is specific to Finance and Operations applications is the Infolog. Often, **info()** statements are added to code to log status messages while the application is running. You can view these Infolog messages directly in Visual Studio. On the **View** menu, click **Infolog**.



7. After you've finished debugging the application, exit the application. Visual Studio will exit debugging mode.

Additional resources

Develop and customize home page

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

EventHandlerResult classes in request or response scenarios

2/18/2021 • 3 minutes to read • Edit Online

Delegate methods and delegate handler methods can be declared to support a request/response scenario, where the delegate calling logic requests the subscribers to provide a response. To support this scenario the **EventHandlerResult** class is most often passed as a parameter, and the delegate handler methods provide their result using one of the result methods on the class. However, the **EventHandlerResult** class can only contain a single result, so if multiple subscribers provide their individual result, the last respondent wins, and the results from the previous subscribers are overwritten.

Before the functionality described in this topic was introduced (platform update 5), there was no mechanism to ensure that, at most, a single subscriber provided a result, and that no results were lost if there were multiple subscribers.

Ensuring, at most, one response

In platform update 5, the **EventHandlerResult** class has an additional static constructor which ensures that the logic fails if more than one subscriber provides a result. The new constructor is named **newSingleResponse**. When instantiating an **EventHandlerResult** object using this method, the framework will throw an exception as soon as a second delegate handler method attempts to provide a result.

```
EventHandlerResult result = EventHandlerResult::newSingleResponse();
this.validateWarehouseTypeDelegate(this.WarehouseType, result);
```

IEventHandlerResultValidator interface

The validation in the **EventHandlerResult** class is handled by injecting an object of a type that implements the **IEventHandlerResultValidator** interface. When instantiating the **EventHandlerResult** object using the **newSingleResponse** static constructor, an **EventHandlerSingleResponseValidator** object is instantiated and injected into the **EventHandlerResult** object, and the injected object becomes responsible for validating any result provided to the **EventhandlerResult** object. Other validation classes can be implemented by having the class implement the **IEventHandlerResultValidator** interface, and injecting it into the **EventHandlerResult** class by instantiating the **EventHandlerResult** object using another new static constructor named **newWithResultValidator**. The constructor takes an argument of type **IEventHandlerResultValidator**, which makes it possible to inject any validator object as long as it implements the **IEventHandlerResultValidator** interface.

For example, the **newSingleResponse** static constructor simply delegates the instantiation to the **newWithResultValidator** static constructor like this.

```
return EventHandlerResult::newWithResultValidator(EventHandlerSingleResponseValidator::construct());
```

Accept and reject request/response scenarios

In certain request/response scenarios, the subscriber is only expected to provide their acceptance or rejection. Using the **EventHandlerResult** class to request acceptance/rejection can be confusing, if the subscriber is only

expected to respond with a Boolean value. In a validation scenario, for example, should the subscriber only respond with Boolean false, when validation fails, or should the subscriber also respond with Boolean true, if validation succeeds? If the response is gathered using an **EventHandlerResult** object, then the second subscriber that validates and replies with Boolean true, might overwrite the Boolean false from the first subscriber.

To mitigate this confusion, two new result type classes have been introduced in Platform update 5: **EventHandlerAcceptResult** and **EventHandlerRejectResult**.

When using the **EventHandlerAcceptResult** class, the delegate handler method can only respond by calling the **accept** method. When using the **EventHandlerRejectResult** class, only the **reject** method can be called.

```
[SubscribesTo(tableStr(InventWarehouseEntity), delegateStr(InventWarehouseEntity,
validateWarehouseTypeDelegate))]
public static void validateWarehouseTypeIsSupportedStandardDelegateHandler(
    InventLocationType _inventLocationType,
    EventHandlerAcceptResult _result)
{
    switch (_inventLocationType)
    {
        case InventLocationType::Standard:
        case InventLocationType::Quarantine:
        case InventLocationType::Transit:
            _result.accept();
            break;
    }
}
```

The two new classes also contain a **newSingleResponse** static constructor for use in scenarios where, at most, one subscriber is allowed to respond with their rejection or acceptance. Whether any subscriber has responded can still be answered by querying the **hasResult** method, and the acceptance/rejection is queried by calling either the **isAccepted** or **isRejected** methods for the **EventHandlerAcceptResult** and **EventHandlerRejectResult** classes, respectively.

```
boolean ret = false;
EventHandlerAcceptResult result = EventHandlerAcceptResult::newSingleResponse();
this.validateWarehouseTypeDelegate(this.WarehouseType, result);
if (result.hasResult())
{
    ret = result.isAccepted();
}
```

Write business logic by using C# and X++ source code

2/18/2021 • 11 minutes to read • Edit Online

The primary goal of this tutorial is to illustrate the interoperability between C# and X++. In this tutorial, you'll write business logic in C# source code and in X++ source code.

In this tutorial, you'll write business logic in C# source code and in X++ source code. You'll get experience with the following:

- New tools in Visual Studio.
- The handling of events in C#.
- The use of Language Integrated Query (LINQ) in C# to fetch data.

Prerequisite

This tutorial requires that you access the environment using Remote Desktop, and be provisioned as an administrator on the instance.

> **NOTE**
> Debugging support for the C# project does not work if the **Load symbols only for items in the solution** check box is selected. Since this option is selected by default, it must be changed prior to running the lab. In Visual Studio, click **Dynamics 365** > **Options**, and clear the **Load symbols only for items in the solution** check box.

Scenario

Too many cars have been rented to drivers who have a history of unsafe driving habits. The Fleet Management rental company needs to check driving records from external sources. Upper management has decided to subscribe to a service that is hosted by the Department of Transportation (DOT), which is the legal entity that manages drivers' licenses and associated information. This service retrieves the number of citations for the given unique license number. It's not easy to call external services directly from X++ source code. Visual Studio has tools for generating the "code-behind" (in C#) that calls the services, and these tools make the development effort easy. The obvious choice would be to leverage Visual Studio to write the code. However, in this tutorial your code won't actually call an external service, because the logistics are beyond the scope of the simple lab environment. Instead, we provide a mock implementation of a service call. The goal of this tutorial is to teach an understanding of the current state of C# and of interoperability with X++.

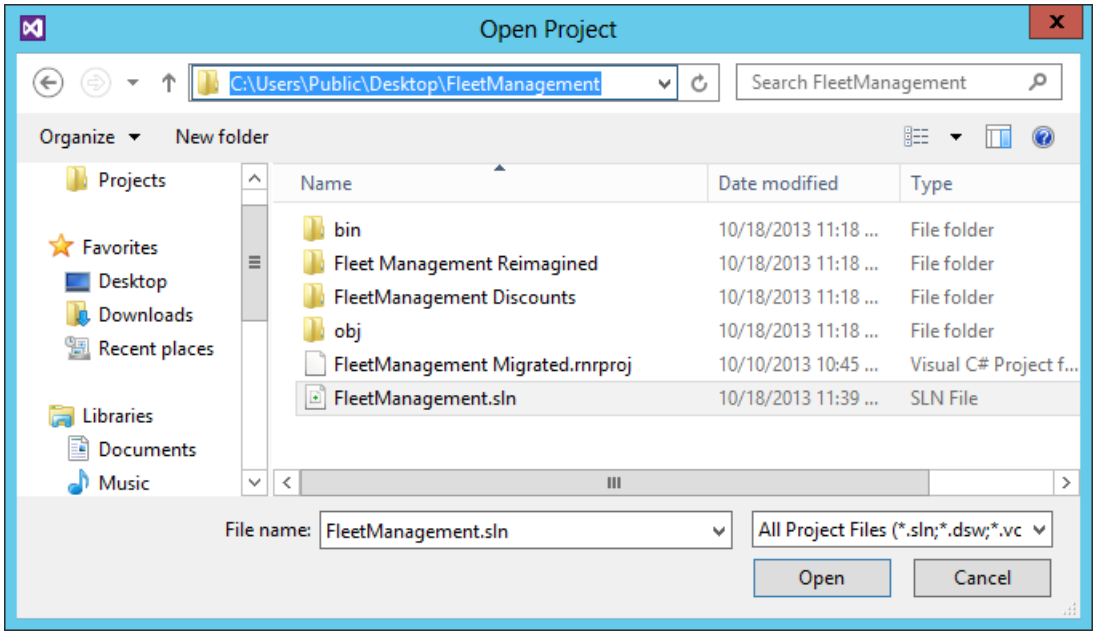
Create a C# class library

You can create a reference from a project to the C# class library, or to any other type of C# project that generates an assembly. Such references affect the build order. The C# project is built before the project that references and depends on it. The infrastructure understands the references, and will make sure that the C# assemblies are deployed correctly to the cloud before execution. Follow these steps to create a C# class library in the Fleet Management solution:

1. In Visual Studio, click **File** > **Open project/solution**.
2. In the **Open Project** dialog box, in the **File name** text box, type the following path, and then press

Enter: *C:\users\public\desktop\FleetManagement*.

3. Select the file named FleetManagement.sln, and then click **Open**. If the solution file is not on your computer, the steps to create it are listed in Tutorial: Create a Fleet Management solution file out of the Fleet Management models in the AOT.



4. Right-click the **FleetManagement** solution, and then click **Add** > **New Project**. The **Add New Project** dialog is displayed.

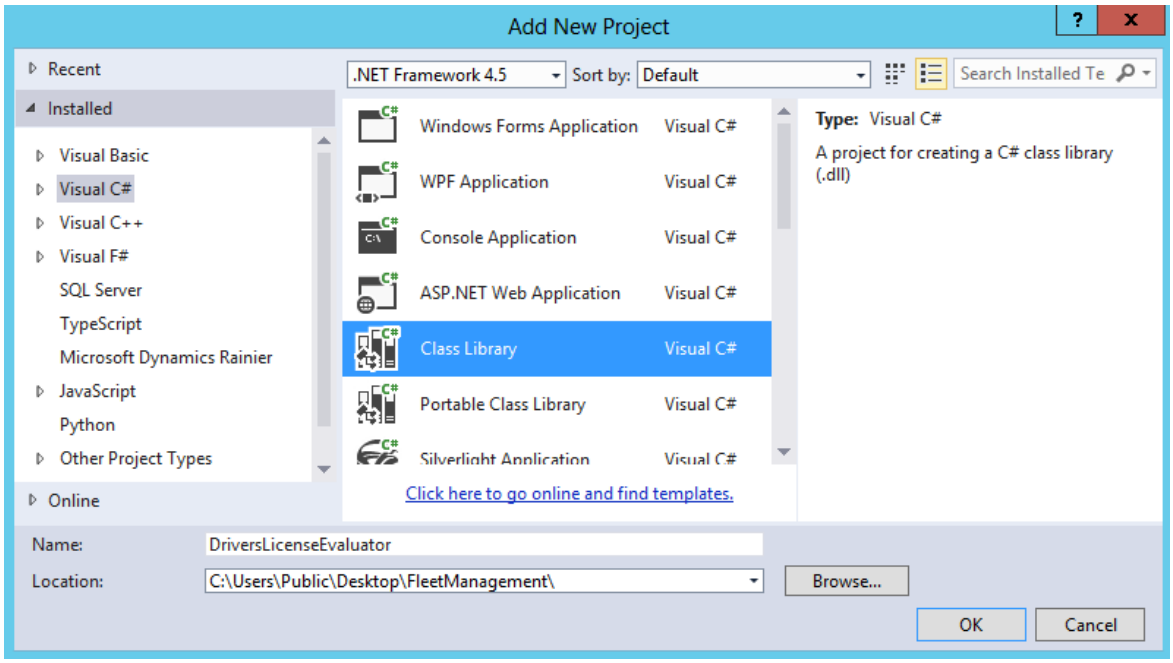
5. In the left pane, click **Visual C#**, and then in the middle pane, click **Class Library**.

6. At the bottom in the **Name** text box, type the name **DriversLicenseEvaluator**.

7. In the **Location** text box, type the following directory path: *C:\users\public\desktop\FleetManagement*.

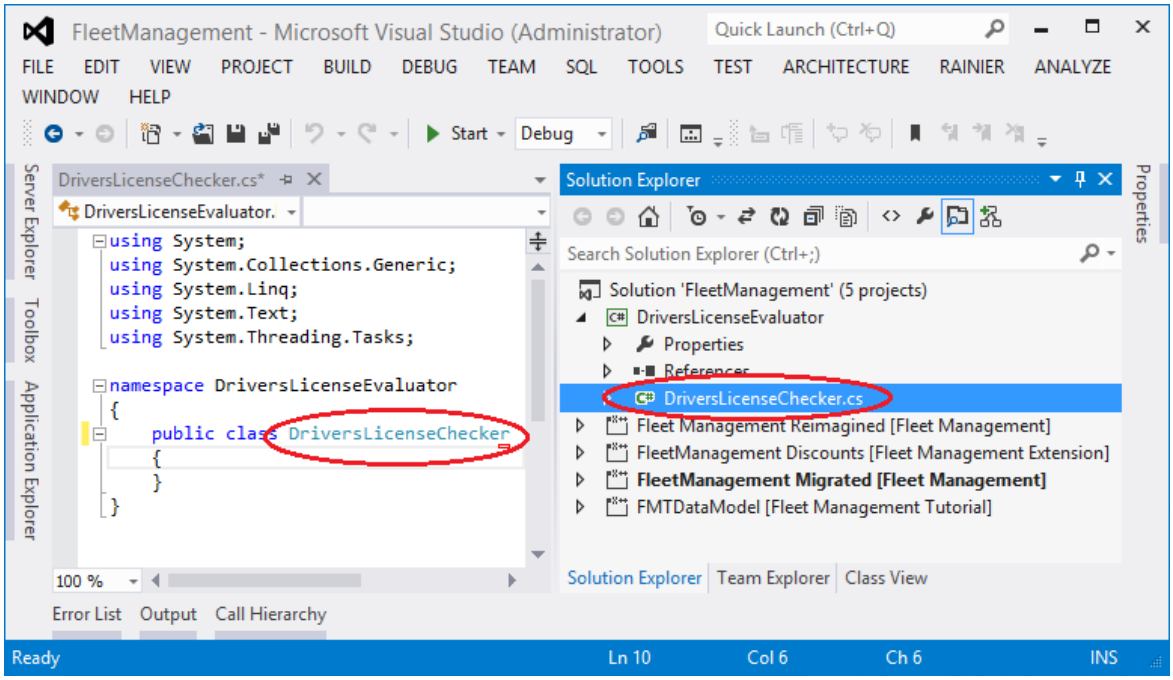
8. Verify that your project is set to ".NET Framework 4.5" in the drop-down list at the top.

9. Click **OK** to create the project.



10. In **Solution Explorer**, under the DriversLicenseEvaluator project, right-click the file name Class1.cs and rename it DriversLicenseChecker.cs.

11. Click **Yes**, when prompted to rename all references to the class.



Write a C# method named CheckDriversLicense

In this section, you add C# code for a method named CheckDriversLicense. The method must validate the driver's license. To do this, the method must retrieve the driver's license number, which is stored in the customer table. The method is given the RecId value for the customer record that contains the information required by the method. Your C# code uses the LINQ provider to read from the customer table. For LINQ to work, you must first add references pointing to the LINQ assemblies. You add these references to the C# project named DriversLicenseEvaluator.

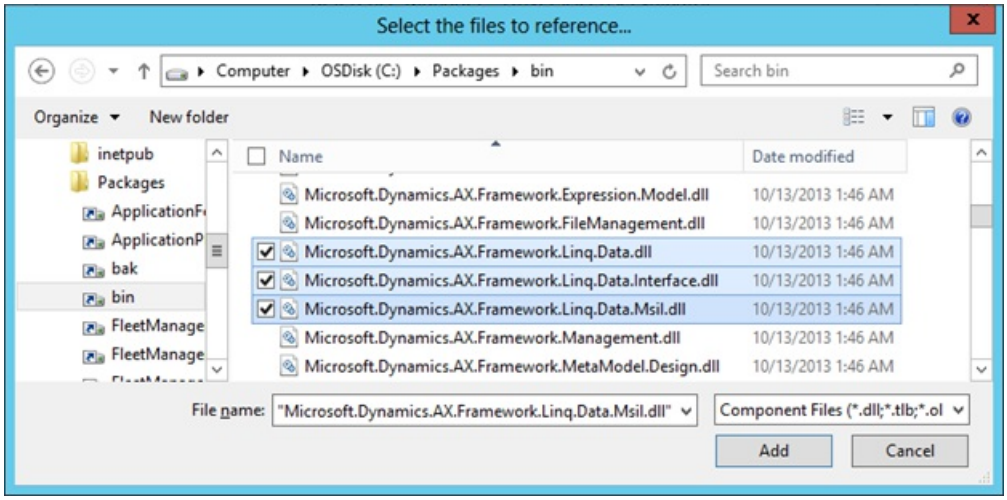
1. In **Solution Explorer**, expand the DriversLicenseEvaluator project node, right-click **References**, and then click **Add Reference**.

2. Click **Browse** and then enter the following path: C:\Packages\bin

   Note that in some environments, the location of the packages folder is not on the c: drive.

3. In the **File name** field, type the pattern *LINQ*.dll and then press **Enter**. You'll see a list of assemblies with the name LINQ in them. From that list, select the following files, and then click **Add**:

   - Microsoft.Dynamics.AX.Framework.Linq.Data.dll
   - Microsoft.Dynamics.AX.Framework.Linq.Data.Interface.dll
   - Microsoft.Dynamics.AX.Framework.Linq.Data.Msil.dll



4. You must also add the support assemblies that contain the Common type that you'll use in the code below. Click **Browse** again, and then type the following file name into the field:

   - Microsoft.Dynamics.AX.Xpp.Support.dll
   - Microsoft.Dynamics.AX.Data.Core.dll

5. Click **Add**, and then click **OK**. The assemblies now appear under the references node in the project.

6. Repeat the **Add Reference** process, except this time, add the following DLL file from the indicated path:

   - Dynamics.Ax.FleetManagement.dll, in C:\Packages\FleetManagement\bin

7. In **Solution Explorer**, select the reference Dynamics.Ax.FleetManagement.dll reference and set the property **Copy Local = False**.

8. In **Solution Explorer**, right-click **DriversLicenseChecker.cs**, and then click **View Code**.

9. Add the following three using statements to the **DriversLicenseEvaluator** namespace, to reduce the verbosity of code that references external classes. using Dynamics.AX.Application; using Microsoft.Dynamics.AX.Framework.Linq.Data; using Microsoft.Dynamics.AX.Xpp; Your C# code should now look something like the following example.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DriversLicenseEvaluator
{
    using Dynamics.AX.Application;
    using Microsoft.Dynamics.AX.Framework.Linq.Data;
    using Microsoft.Dynamics.Ax.Xpp;

    public class DriversLicenseChecker
    {
    }
}
```

10. Replace the class CheckDriversLicense with the following code.

> **TIP**
> If you prefer, you can paste in the code from the DriversLicenseChecker.cs file in the C:\FMLab directory.

```
public class DriversLicenseChecker
{
    public static bool CheckDriversLicense(long customerId)
    {
        // Use LINQ to get back to the information about the license number
        FMCustomer customer;
        QueryProvider provider = new AXQueryProvider(null);
        var customers = new QueryCollection<FMCustomer>(provider);

        // Build the query (but do not execute it)
        var query = from c in customers
            where c.RecId == customerId
            select c;

        // Execute the query:
        customer = query.FirstOrDefault();
        if (customer == null)
        {
            throw new ArgumentException
                ("The customerId does not designate a customer");
        }

        if (string.IsNullOrEmpty(customer.DriverLicense))
        {
            // No driver's license was recorded. Veto the rental.
            return false;
        }

        // Call the DOT web service to validate the license number.
        // This is not practical for this lab, because all the service providers
        // charge for this service. Instead, just assume that any license number
        // that contains the sequence "89" is valid.
        // In the demo data, this is true for Adrian Lannin,
        // but not for Phil Spencer.
        return customer.DriverLicense.Contains("89");
    }
}
```

Understand the LINQ code

Before proceeding with more C# code, verify that you understand the LINQ code you just added. More details about LINQ are provided in the Technical Concepts Guide, so only the basics are described below.

- First, a *provider* is created. It provides access to all the tables.
- Next, a *collection* of all customers is created. The customer of interest is retrieved from this collection.
- Then, a *query* is created with a where clause that designates the requested customer by **RecId**.
- The call to the FirstOrDefault method forces execution of the query.
- The method assigns the single matching customer to the customer variable. (Null is assigned if the RecId value matches no customer.)
- Finally, the customer data is tested to see if the associated driver's license is valid. (Does the license contain "89"?)

Handle the event when a record is added

The following subsections provide the following:

- Explain the upcoming code items and their inter-relationships.
- Show the code for an event handler.
- Associate the handler with the event occurrences.

Preparatory overview

When an attempt is made to add a record to a table, the OnValidateWrite event is raised before the record is written to the database. You want your CheckDriversLicense method to be called each time on the OnValidateWrite event is raised for the FMRental table. To do this, you now need to write a C# method that is invoked by the event, and which calls your checkDriversLicense method. In other words, you need to write an event handler that calls your CheckDriversLicense method. The event handler method receives a parameter of the type, DataEventArgs. The event handler can set a value in the DataEventArgs structure to accept or reject the record. After you write your event handler method, you connect it to the event by assigning, or adding it to the OnValidatedWrite delegate that is a member of the FMRental table. You write this assignment in the init method of the data source of the FMRental form. This assignment to a delegate might seem odd. After all, we're modifying existing code (FMRental) to add handlers, which contradicts the main value proposition of loose coupling that eventing is supposed to offer. This assignment step is temporary. We'll eventually have the same story in C# as we do in X++, where an attribute is applied to the C# event handler as the mechanism that ties the delegate to the handler.

> **NOTE**
>
> The data source init method is called when the form is opened. Technically, the init method is inherited from the FormDataSource class.

Write an event handler method

In C#, write the following event handler method and add it to the DriversLicenseChecker class.

```
public static void OnValidatedWriteHandler(Common table, DataEventArgs args)
{
    var validateEventArgs = args as ValidateEventArgs;

    // Do not check if already rejected.
    if (validateEventArgs.parmValidateResult())
    {
        var rentalTable = table as FMRental;
        if (rentalTable == null)
        {
            throw new ArgumentNullException("table");
        }

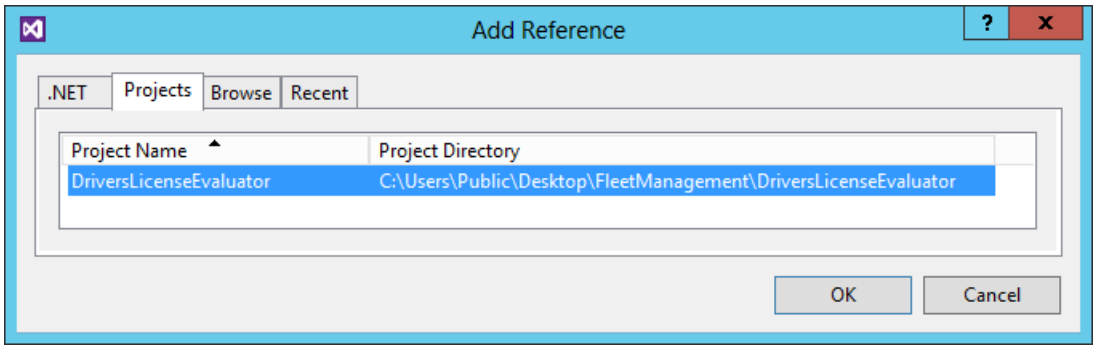
        var result = CheckDriversLicense(rentalTable.Customer);
        validateEventArgs.parmValidateResult(result);
    }
}
```

Build the DriversLicenseEvaluator project by right-clicking the project node and then clicking **Build**.

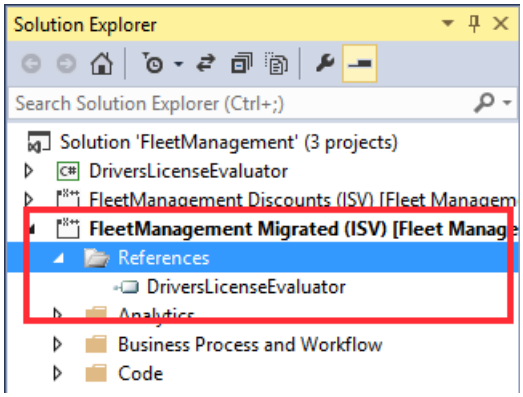
Add a reference pointing to the DriversLIcenseEvaluator project

Create a reference from the X++ project named **FleetManagement Migrated** to the C# project named **DriversLicenseEvaluator**, by completing the following steps.

1. Right-click the FleetManagement Migrated project, click **Add**, and then click **Reference**. Select the row for the DriversLicenseEvaluator project in the **Projects** references tab, and then click **OK**.

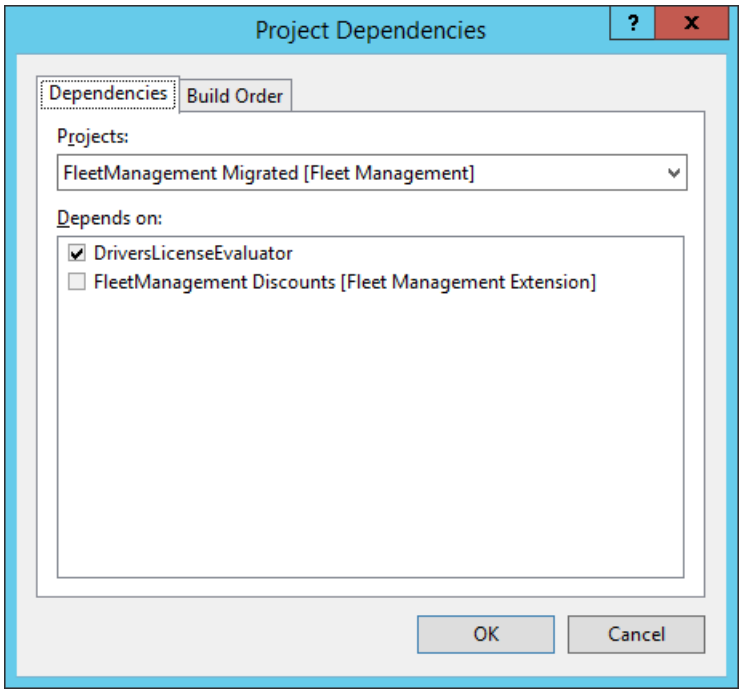


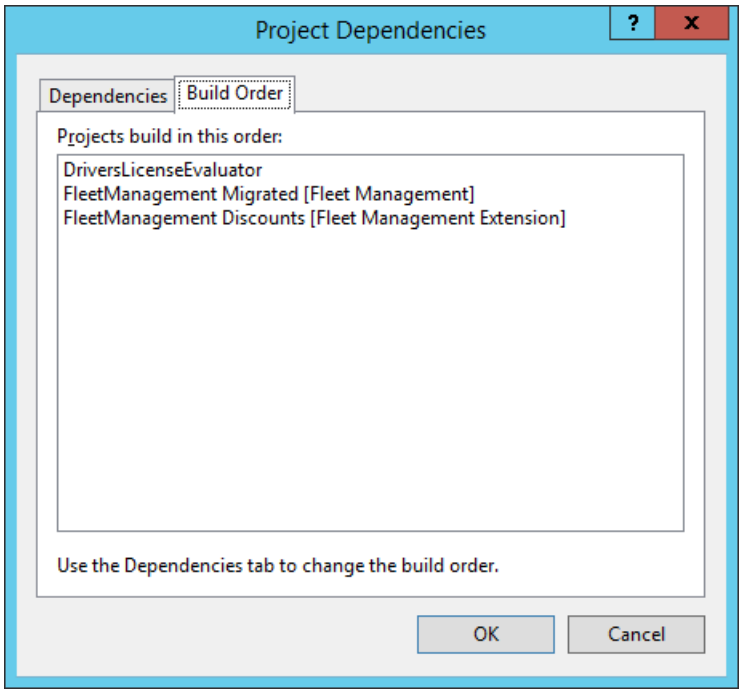
2. Under the FleetManagement Migrated project, expand the **References** node, and there you see new reference to the **DriversLicenseEvaluator** project.



Build sequence

Your C# DriversLicenseEvaluator project will be built before the FleetManagement Migrated project is built. This is because the added reference makes the Fleet project dependent on your project. The build sequence is easy to see if you right-click the FleetManagement solution, click **Project Build Order**, and then click **Dependencies**.





Add your event handler to a delegate

1. In **Solution Explorer**, navigate to **FleetManagement Migrated > User Interface > Forms > FMRental**.
2. Double-click the **FMRental** form. The Visual Studio designer opens to the form.
3. Expand the **Data Sources** node to show the data sources used in the form.
4. Expand the **FMRental** data source, and then the **Methods** node to list the methods defined on the data source.
5. Right-click **Methods**, and then click **Override > init**. The list displays all of the methods on the data source that haven't yet been overridden. When you select **init**, this opens the file **FMRental.xpp** in the X++ code editor with the cursor near the template for the init method.
6. At the end of the **init** method body, use the += operator to add one assignment to a delegate.

```
FMRental.onValidatedWrite += eventhandler
    (DriversLicenseEvaluator.DriversLicenseChecker::OnValidatedWriteHandler);
```

7. Click to save, and then build the entire solution.

Final test

In this section, you set breakpoints and run the Fleet application under the Visual Studio debugger. This enables you to prove the following:

- Your LINQ query runs when the OnValidateWrite event is raised.
- Your LINQ query successfully retrieves the data for one customer.

Prepare the test

1. In **Solution Explorer**, navigate to **FleetManagement Migrated > User Interface > Forms.**
2. Right-click **FMRental**, and then click **Set as Startup Object**.
3. In the code editor for DriversLicenseChecker.cs, find the OnValidateWriteHandler method. Find the following line of code.

```
var result = CheckDriversLicense(rentalTable.Customer);
```

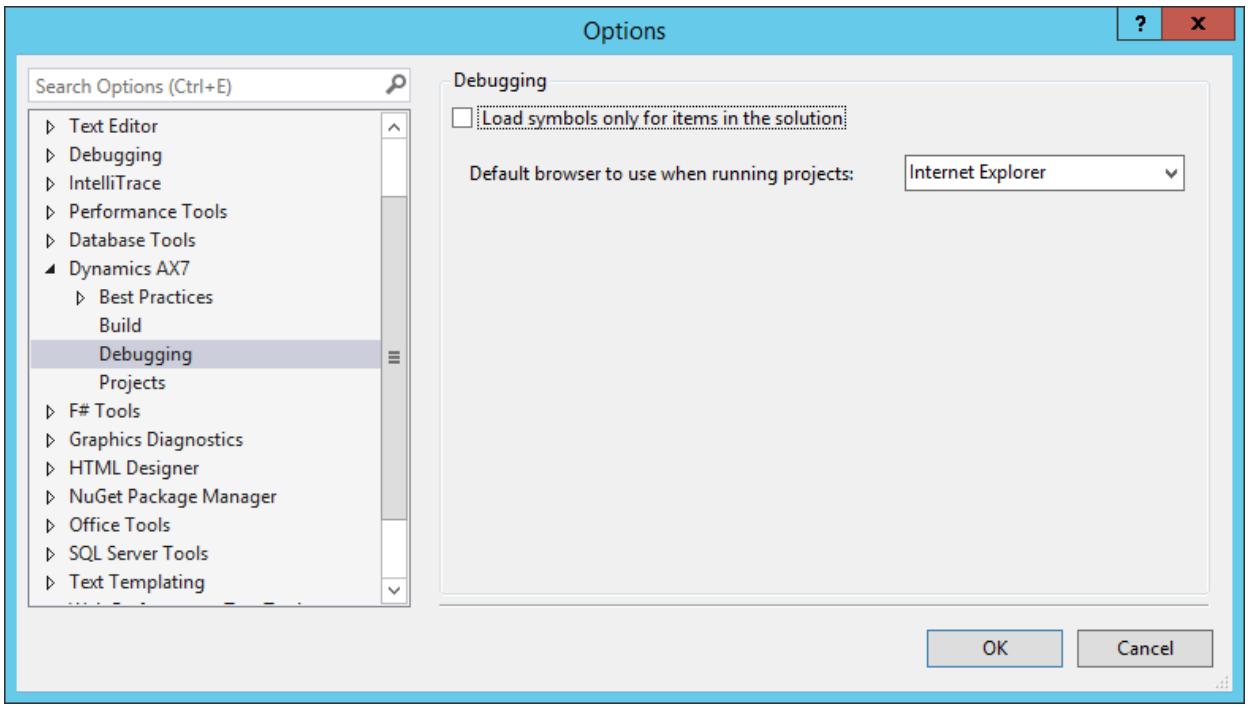
4. Set a breakpoint on that line of code. You do this by clicking in the left margin at that line. A red dot displays when the breakpoint is set.

5. In the CheckDriversLicense method, set another breakpoint at the following line.

```
if (string.IsNullOrEmpty(customer.DriverLicense))
```

Run the test

For this test, we'll be debugging the C# code that we've written. To do this, we need to inform Visual Studio to load the symbols for the assembly that contains the C# code. Go to **Dynamics 365 > Options > Debugging** and verify that the **Load symbols only for items in the solution** check box is not selected.



> **TIP**
> If you're unable to get to the breakpoint in the C# code, you may want to open the **Modules** window (**Debug > Windows > Modules**), find the C# module and load it explicitly.

1. Click **Debug > Start Debugging**. This starts the Fleet application, and a browser window with the **FMRental** form is displayed.

2. Click on any **Vehicle rental ID** to view details.

3. Click the **Edit** icon near the top left of the form. The icon looks like a pencil.

4. In the **To** field of the **Rental** section, increase the date by one day.

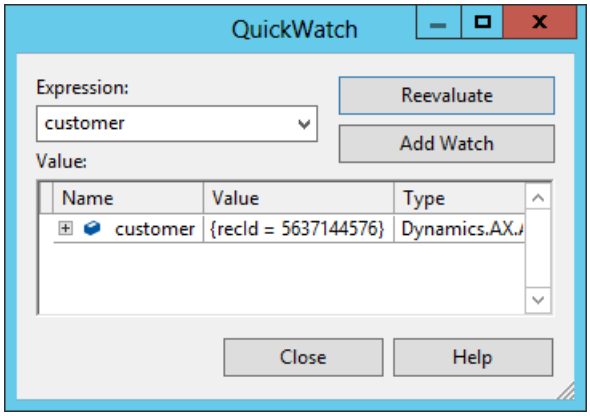
5. Click the **Save** button. This causes the focus to shift to Visual Studio at your highlighted breakpoint. This line shows that the OnValidatedWrite event was raised, and that your handler method was called.

6. Press **F5** to continue the run. Instantly, your other breakpoint becomes highlighted.

7. Find the variable customer a few lines above your breakpoint.

8. Right-click the customer variable, and then click **QuickWatch**. Any long integer value proves that your

LINQ query worked.



9. Press **F5** to complete the **Save** operation.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Changes in X++ and the X++ compiler

2/18/2021 • 6 minutes to read • Edit Online

This topic reviews the changes made to the compiler for Finance and Operations applications. The X++ compiler has been rewritten. No backward-incompatible changes have been introduced to X++ except where required by structural changes to the product. A few language enhancements have been added. A new X++ best practice tool has also been implemented, which allows the addition of user-defined custom rules.

No more p-code, everything is in .NET Framework CIL

Through Microsoft Dynamics AX 2012, X++ source code was compiled into p-code, which was understood by the interpreter at run time. Optionally, you could then compile the p-code into Microsoft .NET CIL (Common Intermediate Language). CIL is what the .NET compilers for C# and Visual Basic generate. However, X++ CIL code was usable only in limited cases, mainly for code executed in services and batch jobs. The new X++ compiler generates CIL only. There is no more p-code. The following tools that worked with p-code are now obsolete and have been removed from Dynamics AX and replaced by .NET tools:

- The X++ compiler that generated p-code.
- The special compiler that input p-code and generated .NET CIL.
- The run time interpreter of p-code including its deterministic garbage collector.
- The Dynamics AX Debugger.
- The Dynamics AX Code Profiler, which helped find performance bottlenecks.
- The AX .NET Business Connector for external applications that interoperated with Dynamics AX.

There are important benefits to X++ code running exclusively as .NET CIL, including:

- CIL runs much faster in most scenarios. In cases where there are many method calls, and a lot of algorithmic content (as opposed to database access), you can expect significant performance improvements.
  - It's easier to write application logic in other managed languages.
  - There's no longer any need for the AX .NET Business Connector or managed proxies, because the assemblies can be consumed directly.
  - The preferred way of consuming business logic externally is by using services.
- CIL can efficiently reference classes that are available in other .NET assembly DLL files, without the run time reflection-based overhead that burdened the p-code interpreter.
- CIL can be operated on by the many .NET tools.

Unit of compilation

Previous versions of Microsoft Dynamics AX could compile X++ at the level of an individual method. Thus if a compilation of a class found an error in one method of the class, the methods that did compile correctly were still runnable. In the standard compilation unit now is the same as for other .NET languages such as C#. If any method in a model element (class, form, query etc.) fails to compile, the whole compilation fails.

Enhancements to X++

X++ is now a first-class citizen in the .NET world. Therefore we are adding to X++ several constructs that are available in C# and other mainstream .NET languages. The changes are generally non-intrusive. In fact, very few changes have been made to the syntax and semantics of existing X++ code. Most of these additions are in the following list:

- The `finally` keyword is now available to follow the `try` and `catch` keywords. The semantics are identical to the semantics in C#. The statements provided in the finally clause are executed irrespective of whether the try block threw any exceptions.
- The `using` keyword has been added as shorthand for referencing .NET namespaces. The following code example illustrates two ways of utilizing the using keyword to reference namespaces:

```
using SysColl = System.Collections;  // SysColl is an alias for the whole namespace.
using           System.CodeDom;      // Contains the class named CodeComment.

public class MyClass2
{
    static SysColl.ArrayList arrayList = new SysColl.ArrayList(); // Initialized on declaration.
    CodeComment codeComment          = new CodeComment("I am a comment.");

    // More X++ code here.
}
```

- You can now initialize a class's field in the field's declaration statement. This is illustrated twice in the previous code example.
- You can declare variables in smaller scopes, not just at the start of methods.
- The `var` keyword is available as a shortcut that allows the compiler to infer the type of the declared variable.
- A class's fields can now be static. In previous versions, only instance fields were allowed.
- A class can now have one static constructor. In previous versions, only instance constructors were available. The following X++ code example shows the syntax for a static constructor, through the new keyword typenew.

```
public class MyClass4
{
    static utcdatetime utcInitialized3;  // Static variable member.
    static void typenew()                // Static constructor member. 'typenew' is a new keyword.
    {
        utcInitialized3 = DateTimeUtil::utcNow(); // Static variable referenced without class name.
    }
}
```

- An attribute decoration, such as on a class or a method, can now omit the suffix of the attribute name if the suffix is `Attribute`. So the X++ joins the C# in allowing `[MyFavorite]` instead of requiring `[MyFavoriteAttribute]`.
- A delegate can now be defined in a table, form, or query, and not just in a class.
- Attributes are now applied to the handlers of delegates and methods, to map the handlers to those targets.
- Classes can now be nested in X++ source code. Nested classes are available only inside forms (such as a class that extends FormRun) to represent controls, data sources, or data fields.

Backward-incompatible changes to X++

There are a few changes to X++ that require corresponding changes in legacy custom X++ source code. Most of these changes are in the following list:

- The following keywords are no longer part of the X++ language, and their use causes compilation errors:

    - `changeSite`
    - `pause`
    - `window`
- In legacy X++, it was possible to designate a method to run either on the client or the server. This is no longer possible. All compiled X++ code is executed as .NET CIL on the server. There is no longer any X++ code that is evaluated at the client site or in the browser, therefore, the two keywords, *client* and *server*, are now ignored. Their use doesn't cause a compile error, but they should not be used in any new X++ code.
- In Microsoft Dynamics AX 2012, there were a few areas where X++ behaved differently when compiled to p-code versus CIL. In Finance and Operations applications, all these areas behave as they did in CIL in Microsoft Dynamics AX 2012. The significant behavioral differences between X++ p-code versus X++ as CIL were as follows:
    - In CIL, the `real` data type is represented as `System.Decimal`. This means the range and precision for each `real` is different than it was under p-code. This change was already in effect in Microsoft Dynamics AX 2012 when .NET CIL was run.
    - An assignment of one entire array to another was performed in value in p-code mode, but it's performed by reference in CIL mode.
    - CIL helper methods such as `Global::runClassMethodIL` have been removed, since they're no longer relevant.
- There is no concept of a job, in the sense of **AOT** > **Jobs** > **MyJob**. To quickly and easily run an X++ method, you can still add in a `static Main` method to a class, and then set the class as the startup object form for the project in Microsoft Visual Studio. When the project is run, the `Main` method will be run.

Additional resources

Language Integrated Query (LINQ) provider for C#

Develop and customize home page

Language Integrated Query (LINQ) provider for C#

2/18/2021 • 6 minutes to read • Edit Online

This topic discusses the LINQ provider.

LINQ (Language Integrated Query) is a set of classes and methods that enable you to access data that is stored in a variety of places and formats. The LINQ framework is the standard for accessing data in managed languages. LINQ presents to programmers a unified and consistent API for data access from heterogeneous data sources, such as:

- In-memory object graphs
- Active Directory entries
- Flickr pictures and XML
- SQL Server

The LINQ provider allows the user to access business data by using .NET managed languages.

Two syntactical mechanisms for accessing LINQ

There are two syntactical approaches for using LINQ, as described in the following table.

APPROACH	X++	C# AND VISUAL BASIC
LINQ by standard method call syntax.	Impractical. Language support for generics is vital for LINQ and is not supported in X++.	Available, requires lambda syntax.
LINQ by specialized syntax that is understood by the compiler.	Not available.	Available, easier to use.

There are two syntactic mechanisms for accessing the LINQ provider in C# (or in Visual Basic):

- By standard, or fluent, method call syntax.
- By specialized syntax that the C# compiler has been enhanced to understand as equivalent to the LINQ method calls. (Such syntax is sometimes called "syntactic sugar".)

This topic is going to review each syntactic mechanism for LINQ, starting with the easier specialized syntax.

LINQ by specialized syntax in C#

Some .NET languages understand specialized syntax for LINQ as an alternative that is easier for us to write. C# is one such language.

To use LINQ in C#, you must understand the C# keyword **var**, which is used to declare variables. The var keyword tells the compiler to figure out the data type of the variable by what is assigned to the variable. This feature is now also available in X++. The type is implicit in the source code, and the type is settled and unvarying after the compilation completes.

Comparing X++ to C# LINQ

The X++ language supports the useful and easy to use `while select` statement. This lest you compare the X++ `while select` syntax to the specialized C# LINQ syntax. First, here is the X++ sample.

```
CustTable ct;     // X++, traditional while select.
CustTrans trans;

while select * from ct
    where ct.AccountNum >= '4000'
    join RecId from trans
    where trans.RecId == ct.RecId
    order by ct.AccountNum desc
{
    print ct.AccountNum;
}
```

Next is an equivalent query in C# with the specialized LINQ syntax.

```
// C#, with specialized LINQ syntax.
// Get access to the data provider:
var provider = new QueryProvider(null);

var customers = new QueryCollection(provider);
var customerTransactions = new QueryCollection(provider);

var query = from ct in customers
            from trans in customerTransactions
            where ct.AccountNum >= "4000"
            where trans.AccountNum == ct.AccountNum
            orderby ct.AccountNum descending
            select ct;

foreach (var ct in query)
{
    System.Console.WriteLine(ct.AccountNum);
}
```

LINQ query in C# by method syntax, using the lambda operator >

Next is another use of LINQ in C#, except this time the more standard syntax is used to call the LINQ API. The approach also involves use of the lambda operator `>`. The following C# query is functionally equivalent to the preceding C# query.

```
var query = customers
    .Where(c => string.Compare(c.AccountNum, "4000") >= 0)
    .Join(customers,
        primary => primary.AccountNum,
        foreign => foreign.AccountNum,
        (primary, foreign) => new { P = primary, F = foreign })
    .OrderBy(primaryAndForeign => primaryAndForeign.P.AccountNum)
    .Select(primaryAndForeign => primaryAndForeign.P);
```

There's a good match between the `while select` syntax used in X++ and the specialized LINQ syntax in C# (Visual Basic has particularly good LINQ syntax). It's evident that the specialized LINQ syntax is actually very useful in expressing joins, but the specialized syntax built into the C# compiler doesn't handle the extensions provided by the Finance and Operations applications.

Limitation of the specialized LINQ syntax

A limitation of the specialized LINQ syntax is that it can't be augmented with extensions to the LINQ provider. In contrast, standard syntax of method calls plus the lambda operator can be extended as needed. For instance, the LINQ framework provides a method for cross-company hints that can't be expressed in the special syntax for LINQ in C#. Fortunately, due to the ability to compose queries, this limitation need not be a major problem. Calls

to esoteric LINQ methods can be appended to the specialized LINQ syntax. The following C# code shows this being done for the **crosscompany** method.

```
// C#, mixing LINQ syntax mechanisms.
var query = (from ct in customers
             from trans in customerTransactions
             where ct.AccountNum >= “4000”
             where trans.AccountNum == ct.AccountNum
             orderby ct.AccountNum descending
             select ct).crosscompany();
```

LINQ query execution

The code generated for a LINQ query builds a tree at run time. When the results of the query are required, this tree is passed to the backend that will interpret it, and will provide the data as expressed in the query. The X++ compiler also builds a tree to express the query, but the X++ compiler has intimate knowledge about the capabilities of the database backend. This has several important implications as described in the following subsections.

Inability to diagnose problems at LINQ compile-time

The C# compiler is largely unable to foresee and diagnose errors that will occur at run time due to the inability of the backend to process an incompatible LINQ query. For instance, in the following C# code block, the specialized LINQ syntax is valid according to the C# compiler. Yet at run time, an error would occur.

```
var customerQuery = from c in db.Customers
                    where (from o in db.Orders
                    where o.ShipCountry == “Germany”
                    select o.CustomerID).Contains(c.CustomerID);
```

This query can't be handled by the current data layer, and while no errors are diagnosed at compile time, an error would occur at run time.

Performance penalty with LINQ

There is an overhead penalty paid at run time, when analysis of the tree occurs, and a suitable access language is generated. As we might expect, the performance penalty is incurred when analyzing the LINQ expression tree. The time required at run time to actually fetch the data doesn't vary much between C# and LINQ versus X++ with `while select`. Our preliminary numbers show that the beginning-to-end performance of the query is about three times longer with C# LINQ, compared to X++ `while select`, when very few records are fetched. But when many records are fetched, the total times are about the same between C# and X++. The conclusion is that it takes much longer to fetch a lot of records than it takes to analyze the language tree.

Composing queries with LINQ

The model that's provided by LINQ allows queries to be composed of subqueries. The X++ language can't cleanly provide this feature. To understand this, consider the following C# LINQ code. A flag is passed to a method to control the ordering of data results.

```
private IEnumerable RichCustomers(bool orderByName)
{
    // Create a query for the rich customers. Note carefully
    // that no data is fetched when this is executed.
    var q = from c in customers where c.AmountMst > 1000000.0m select c;

    if (orderByName)
    {
        // Add the order by clause to the existing query.
        // Still no data is yet fetched.
        return q.OrderBy(c => c.AccountNum);
    }
    else
    {
        return q;
    }
}
```

Set based operations with LINQ

LINQ queries can be applied for CRUD operations. But the model for updating, deleting, and inserting records isn't useful for the expression of set based operations. We're now working on extensions to add to the LINQ model that will translate into set based operations.

Additional resources

Changes in X++ and the X++ compiler

Develop and customize home page

Write best practice rules

2/18/2021 • 17 minutes to read • Edit Online

This topic describes how you can author best practice rules in C#, for both metadata and X++ code. Best practice checks are run by the compiler. You can run them in daily builds to catch objectionable practices that are unacceptable in shipping code. The features can also be used to author simple one-of tools to gather information about the application.

You can also use best practice rules to author simple tools that gather information about your application. The framework is built on top of a managed framework called XLNT (shorthand for X++ LaNguage Toolkit). You can use the framework to build custom tools that extract information from, and modify, X++ code. There are two types of best practice rules: rules that deal with metadata and rules that deal with source code.

Code Best Practice framework

The Code Best Practice Framework (CBPF) enables you to write your own tools for analyzing X++ source code. These rules diagnose things that you consider to be problems with X++ source code. This section describes the foundation of the Best Practice functionality. This information is helpful for understanding the later sections that describe creating your own rules in greater detail. It is also helpful to developers who want to code rules that are more complex than those demonstrated in this document. The CBPF API lets you focus on the rule you are expressing, without having to deal with infrastructure issues. You don't need to read tokens and piece them together to create something intelligible from them. Instead, the CBPF provides the following parts:

- A parser that builds an Abstract Syntax Tree (AST) from X++ source code.
- A pipeline that runs a sequence of passes over the X++ code.
- A number of prebuilt passes. The first pass is the parsing of the source code.
- Infrastructure to read metadata.

Because rules are based on ASTs, it is important to understand that concept before starting to write rules.

The parser and ASTs

The parser reads X++ code and produces an AST from it if it does not contain egregious syntax errors. The parser has a built-in error recovery scheme, so it can recover from most syntax errors reasonably well. When a syntax error happens, the parser will read symbols until it encounters a semicolon, and then try to building the AST by unstacking its state until it reaches a state where a semicolon is a correct symbol. In addition, the parser is able to suggest the correct set of symbols when a syntax error occurs. The parser is not intended to be directly interacted with by consumers of the API, but should be considered a black box that works without user intervention. As the parser recognizes the language constructs in the source code, it builds an AST. The AST consists of nodes that are abstractions of the X++ artifact they represent. We will illustrate the concept by showing a few AST nodes below:

```
public abstract class Statement : Ast
{
    /// <summary>
    /// Gets or sets the comments in the statement
    /// </summary>
    public string Comments { get; set; }

    public abstract string ToString(int indent);
}
```

The **Statement** class is abstract, because it doesn't make sense to instantiate a "statement". Only concrete derived classes, like **if** and **while** statements, can be instantiated. Since the Comments property is placed on this base class, it applies to all derived classes. In other words, all statements can have comments preceding the statement. The comments are accessible through the given property. There are many different kinds of statement in X++ and each one is described by a class derived in one or more steps from the abstract **Statement** class shown above. The following example shows the definition of a **while** statement.

```
public class WhileStatement : Statement
{
    /// <summary>
    /// Gets or sets the while condition.
    /// </summary>
    public Expression Condition { get; set; }

    /// <summary>
    /// Gets or sets the constituent while statement.
    /// </summary>
    public Statement Statement { get; set; }
}
```

The **while** statement consists of the condition (an expression) and the constituent statement, that is executed as long as the condition evaluates to true. The parser will maintain the source code positions where the represented artifact starts and ends (that is, its extent). As the ASTs are traversed, it may be useful to add information to the individual nodes. For example, every expression has a type. As the tree is traversed to diagnose type compatibility problems it becomes useful to be able to place that information on the individual node. Rather than having to modify the AST nodes for each requirement, there is a property collection that can be used to provide name/value pairs to each node. Each AST node has a **ToString** method that will return a high fidelity string representation of the node, which is useful in debugging scenarios.

The AstSweeper class

The AstSweeper applies a visitor pattern to the AST instance that it is given. The visitor pattern allows the programmer to separate the underlying data structure (that is, the AST) from the operations that the user wants to perform on the nodes (that is, the logic reasoning about the code). The **AstSweeper** class has a virtual method for each of the AST node types, and it will call them as directed by the structure of the AST. The following examples show how the sweeper works. Some details have been omitted for clarity.

```
/// <summary>The AST sweeper visits each node in the AST</summary>
/// <typeparm name="TReturn">The type returned by each of the sweeper methods</typeparm>
/// <typeparm name="TPayload">
/// The type of the payload passed to each of the sweeper methods
/// </typeparm>
public class AstSweeper<TReturn, TPayload> where TReturn : class
{
    protected virtual TReturn VisitStatement(TPayload o, Statement statement)
    {
        var compoundStatement = statement as CompoundStatement;
        if (compoundStatement != null)
        {
            return this.VisitCompoundStatement(o, compoundStatement);
        }

        var whileStatement = statement as WhileStatement;
        if (whileStatement != null)
        {
            return this.VisitWhileStatement(o, whileStatement);
        }
    }

    protected virtual TReturn VisitWhileStatement(TPayload o, WhileStatement statement)
    {
        this.VisitExpression(statement.Condition);
        this.VisitStatement(statement.Statement);

        return null;
    }
}
```

The name of the virtual method handling a particular AST node is the name of the AST class prepended with Visit. The parameters are the node to visit and a payload that may be passed to all the visitors as they are called. In this way, the sweeper will call the virtual method once for each and every one of the nodes in the AST that is passed to it in a depth-first traversal. The payload parameter can be used to pass information (for example, a symbol table) to each node as required. Developers will build classes derived from the AstSweeper class, overriding the methods of particular interest to them.

Example

Suppose you need to determine the percentage of parameter names starting with an underscore character, thus conforming to the X++ coding guidelines. You would then write a class deriving from the **AstSweeper** class with logic that calculated the number of parameters and the number of parameters starting with underscore. The following example shows this code.

```
public class ParameterCounter : AstSweeper<object, object>
{
    /// <summary>
    /// The parameter count maintained as the methods are encountered.
    /// </summary>
    public int ParameterCount { get; set; }

    /// <summary>
    /// The number of parameters that start with an underscore character.
    /// </summary>
    public int UnderscoredParameters { get; set; }

    /// <summary>
    /// Visits the method parameters.
    /// </summary>
    /// <param name="o">The payload. Not used in this scenario.</param>
    /// <param name="parameters">The list of parameters to visit.</param>
    /// <returns>The method parameters.</returns>
    protected override object VisitMethodParameters(object o,
        IEnumerable<ParameterDeclaration> parameters)
    {
        this.ParameterCount += parameters.Count();
        this.UnderscoredParameters += parameters
            .Where(p => p.Name.StartsWith("_")).Count();

        return null;
    }
}
```

In this case, the tally is maintained in the **ParametersCount** and **UnderscoredParameters** properties that are defined in the class scope. Another equally valid approach would be to pass this information into the payload that is passed to all the **Visit** methods. In most cases, the user should unconditionally call **super()** from the overridden method to make sure that the **Visit** methods are called for all nodes below the one being visited. In the case above, it does not make a difference so we opt to improve performance by pruning the AST tree traversal.

Writing code for Best Practice rules

To author a business rule:

1. Define the situation you want to diagnose in terms of properties of the AST. You will write **Visit*** methods that can do the analysis.
2. When the error condition has been found, a diagnostic message must be generated. There is an API that is used for this purpose; basically you need to write some boilerplate code for each diagnostic message.
3. You need to hook your new best practice rule into the rest of the framework, so the user can decide whether or not to include your rule and to run it if so directed.

Create a best practice rules project in Visual Studio

In this walkthrough, we imagine the following scenario:

- Some methods are adorned with an Author attribute that provides the name of the individual who wrote the code. The attribute is useful when finding who to point the finger at when stack traces containing that method appear.
- Since we have a significant turnaround of developers, the names of the developers listed cannot be static. We want to find the names that are used in the Author attributes, and match them against a list of names of current developers.

The author attribute class is defined as:

```
class AuthorAttribute extends SysAttribute
{
    private str theAuthor;

    public void new(str _author)
    {
        this.theAuthor = _author;
    }
}
```

In production code, we would put in documentation comments and assertions to validate key assumptions about parameter values etc. For the sake of clarity, we omit these steps in this walkthrough, where the code is written for clarity. Now that we have set the stage for what we want to achieve, we can start up Visual Studio and create a best practice rules project. Provide a meaningful name that properly conveys what the rules are intended to do: Visual Studio creates a project with some source code snippets and project references set up. You can save considerable time by using this source code as a starting point for your own code. The pre-canned example contains rules that prohibit the word “Microsoft” in any method names (presumably for copyright reasons) and a metadata-based rule prohibiting certain characters in names. Since we are not concerned with the metadata checks for now, you can delete the InvalidCharactersDiagnosticItem.cs and DemoMetadataCheck.cs files from the project. Also, since we are not interested in the Microsoft name check, go ahead and delete the content of the VisitMethod method in the DemoAST class. The first thing we need to do is to find out if there are one or more Author attributes for a particular method. You will notice that the Method type (that is passed as a parameter to the VisitMethod method) has an Attributes property of type AttributeList. Let's use it to see if any Author attributes are defined on this method:

```
protected override object VisitMethod(BestPracticeCheckerPayload payload, Method method)
{
    var names = new List<string>();
    foreach (var attribute in method.Attributes.Attributes)
    {
        if (string.Compare(attribute.Name, "Author",
            ignoreCase: true, culture: CultureInfo.InvariantCulture) == 0)
        {
            var authorNameLiteral = attribute.Parameters.First().Literal as StringAttributeLiteral;
            // The name contains quotes (either single or double). Get rid of those
            var authorName = authorNameLiteral.Value.Trim('\'', '"');
            names.Add(authorName);
        }
    }

    // More to come...
    return null;
}
```

At this point we have looped through any attributes, and collected a list of author names, that is names that are provided as the first parameters to the Author attributes. Now we need to compare the list against a list of acceptable authors, that we maintain in a static list. Whenever an author is provided that is not mentioned in the list we need to issue an appropriate diagnostic message. At this time, we have something like:

```
public class AuthorListRule : BestPracticeAstChecker<BestPracticeCheckerPayload>
{
    private static HashSet<string> authorlist = new HashSet<string>()
    {
        "Alan Kay", "John von Neuman", "C.A.R. Hoare"
    };

    public AuthorListRule() : base()
    {
    }

    protected override object VisitMethod(BestPracticeCheckerPayload payload, Method method)
    {
        var names = new List<string>();
        foreach (var attribute in method.Attributes.Attributes)
        {
            if (string.Compare(attribute.Name, "Author",
                ignoreCase: true, culture: CultureInfo.InvariantCulture) == 0)
            {
                var authorNameLiteral = attribute.Parameters.First().Literal as StringAttributeLiteral;
                // The name contains quotes (either single or double). Get rid of those
                var authorName = authorNameLiteral.Value.Trim('\'', '"');
                names.Add(authorName);
            }
        }

        foreach (var name in names)
        {
            if (!authorlist.Contains(name))
            {
                // TODO: Add a diagnostic message
            }
        }
        return null;
    }
}
```

In other words, we need to create a diagnostic message to let the user know about the transgression of the rule. As noted before, it is important to call the base implementation of your visitor, which will then call visitor methods for all the nodes that are contained in the method. However, in this case, we do not want to do any further processing once we have determined if the author attribute is on the list.

Add a class for the diagnostic message

The project already includes boilerplate code for an error message, so we will use that as our starting point to create the diagnostic message that will be returned if the rule is violated. Each message is implemented as a class of its own. Each error message may have any amount of contextual information encoded into it. In this case, the contextual information is the name of the author that is not found in the list. We will start by adding the message to the messages resource file: Open that file in the project and add a string to it. We will use the name (also known as the error moniker) AuthorNotCurrent. The '{0}' string is a placeholder for the contextual information, in this case the name of the author who is not in the list. In addition to the actual text that will appear in the error message, there is also a string containing a description of the rule. This information is shown in the best practice dialog within Visual Studio and is designed to help the user decide which rules to enable on the system. Create a class for the diagnostic message, and call it **AuthorNotCurrentDiagnosticItem.cs**. Add the following code inspired from the **NotAllowedWordDiagnosticItem** class.

```
namespace CompareAuthorsToList
{
    using System;
    using System.Collections.Generic;
    using System.Runtime.Serialization;
    using System.Xml.Linq;
    using Microsoft.Dynamics.AX.Metadata.XppCompiler;

    [DataContract]
    public class AuthorNotCurrentDiagnosticItem : CustomDiagnosticItem
    {
        private const string AuthorNotCurrentKey = "Author";
        public const string DiagnosticMoniker = "AuthorNotCurrent";

        public AuthorNotCurrentDiagnosticItem(string path, string elementType, TextPosition textPosition,
string author)
            : base(path, elementType, textPosition, DiagnosticType.BestPractices, Severity.Warning,
DiagnosticMoniker, Messages.AuthorNotCurrent, author)
        {
            this.AuthorNotCurrent = author;
        }

        public AuthorNotCurrentDiagnosticItem(Stack<Ast> context, TextPosition textPosition, string author)
            : base(context, textPosition, DiagnosticType.BestPractices, Severity.Warning, DiagnosticMoniker,
Messages.AuthorNotCurrent, author)
        {
            this.AuthorNotCurrent = author;
        }

        // Serialization support
        public AuthorNotCurrentDiagnosticItem(XElement element)
            : base(element)
        {
        }

        [DataMember]
        public string AuthorNotCurrent { get; private set; }

        /// <summary>
        /// Hydrate the diagnostic item from the given XML element.
        /// </summary>
        /// <param name="itemSpecificNode">The XML element containing the diagnostic.</param>
        protected override void ReadItemSpecificFields(System.Xml.Linq.XElement itemSpecificNode)
        {
            this.AuthorNotCurrent = base.ReadCustomField(itemSpecificNode, AuthorNotCurrentKey);
        }

        /// <summary>
        /// Write the state into the given XML element.
        /// </summary>
        /// <param name="itemSpecificNode">The element into which the state is persisted.</param>
        protected override void WriteItemSpecificFields(System.Xml.Linq.XElement itemSpecificNode)
        {
            this.WriteCustomField(itemSpecificNode, AuthorNotCurrentKey, this.AuthorNotCurrent);
        }
    }
}
```

The diagnostic message is ready for consumption. There is just one piece of bookkeeping that needs to be done; the rule needs to publish declaratively which diagnostics it potentially issues. Go back to your rule and modify the BestPracticeRule attribute to reflect the new diagnostic message:

```
[BestPracticeRule(
    AuthorNotCurrentDiagnosticItem.DiagnosticMoniker,
    typeof(Messages),
    AuthorNotCurrentDiagnosticItem.DiagnosticMoniker + "Description",
    BestPracticeCheckerTargets.Class)]
public class AuthorListRule : BestPracticeAstChecker<BestPracticeCheckerPayload>
{ // ... }
```

As you can see, there are four parameters specified for the **BestPracticeRule** attribute:

1. The rule moniker.
2. The type of the resource file holding the rule description. In this example, we are using the default resource file named. Messages, which created a class called Messages. We want the type of this class as the second argument.
3. The name of the string resource that contains the description of the rule. This name is the string called **AuthorNotCurrentDescription** that we added to the resource file above; it contains a human legible string to describe the rule. This string is used to describe the rule to the user in a best practice dialog within Visual Studio. In Visual Studio, select **Dynamics 365 > Best Practices Configuration** to view the dialog.
4. A description of the artifacts to check. In this case, the value specifies that the rule should only be applied to classes. Modify the description as needed by using one of the other literals in the **BestPracticeCheckerTargets** enumeration.

Instantiate the class that describes the diagnostic message and add it to the set of diagnostics:

```
foreach (var name in names)
{
    if (!authorlist.Contains(name))
    {
        // Create the custom error message, including
        // the contextual name information...
        var warning = new AuthorNotCurrentDiagnosticItem(
            this.Context, method.Position, name);

        // and add it to the set of reported messages.
        this.ExtensionContext.AddErrorMessage(warning);
    }
}
```

At this point, you have a complete best practice rule, ready to provide value in your organization. Build it and fix any errors.

Metadata based Best Practice rules

Until now we have been describing how to write rules that deal with code. In this section we show how to author rules that apply to metadata, not code. Classes that deal with metadata rules are derived from **BestPracticeMetadataChecker**. The derived instance receives an instance of the metadata describing the artifact that must be checked. You then use the APIs in the **Microsoft.Dynamics.AX.Metadata.Metamodel** to fetch further metadata as needed, and use LINQ queries over the metadata graphs. The template for best practice checks contains a class performing metadata checks as well as a code based one we discussed in the previous section. The mechanics involved in issuing diagnostic messages is the same as we covered above.

Install, run, and test your rule

When your code compiles cleanly, a DLL will be created. In order for the tooling to be able to pick up the new rule, this DLL must be installed before running it. Installing the DLL can be done in two ways:

- By using the button on the Best Practice configuration dialog. Click the **Install extension** button. You will be asked to point to the assembly file that contains your rule (that is, the DLL generated when you build the rule). Press OK, and the system will copy the DLL where it needs to be (see below).
- By manually installing the DLL into the C:\Packages\bin\BPExtensions folder.

If you want to debug your rule, you will find it useful to copy the .pdb file to the same directory as the assembly. After the DLL has been deployed to the target directory, Visual Studio needs to be restarted. After that, the rule is available for use. You may have to debug your rule to iron out any remaining kinks. In fact, stepping through your rule and inspecting the ASTs is valuable when you are getting started. To debug a rule, you need to know that the best practice rule is run by the **xppAgent** process; it is therefore not run within the context of VS itself. Make sure you have selected **Run best practice checks** in the Visual Studio Options dialog, in the **Finance and Operations** page. Otherwise, your check will not run. Set a breakpoint in the **VisitMethod** method, and then do a build of a model that has the new rule switched on as shown above for the Fleet management model. Attach your VS instance to the **xppcAgent** process. When you do a build your breakpoint will be hit, and you can start drilling into your code. You can see all the properties, the list of declarations and statements, and find out all the details about them.

Running rules in XppBp.exe

As described above the best practice rules are often run as part of the build of a project from Visual Studio, but there is also a dedicated command-line tool to run them. This tools is the **xppbp.exe** tool, and it is intended mainly for nightly build scenarios. Invoking it from the command line yields a useful overview of the command-line switches and arguments. Here are some useful examples:

- Run BP on all forms in a module: `xppbp -module:FleetManagement form:*`
- Run BP on specific elements: `xppbp -module:FleetManagement class:MyClass form:MyForm`
- Run BP on all items in the model (and only for this one model in the module): `xppbp -module:FleetManagement -model:FleetManagement –all`
- Run BP on all items in all models in the module: `xppbp -module:FleetManagement –all`
- Write the output to log files: `xppbp -module:FleetManagement -all -xmllog=Log.xml -log=Log.txt`

Application Explorer properties

2/18/2021 • 114 minutes to read • Edit Online

This topic describes the properties that appear in the Properties window of Microsoft Visual Studio for items in Application Explorer.

Many nodes in Application Explorer represent elements that have properties associated with them. You can read or modify these properties in the **Properties** window of Microsoft Visual Studio.

System and common properties

Most application objects in Application Explorer have a standard set of system properties. These system properties are read-only. You can use the **Properties** window to view the properties for any item in Application Explorer. To open the **Properties** window, right-click a node in Application Explorer, and then click **Properties**. On the **Categories** tab of the **Properties** window, many system properties are listed under the **Statistics** node. This article lists additional common properties that are repeated on many, but not all, Application Explorer nodes. The following table shows the system properties that are found on almost all Application Explorer nodes. All these system properties are read-only.

PROPERTY	DESCRIPTION
ChangedBy	The user who last changed the object (often the release version).
ChangedDate	The date when the object was last changed.
ChangedTime	The time when the object was last changed.
CreatedBy	The user who created the object.
CreationDate	The date when the object was created.
CreationTime	The time when the object was created.

The following table shows other common properties that are found on many, but not all, Application Explorer nodes.

PROPERTY	DESCRIPTION
ConfigurationKey	Specify the configuration key that controls access to or display of an element. If a user doesn't have access to the configuration key, the element isn't visible. Elements include pages, controls on pages, tables, and other elements.
LegacyID	An identifier element from an earlier version. During upgrade from a previous version, the old identifier is assigned to **LegacyID**. An installation-specific identifier isn't assigned, and business logic remains intact. This property isn't used for new elements.

PROPERTY	DESCRIPTION
NeededAccessLevel	The minimum access level that a user requires. This property is read-only.
Origin	The globally unique identifier (GUID) of an Application Explorer element. This property is used to identify elements during synchronization and in upgrade scenarios. It's a read-only property, and the value never changes after the system assigns it. No origin GUID value is duplicated anywhere in the system.
SecurityKey	This property is obsolete but is retained for reference in systems that were upgraded from an earlier version.

Base enum properties

The following table describes the properties that are available for enumerations.

PROPERTY	DESCRIPTION
AnalysisUsage	Specify the role of the enumeration in a cube. This setting is automatically propagated to all table fields that reference the enumeration. However, you can override the setting on a table field. The following options are available: • **Attribute** – A field that references the enumeration is a dimension attribute. • **None** – A field that references the enumeration isn't a dimension attribute.
ConfigurationKey	Specify the configuration key.
CountryRegionCodes	Specify the codes for the countries/regions where the view is applicable or valid. This property is implemented as a comma-separated list of International Organization for Standardization (ISO) country/region codes in a single string. The values must match data in the global address book. The client framework and application might use this property to enable or disable country/region-specific features.
DisplayLength	Specify the number of characters that are shown. The default value is **Auto**.
Help	Create a Help string for the field. The Help string is shown when the field is used on a page.
Label	Specify the label that is shown on pages and reports.
Model	Specify the model that the table is in. A model is a logical grouping of elements in a layer. Examples of elements include a table and a class. An element can exist in exactly one model in a layer. The same element can exist in a customized version in a model that is in a higher layer.

PROPERTY	DESCRIPTION
Name	Specify the enumeration name. An enumeration name must indicate either the possible enumeration values or the type of the enumeration value. Examples of enumerations that are named according to the possible values are **InclExcl** and **NextPrevious**. Examples of enumerations that are named according to the type of the enumeration value are **ArrivalPostingType** and **ListStatus**.
Style	Change the default appearance of the enumeration. The following options are available: • Combo box • Radio button
UseEnumValue	A value of **Yes** indicates that default values of the **EnumValue** property were modified. A value of **No** resets the **EnumValue** property to the default values.

Extended data type properties

Extended data type (EDT) properties are divided into the following groups, based on whether they are common to all EDTs or available only for certain base data types.

Properties that are common to all EDTs

PROPERTY	DESCRIPTION
Alignment	Change the alignment of the text. The available options are **Left**, **Right**, and **Center**.
AnalysisDefaultSort	Specify the default sort order for a field in a report model that has this EDT.
AnalysisDefaultTotal	Specify the aggregate function for a measure. Use this property when the **AnalysisUsage** property is set to **Measure**. The following options are available: • **Sum** – Return the sum of all the values in a set. • **Count** – Return the number of non-null items in a set. • **CountDistinct** – Return the number of distinct non-null items in a set. • **Min** – Return the minimum value in a set. • **Max** – Return the maximum value in a set. • **None** – No aggregate function is applied. • **Auto** – This option applies to derived EDTs. The value of the **AnalysisUsage** property for the parent EDT is used. You can override the aggregate function at the field level. In other words, you can change the aggregate function for the field by using the **AnalysisDefaultTotal** property for that field.

PROPERTY	DESCRIPTION
AnalysisGrouping	Specify whether a field that has this EDT is grouped by default when the field is added to a report by using Report Builder for Microsoft SQL Server Reporting Services (SSRS). This property is automatically set to **Discouraged** for currency amounts. For other fields that are unique, you should set this property to **Discouraged**.
AnalysisUsage	Specify the role of the EDT in a cube. This setting is automatically propagated to all table fields that reference the EDT. However, you can override the setting on a table field. The following options are available: • **Attribute** – A field that references the EDT is a dimension attribute. • **Measure** – A field that references the EDT is a measure. • **Both** – A field that references the EDT is both a dimension attribute and a measure. • **None** – A field that references the EDT is neither a dimension attribute nor a measure. • **Auto** – This option applies to derived EDTs. The value of the **AnalysisUsage** property for the parent EDT is used. **Note:** Data types that are based on enumerations can't be measures.
ArrayLength	This property is a read-only. The default value is **1**. To add array elements to the EDT, right-click the **Array Element** node, and then click **New Array Element**. The value of the **ArrayLength** property is increased to reflect this change.
ButtonImage	Specify the image that is shown when the EDT is used for a lookup button on a page. The following options are available: • **Arrow** • **Mail** – You can select this option for the **e-mail** type, for example. • **URL** – You can select this option for the **URL** type, for example. • **ThreeDots** (...) • **OpenFile** – You can select this option for the **FilenameOpen** and **FilenameSave** types, for example. • **Calendar** – You can select this option for date types, for example. The default value is **Arrow**.
CollectionLabel	Specify the label that is used to show the plural name of a field that has this EDT.
ConfigurationKey	Specify the configuration key for the EDT.

PROPERTY	DESCRIPTION
CountryRegionCodes	Specify the codes for the countries/regions where the menu is applicable or valid. This property is implemented as a comma-separated list of ISO country codes in a single string. The values must match data in the global address book. The client uses this property to enable or disable country/region-specific features.
DisplayLength	Specify the maximum number of characters that are shown on a page or report.
EnumType	Specify an enumerated data type. This property must be set for EDTs of the **enum** type.
Extends	Use this property to base the EDT on another EDT.
FormHelp	Specify the page to use when you perform a lookup from a field on a page.
HelpText	Create a Help string for the EDT. The Help string is shown when the type is used on a page.
ID	This property is read-only.
Label	Specify the label that is used for the type when the type is used on a page or report.
Model	Specify the model that the table is in. A model is a logical grouping of elements in a layer. Examples of elements include a table and a class. An element can exist in exactly one model in a layer. The same element can exist in a customized version in a model that is in a higher layer.
Name	Specify the name of the type. The name is used to refer to the type from X++.
PresenceClass	Specify the X++ class that is used together with the **PresenceMethod** property to return an instance of the **PresenceInfo** object.
PresenceIndicatorAllowed	Specify whether the control that references the EDT should use presence. The default value is **Yes**.
PresenceMethod	For the X++ class that is specified in the **PresenceClass** property, specify the X++ static class method that should be called by using a controls data value. This method returns an instance of the **PresenceInfo** object that contains the data that the Presence indicator requires.
ReferenceTable	Specify the table that is referenced by this EDT, and that has the primary key. In other words, this property indicates the primary key table that this EDT references.

PROPERTY	DESCRIPTION
Style	Change the default appearance of the EDT. The following options are available: • Auto • Combo box • Radio button

Properties that are available for only some base data types

Unless the following table specifies otherwise, you should leave all these properties set to **Auto**.

PROPERTY	TYPE THAT THE PROPERTY EXISTS FOR	DESCRIPTION
Adjustment	String	For strings of fixed length, specify whether the characters that are entered should be stored on the left side or the right side of the padding spaces. The available options are **Left** and **Right**. The default value is **Left**.
AllowNegative	IntegerInt64Real	Specify whether the field can accept negative values.
AutoInsSeparator	Real	Specify whether the system should automatically insert a decimal separator. For example, if you enter **2222**, the system automatically shows **2222.00**.
ChangeCase	String	Specify how text that is entered in a string control should be formatted. For example, the text can be formatted as all uppercase letters, or it can use title capitalization. **Note:** This property isn't supported for Enterprise Portal.
DateDay	DateUtcDateTime	Specify how the day should be shown.
DateFormat	DateUtcDateTime	Specify the layout of a date.
DateMonth	DateUtcDateTime	Specify how the month should be shown.
DateSeparator	DateUtcDateTime	Specify the separators between year, month, and day.
DateYear	DateUtcDateTime	Specify how the year should be shown.
DecimalSeparator	Real	Specify the decimal separator. When the default setting (**Auto**) is used, the decimal separator that is specified in the system setup is used.
DisplaceNegative	IntegerInt64Real	Specify whether to align negative numbers to the left.

PROPERTY	TYPE THAT THE PROPERTY EXISTS FOR	DESCRIPTION
DisplayHeight	String	Specify the number of lines to show at the same time when the EDT is shown on a page.
EnumType	Enum	Specify the base enum that is used to create the EDT.
FormatMST	Real	Specify master currency values should be formatted. The following options are available: • Auto • Yes • No The default value is **Auto**.
NoOfDecimals	Real	Specify the number of decimal places when a value is shown on a page or a report.
RotateSign	IntegerInt64Real	Select this option to reverse the sign for the number. In other words, change a minus sign (–) to a plus sign (+) or a plus sign to a minus sign.
ShowZero	IntegerInt64Real	Specify whether to show a field that has a value of 0 (zero) as an empty field. If a value of 0 in fields of this type means null/nothing, set this property to **No**.
SignDisplay	IntegerInt64Real	Specify whether to show the sign of a negative number, and also whether the sign should appear before or after the number. Typically, you should set this property to **Auto**. However, you can set it to **None** if the **DisplaceNegative** property is used.
StringSize	String	Specify the maximum size of the string.
ThousandSeparator	Real	Specify the symbol that is used to separate thousands.
TimeFormat	TimeUtcDateTime	Specify how times should be formatted.
TimeHours	TimeUtcDateTime	Specify whether to include hours.
TimeMinute	TimeUtcDateTime	Specify whether to include minutes.
TimeSeconds	TimeUtcDateTime	Specify whether to include seconds.
TimeSeparator	TimeUtcDateTime	Specify the separator that is used for times.

PROPERTY	TYPE THAT THE PROPERTY EXISTS FOR	DESCRIPTION
TimezonePreference	UtcDateTime	Specify the time zone to convert the value to from Coordinated Universal Time (UTC).

Perspective properties

In Application Explorer, under the **Data Dictionary** node, there is a **Perspectives** node. A perspective is a collection of tables and views that contain the measures and dimensions for a cube. The following table describes the properties that can be set for each perspective. For information about the system properties that are available for a perspective, see the "System and common properties" section. For information about the properties for tables that are associated with a perspective, see the "Table properties" and "Table field properties" sections.

PROPERTY	DESCRIPTION
ConfigurationKey	Specify the configuration key that is assigned to the perspective. The configuration key determines which configurations of a perspective are included for in report models that are generated.
HelpText	Create a string to use as a description for the perspective in a report model.
ID	Specify the identifier of the perspective.
Label	Specify the name that is shown for the perspective in a report model.
Model	Specify the model that the perspective is in. A model is a logical grouping of elements in a layer. An element can exist in exactly one model in a layer. The same element can exist in a customized version in a model that is in a higher layer.
SharedDimensionContainer	Specify whether to share items in the perspective. When this property is set to **Yes**, the items in the perspective are added to all other perspectives that are in the project, and no cube is created for the perspective. The default value is **No**.
Usage	Specify the materialization options for a perspective. The following options are available: • **AdHocReporting** – The perspective will be used to generate a transactional Semantic Model Definition Language (SMDL) model. • **OLAP** – The perspective will be used to generate a cube in a Microsoft SQL Server Analysis Services (SSAS) Business Intelligence project. • **Both** – The perspective will be used to generate both a transactional SDML model and a cube in an SSAS Business Intelligence project. • **None** – The perspective won't be materialized. The default value is **None**.

Table properties

This section describes the properties that appear in the **Properties** window for table elements in Application Explorer. Table elements are located under **Data Dictionary** > **Tables**.

Table properties

The following table describes the properties of table elements in Application Explorer.

PROPERTY	DESCRIPTION
Abstract	Specify whether the table supports inheritance. The default value is **No**. If the value is set to **Yes**, the table can't be a direct target of X++ SQL statements such as **update_recordset** and **select**. **Note:** This property is unavailable when the **SupportInheritance** property is set to **No**.
AnalysisDimensionType	Specify the type of dimension that is created, based on the setting of the **IsLookup** property. If the **IsLookup** property is set to **Yes**, the following options are available: • **Auto** – The table can contain both factual and dimensional data. The BI Wizard will extract dimensional data, and create dimensions and attributes. Factual data will be extracted to create measures. One child dimension is created that has attributes from the parent table. • **MasterInner** – An inner (full) join is used to create relationships with this table to the child table. Each record combination for this table and the child table are generated in the dimension. One child dimension is created that has attributes from the parent table. • **MasterLeftOuter** – A left outer join is used to create relationships with this table to the child table. Dimensions will have additional attributes, based on values in this table that can also be empty. One child dimension is created that has attributes from the parent table. • **Transaction** – The table should be used to generate only factual data (measures). You should use this option when a table contains only transactional data. One child dimension is created that contains only enumeration fields from the table. If the **IsLookup** property is set to **No**, the following options are available: • **Auto** – Table can contain both factual and dimensional data. The BI Wizard will extract dimensional data, and create dimensions and attributes, Factual data will be extracted to create measures. One parent and child dimension are created. • **MasterInner** – Not applicable. This option is the same as **Auto**. • **MasterLeftOuter** – Not applicable. This option is the same as **Auto**. • **Transaction** – The table should be used to generate only factual data (measures). You should use this option when a table contains only transactional data. One child dimension is created that contains only enumeration values from the table.

PROPERTY	DESCRIPTION
AnalysisIdentifier	Specify the field to use as the identifier for the dimension in an SSAS cube.
AOSAuthorization	Specify the type of operation that a user can perform on a table, depending on the user's permissions. When this property is set to **None**, no authorization check is performed.
CacheLookup	Specify how to cache the records that are retrieved during a lookup operation. This property exists only on tables that don't inherit from another table. On an inheritance root table, you can' set this property to **EntireTable** by using the Application Explorer **Properties** window. You must not use other techniques to assign this value to inheritance root tables. For example, don't use the **AOTsetProperty** method of the **TreeNode** class to assign this value.
ClusterIndex	Specify the cluster index. This property is used only for SQL optimization.
ConfigurationKey	Specify the configuration key for the table. Configuration keys let a system administrator enable and disable specific parts of an application.
CountryRegionCodes	Specify the codes for the countries/regions where the table is applicable or valid. This property is implemented as a comma-separated list of ISO country codes in a single string. The values must match data in the global address book. The client framework uses this property to enable or disable country/region-specific features.
CountryRegionContextField	Specify the field that is used to identify the country/region context. This property is related to the **CountryRegionCodes** property.
CreatedBy	Specify whether the system maintains the **CreatedBy** field for the records in a table. This field contains information about the person who created a record.
CreatedDateTime	Specify whether the system maintains the **CreationDate** and **CreationTime** fields for the records in a table. This field contains the date when a record was created.
CreatedTransactionId	Specify whether the system maintains the **CreatedTransactionId** field for the records in a table. This field contains information about the transaction that created a record.
CreateRecIdIndex	Specify whether an index on the **Record ID** field is created.
DeveloperDocumentation	Describe the purpose of a table, and explain how it's used in the program. Typically, a description contains no more than five sentences and is written as a single paragraph.

PROPERTY	DESCRIPTION
EntityRelationshipType	Classify a table according to common entity relationship (ER) data model notation. A table is classified as either an entity or a relationship. An entity represents an object, whereas a relationship represents an association between two objects.
Extends	Derive the table from the specified table. The value of this property is **null** when the **SupportInheritance** property is set to **Yes**.
FormRef	Specify the display menu item that is activated when a table is referenced. A display menu item is associated with a page. When you use a primary index field on a report, this page is available as a link in the report. You specify a primary index by using the **PrimaryIndex** property. If you leave this property blank, the system tries to display a page that has the same name as the table.
ID	The system-generated table ID.
IsLookup	For report models, use this property to specify whether the table information is incorporated into other tables that reference it when a report model is generated. For online analytical processing (OLAP) cubes, use this property to specify whether to generate a consolidated dimension or a distinct dimension. The following options are available: • **Yes** – Attributes from the table should be consolidated into the parent dimension (star schema). • **No** – A separate dimension should be generated for the table (snowflake schema).
Label	Specify the label for a table.
ListPageRef	Specify a display menu item that points to a page that can show a list of this record type.
Model	Specify the model that the table is in. A model is a logical grouping of elements in a layer. Examples of elements include a table and a class. An element can exist in exactly one model in a layer. The same element can exist in a customized version in a model that is in a higher layer.
ModifiedBy	Specify whether the system maintains the **ModifiedBy** field for the records in a table. This field contains information about the person who last modified a record.
ModifiedDateTime	Specify whether the system maintains the **ModifiedDate** field for the records in a table. This field contains the date when a record was last modified.
ModifiedTime	Specify whether the system maintains the **ModifiedDateTime** field for the records in a table. This field contains the date and time when a record was last modified.
Name	Specify the table name.

PROPERTY	DESCRIPTION
OccEnabled	Specify whether the optimistic concurrency mode is enabled for a table. When this mode is enabled, data isn't locked from future modification when it's fetched from the database. Data is locked only when the actual update is performed.
PreviewPartRef	Specify the info part or form part to use in the enhanced preview. An info part shows a collection of data fields from a specified query. It uses metadata to describe how the data appears. A form part represents a pointer to a page.
PrimaryIndex	Specify the primary index. Only a unique index can be selected. This property is used for database optimization and to indicate which unique index should be used as the caching key. If you don't specify a primary index, the unique index that has the lowest ID is used as the caching key.
ReplacementKey	Specify the fields to display as the identifier for data in some page controls.
ReportRef	Specify the output menu item that is activated when a table is referenced. An output menu item is associated with a report. When you use a primary index field on a report, this report is available as a link in the report. You specify a primary index by using the **PrimaryIndex** property.
SaveDataPerCompany	Specify whether the data for the current company is saved. If you set the property to **No**, data is saved without a company identifier (DataAreaId). **Note:** If the **SaveDataPerCompany** property on a table is set to **Yes**, the **SetCompany** property on a page design that uses the table as a data source must also be set to **Yes**. **Tip:** The status line shows the acronym for the company. Double-click the acronym to open a dialog box where you can change the company.
SaveDataPerPartition	A value that indicates whether the table has a system field that is named **Partition**. This property is intended to be read-only. If the table has a **Partition** field, each record is assigned to one partition. Each record is hidden from data access operations that are run under the context of other partitions.
SearchLinkRefName	Specify the name of the menu item that links to information on a website about a table record that is listed in the Enterprise Portal search results. If the **SearchLinkRefType** property is set to **URL**, select a menu item that links to a Web Part page that shows the table data. Forms and reports on Web Part pages can display data.
SearchLinkRefType	Specify the type of the menu item that links to information on a website about a table record that is listed in the Enterprise Portal search results.
SingularLabel	Specify the label that is used in a report model or a cube to show the singular name of items that are stored in the table.

PROPERTY	DESCRIPTION
SupportInheritance	When you set this property to **Yes**, you can set a value for other inheritance-related properties, such as **Extends** and **Abstract**. **Caution:** If you set this property to **Yes**, any fields on the table are dropped and must be created again.
SystemTable	Indicate whether a table appears as a system table. A table that appears as a system table can be filtered during export and import. System tables are always synchronized when you sign in. Therefore, this property might be useful for tables that you use as soon as you sign in.
TableContents	Specify how setup/parameter data can be reused from one customer to another. The following options are available: • **Not specified** – Use this option for most tables. • **Default Data** – Use this option for customer-independent data, such as postal codes, units, and time intervals. • **Base Data** – Use this option for customer-dependent data, such as calendars, groups, and parameters. • **Default+Base data** – Use this option for data where the local perception varies. For example, Chart of Accounts is customer-independent in Germany but is customer dependent in most other places.
TableGroup	Specify the group that the table belongs to. Table groups provide a method for categorizing tables according to the type of data that they contain. You can use table groups to define whether the system should prompt users when they update or delete data from the table on pages by using the table as the data source. When you export data, you can use table groups to filter records.
TableType	This property replaces the **Temporary** property that is found in Microsoft Dynamics AX 2009.
TitleField1, TitleField2	You can use this property in the following ways: • Add table field data to a form caption. • Show additional fields on a lookup page. The **TitleField1** property is also used when you activate the lookup list in a field on a page. The fields that you specify for the **TitleField1** and **TitleField2** properties can be merged with the key value. • Show field information in a tooltip.
TypicalRowCount	Specify the number of records that typically appear in a table. If the **AnalysisSelection** property isn't set, this property determines how records are selected by using Report Builder for SSRS. The setting of this property affects whether a drop-down list, a list box, or a filtered list box is used to select table records.
ValidTimeStateFieldType	Specify the type of date-time field that the system uses when it tracks data within time spans.

PROPERTY	DESCRIPTION
Visible	Specify the access rights when the table is used as a data source on a page or a report. If the table is used as a data source on a page, the access rights on the page can't exceed the access rights that are defined for the table.

Tables and report models

The following properties are related to report models that are used to add information to a report:

- AnalysisSelection
- AnalysisVisibility
- IsLookup
- SingularLabel
- TypicalRowCount

Table field properties

The following properties are related to report models that are used to add information to a report:

- AnalysisDefaultTotal
- AnalysisLabel
- AnalysisTotaling
- AnalysisUsage
- AnalysisVisibility
- CurrencyCode
- CurrencyCodeField
- CurrencyCodeTable

PROPERTY	DESCRIPTION
Adjustment	Specify whether the string field should be left-aligned or right-aligned when it's stored in the database. For example, if the 11-character string "hello world" is stored in a right-aligned field that has a **StringSize** setting of **40**, 29 space characters are stored as the prefix. **Note:** The **Adjustment** setting affects the search results when you search for a value in a table by using the >, <, >=, and <= relational operators. It doesn't affect the search results when you use the == operator. The **Adjustment** setting is ignored when the **StringSize** property is set to **(Memo)**.
AliasFor	Specify the table field that the field is an alias for.
AllowEdit	Specify whether users are allowed to modify data in an existing record on a page.
AllowEditOnCreate	Specify whether users are allowed to enter data in the field when a new record is created from a page.

PROPERTY	DESCRIPTION
AnalysisDefaultTotal	For report models, use this property to specify how field data is aggregated when an automatic total for the table is displayed in a report that is built by using SSRS and report models. The default value is **No**, which indicates that the field isn't automatically shown as a total. For OLAP cubes, use this property to specify the aggregate function for a measure. Use this property when the **AnalysisUsage** property is set to **Measure**. The following options are available: • **Sum** – Return the sum of all the values in a set. • **Count** – Return the number of non-null items in a set. • **CountDistinct** – Return the number of distinct non-null items in a set. • **Min** – Return the minimum value in a set. • **Max** – Return the maximum value in a set. • **None** – No aggregate function is applied. • **Auto** – This option applies to derived EDTs. The value of the **AnalysisUsage** property for the parent EDT is used.
AnalysisLabel	Specify the label to use as the caption in an SSAS cube for the table field. The label is applied to either a dimension attribute or a measure. This property is intended for situations where one of the following conditions is true: • The **Label** property isn't defined. • The **Label** property doesn't work as a caption for a dimension attribute or a measure in a SSAS cube.
AnalysisUsage	Specify the role of the field in a cube. The following options are available • **Attribute** – The field is a dimension attribute. • **Measure** – The field is a measure. • **Both** – The field is both a dimension attribute and a measure. • **None** – The field is neither a dimension attribute nor a measure. • **Auto** – The value of the **AnalysisUsage** property for the EDT or enumeration that the field is based on should be used.
ConfigurationKey	Set the configuration key for the field.
CountryRegionCodes	Specify the codes for the countries/regions where the table field is applicable or valid. This property is implemented as a comma-separated list of ISO country codes in a single string. The values must match data in the global address book. The client framework and application might use this property to enable or disable country/region-specific features.
CountryRegionContextField	Specify the field that is used to identify the country/region context. See the description of the **CountryRegionCodes** property.

PROPERTY	DESCRIPTION
ExtendedDataType	Specify the EDT to use for this field.
GroupPrompt	Specify a label that is used for the field when it appears in a group. **Tip:** You can use this property to help guarantee that a field label doesn't repeat text that appears in the label for a field group. For example, if a field group on a page is labeled **Customer**, don't include this text in the **GroupPrompt** property for fields that are included in the field group.
HelpText	Specify the Help string for the field. The Help string is shown when the field is used on a page.
ID	The system-generated field ID.
IgnoreEDTRelation	This property is used during the migration of EDT relations. When you migrate relations from an EDT node to a table node, you can skip an invalid relation for a given table field. To skip invalid relations, set this property to **Yes**. The default value is **No**.
Label	Specify a label for the field. This label will appear on pages and reports. Also see the description of the **AnalysisLabel** property earlier in this table.
Mandatory	Specify whether a user must add data to a field on a page. Set this property to **Yes** to indicate that the default or initialization value for each data type isn't acceptable for persistence into the database. The following list shows some default values that can't be used for mandatory fields on a page: • Empty isn't acceptable for a str (string) field. • The minimum date-time isn't acceptable for date-time fields such as date and utcdatetime. • A value of 0 (zero) isn't acceptable for numeric fields such as int, real, and enum. Finance and Operations doesn't support the semantics for the **null** value that is standard in most SQL database products. Field can't be nulled in the database. Therefore, the **Mandatory** property has nothing to do with the concept of a **null** value. **Caution:** A mandatory table field can have its **EnumType** property set to an enumeration. You might define a field as an enum type that includes an item that has the integer value **0**. In this case, **0** isn't an item that's available for selection on the page. The forms system automatically calls the **validateWrite** method to enforce the setting of the **Mandatory** property. However, the **Mandatory** property has no effect on the behavior of direct X++ SQL that inserts or updates the value of a table field. In your direct X++ SQL, you can include calls to the **validateWrite** method on your table buffer variable. Your buffer variable inherits the method from the **xRecord** class.

PROPERTY	DESCRIPTION
MinReadAccess	Specify the mode of the automatic authorization feature. Automatic authorization has two modes of operation: surrogate foreign key and lookup. If a table in a query is tagged for surrogate foreign key authorization, and the user doesn't have access to that table but hasn't been explicitly denied, view access is granted to the table. However, not all fields will be visible. The visibility is determined by the following rules: • If **MinReadAccess** is set to **No**, no access is granted to the field. • If **MinReadAccess** is set to **Yes**, view access is granted to the field. • Otherwise, view access is granted if the field is part of the natural key automatic identification group, if it's a title field, or if it's a system field. If a table in a query is tagged for lookup authorization, access is determined by the following rules: • If **MinReadAccess** is set to **No**, no access is granted to the field. • Otherwise, view access is granted to the field.
Model	Specify the model that the table field is in. A model is a logical grouping of elements in a layer. Examples of elements include a table or class. An element can exist in exactly one model in a layer. The same element can exist in a customized version in a model that is in a higher layer.
Name	Specify the name of the field.
RelationContext	Specify the mapping of a field to a specific table relation. Typically, this property is used in unit-of-measure scenarios to model data that is related to currency codes or quantities. The relation that is associated with the field can then be used to show a lookup of currency codes or quantities. There is no default value.
SaveContents	Specify whether the field data is saved in the database or treated as virtual field data. Virtual field data is calculated at run time when the field is displayed. This data has no physical representation in the database. **Tip:** Instead of virtual fields, you can use display and edit methods.
StringSize	Set the field length, in the number of characters. The maximum field length depends on the database. A value of **(Memo)** indicates that the field length is unlimited.
Type	Specify the base type of a field.
Visible	Specify whether the field should be visible in the user interface.

Table index properties

The following table describes the properties that are available for indexes on tables.

PROPERTY	DESCRIPTION
AllowDuplicates	If you set this property to **Yes**, the index can be non-unique. If you don't create at least one unique index, a unique index is created by combining the first index and the RecId.
AlternateKey	Specify whether this index is part of an alternate key. The index field must have a unique value in every record.
ConfigurationKey	Set the configuration key. An index field that is disabled through a configuration key is automatically removed from the index.
Enabled	You can use this property to disable the index.
ID	The internal identifier of the object.
Model	Specify the model that the table index is in. A model is a logical grouping of elements in a layer. Examples of elements include a table or class. An element can exist in exactly one model in a layer. The same element can exist in a customized version in a model that is in a higher layer.
Name	Specify the index name.
UniqueAcrossCompanies	This property is for internal Microsoft use only. The available values are **Yes** and **No**. The default value is **No**. The value of this property is ignored when the **AllowDuplicates** property is set to **No**. However, when **AllowDuplicates** is set to **Yes**, a value of **Yes** for **UniqueAcrossCompanies** can improve the performance of some cross-company queries. The performance improvement is caused by changes to the caching of data.
ValidTimeStateKey	Specify whether this index key is used to determine the valid time state relationship with the parent table. The default value is **No**. **Tip:** To enable this property, you must set the **AllowDuplicates** property to **No** and the **AlternateKey** property to **Yes**.
ValidTimeStateMode	Specify whether gaps are allowed between two date-effective records. The default value is **NoGap**. **Tip:** To enable this property, you must set the **AllowDuplicates** property to **No**, the **AlternateKey** property to **Yes**, and the **ValidTimeStateKey** property to **Yes**.

NOTE

Pages sort on the first index.

Table relation properties

List of properties

The following table describes the properties for a table relation in Application Explorer.

PROPERTY	DESCRIPTION
Cardinality	The number of times that each primary key value from the referenced table must occur in the foreign key column of the current table. For example, the **OneMore** value means one or more, but not zero. This value indicates that every parent key value must occur in the child table's foreign key column at least one time. A relation node under a SalesLine table might use the **OneMore** value when the business rule requires that every record in the parent SalesTable table relate to at least one item that is being sold. Currently, the **Cardinality** property is not used. However, future releases might use this property and the **RelatedTableCardinality** property.
CreateNavigationPropertyMethods	A value of **Yes** instructs the system to generate navigation methods on the table buffer class for each foreign key relation node.
EDTRelation	If the value is set to **Yes**, a software tool was used to migrate this relation to its current location from an old EDT relation.
EntityRelationshipRole	This property is used to clarify the semantics of a relationship that is defined on a table. A role name should be either a noun or a noun phrase. The role name should indicate the role of the associated table in relation to the associating object. Alternatively, the role name should be a short phrase that starts with a present-tense verb that indicates the role that the table plays in the relationship. Role names aren't required when the relationship is unambiguous.
Model	The model that this relation is part of.
Name	A descriptive name that you choose for the relation.
NavigationPropertyMethodNameOverride	Specify the name of the navigation method. If no value is specified, the navigation method uses the value from the **RelatedTableRole** property.
RelatedTableCardinality	Specify whether the foreign key field value in the current table can be **null** in some or all records of the current table. The following options are available: • **ZeroOne** means zero or one. This value indicates that the foreign key field in a child record can be **null**. • **ExactlyOne** indicates that the foreign key field can't be **null** in any child record.

PROPERTY	DESCRIPTION
RelatedTableRole	Enter a text value to describe the purpose of the referenced parent table in this relationship. When a table has only one relation that references a given parent table, you can use the name of the parent table. Sometimes, a table has more than one relation to a given referenced parent table. this case, the value of the **RelatedTableRole** property should describe the relation well enough to distinguish the relation's purpose from the other relation to the same parent table. The value of this property can be used as the value of the **JoinRelation** property of a data source relation under an Application Explorer query. In standard cases, this usage is recommended, because it reduces denormalization. This property interacts with the **UseDefaultRoleNames** property.
RelationshipType	Select a value that describes the subtle relationship between two tables. For example, the **Composition** value indicates that the child record can't meaningfully exist unless it's related to a specific parent record. The record for the fourth floor in the Floor table can't exist unless it references a record in the parent Building table. **Note:** The DeleteActions should be compatible with this property setting. For a Composition relationship, the DeleteActions should include delete cascade behavior. Currently, the **RelationshipType** property is not used. However, a future release might use this property.
Role	Specify a name that describes the meaning or role of the relation. For example, one relation to a Department table could track the department that the employee currently belongs to. Another relation could track the department that the employee has requested a transfer to. Although both these relations are relations to the Department table, they fill different roles. As the value of this property, it's a good idea to join the names of the child table and parent table by using an underscore (_) character. For example, enter **SalesTable_SalesLine**. This property interacts with the **UseDefaultRoleNames** property.
Table	The table that the relation refers to.
UseDefaultRoleNames	A value of **Yes** indicates that the system must generate default values for the **Role** and **RelatedTableRole** properties. Even when this property is set to **Yes**, the values for that are generated for **Role** and **RelatedTableRole** don't appear in the **Properties** window. Additionally, the **TreeNode** class doesn't use the generate values. However, the **DictRelation** reflection class does use the generated values.

PROPERTY	DESCRIPTION
Validate	A value of **Yes** indicates that when a page inserts a record into the child table, the insertion is rejected unless the related record exists in the referenced parent table. Additionally, when a page deletes a record from the parent table, the deletion is either rejected or cascades to the related records in the child table. Set the value to **No** when the **RelationshipType** property is set to **Link**. You might also set the value to **No** in special temporary cases, such as during some upgrade scenarios. When the value is set back to **Yes**, no validation occurs for records that were inserted or deleted while the value was **No**. **Caution:** A value of **Yes** for the **Validate** property doesn't prevent direct X++ SQL data operations from deleting parent records or inserting child records that violate the integrity of foreign key data.

> **NOTE**
>
> When the **SaveDataPerCompany** property is set to **Yes** for both tables, the system adds the **DataAreaId** field to each relationship.

RelatedTableRole and query JoinRelation

This section describes how you can use the **RelatedTableRole** property to simplify the creation of a new query. If you enter an explicit value for the **RelatedTableRole** property on a table relation, you can use that value to populate the **JoinRelation** property on a data source relation under a **Queries** > **MyQuery** node in Application Explorer. You can use this method to specify the fields of the join in one location. If the join fields ever change, you must update the join in only one location. Before you can set a value for the **JoinRelation** property, you must delete the values of the **Field** and **RelatedField** properties.

CreateNavigationPropertyMethods and RelatedTableRole

When you set the **CreateNavigationPropertyMethods** property to **Yes** on a table relation, the system generates navigation methods for the table buffer class. A navigation method links two table buffer instances by using their foreign key relationship. The **UnitOfWork** class is one area where this navigation linkage is used. The name of a navigation method is copied from the value of the **RelatedTableRole** property on the table relation. This behavior is used when the **RelatedTableRole** value is explicitly set in the **Properties** window, and when the system generates the **RelatedTableRole** value because the **UseDefaultRoleNames** property is set to **Yes**. The property values generate the following navigation method on the child CustTable buffer. Most directly, the navigation method name is copied from the value of the **RelatedTableRole** property.

```
public final CustBankAccount BankAccounts([CustBankAccount relatedTable])
```

NavigationPropertyMethodNameOverride property

The following list describes cases where you must override the name that the system generates for a navigation method on a table buffer class:

- The table class already has a method name that matches the values of the **RelatedTableRole** property.
- The value of the **RelatedTableRole** property exceeds the maximum length that can be used for a method name.

In these cases, you must choose a valid name for the navigation method and assign that name as the value of the **NavigationPropertyMethodNameOverride** property on the table relation.

Understanding the RelationshipType enumeration

When you add a node under table relations, you can set the value of the **RelationshipType** property for the new relation. The list of possible values for the **RelationshipType** property is the list of elements in the **RelationshipType** enumeration. This section describes the meaning of each element in the **RelationshipType** enumeration.

Description of elements

The following table describes the elements of the **RelationshipType** property.

ELEMENT NAME	DESCRIPTION	AUTOMATIC INFERENCE
NotSpecified	This element is often the default value of the **RelationshipType** property.	When the **RelationshipType** property has a value of **NotSpecified**, the system infers an appropriate value. The system infers the value in the following sequence: 1. Specialization 2. Link 3. Composition 4. Aggregation 5. Association For example, if the criteria for both Composition and Aggregation are met, the system infers Composition, because Composition occurs earlier in the list.
Specialization	This element applies only to table inheritance, to relationships between base and derived tables.	The system sets the **RelationshipType** property to **Specialization** whenever table inheritance is involved.
Link	This element is a non-relational relationship. It requires that the **Validate** property be set to **No**. This type of relationship supports navigation between pages that list many records from a table and pages that provide detail fields for one record from the table.	Link is used only to support the migration of EDT link relations during upgrade from earlier versions. Migration tools create this type relationship, but you must not.
Composition	This element is a stronger type of Aggregation relation. A table must not have more than one Composition relation. For example, a building is composed of rooms, and a given room can't exist in more than one building.	If the criteria for Composition are met, but you manually assign a value of **Aggregation** or **Association**, the system leaves the value as **Aggregation** or **Association**.

ELEMENT NAME	DESCRIPTION	AUTOMATIC INFERENCE
Aggregation	This element is appropriate when the child table is considered subordinate to the entity of the parent table.	The system infers Aggregation when one of the following conditions is true: • The parent table has a delete action node that is defined to use this relation node. • Any foreign key field for this relation in the child table has the **Mandatory** property set to **Yes**. If the criteria for Aggregation are met, but you manually assign a value of **Association**, the system leaves the value as **Association**.
Association	This element is the concept of a standard foreign key.	You must set the **RelationshipType** property to **Association** if the system doesn't set the value of the property to anything, and if both Aggregation and Composition are inappropriate.

View properties

The properties for views are the same as the properties for tables. However, because views are read-only, most of their properties can't be changed. Some of these properties have fixed values, and some are inherited from the data sources that are used in the query that defines the view. The following properties for views are related to data analysis when you're using SSRS. All these properties can be changed.

- AnalysisVisibility
- AnalysisSelection
- TypicalRowCount
- IsLookup
- SingularLabel

The following table describes the properties that can be set for a view.

PROPERTY	DESCRIPTION
AOSAuthorization	Use this property to specify which data access operations require verification of user permissions.
CacheLookup	The record cache level for the table.
ClusterIndex	The cluster index of the table, if there is a cluster index.
ConfigurationKey	Set the configuration key for the view.
CountryRegionCodes	Specify the codes for the countries/regions where the menu is applicable or valid. This property is implemented as a comma-separated list of ISO country codes in a single string. The values must match data in the global address book. The client uses this property to enable or disable country/region-specific features.

PROPERTY	DESCRIPTION
CountryRegionContextField	Specify the field that is used to identify the country/region context. See the description of the **CountryRegionCodes** property.
DeveloperDocumentation	Describe the purpose of a view, and explain how it's used in the program. Typically, a description contains no more than five sentences and is written as a single paragraph.
EntityRelationshipType	Classify a view according to common entity relationship (ER) data model notation. A view is classified as either an entity or a relationship. An entity represents an object, whereas a relationship represents an association between two objects.
FormRef	Specify the default page for the view. The default page is the page that is shown when the user activates Jump to Main Table by using the shortcut menu for a field on a page. The page is referenced through a menu item of the **Display** type. If you leave this property blank, MorphX tries to activate a page that has the same name as the table that you're referring to.
ID	The internal identifier of the object.
Label	Specify a user-friendly name for the view.
ListPageRef	Specify a display menu item that points to a page that can show a list of this record type.
Model	Specify the model that the view is in. A model is a logical grouping of elements in a layer. An element can exist in exactly one model in a layer. The same element can exist in a customized version in a model that is in a higher layer.
Name	Specify the name of the view. This name is used when you refer to the view from the X++ language.
PreviewPartRef	Specify the info part or form part to use in the enhanced preview. An info part shows a collection of data fields from a specified query. It uses metadata to describe how the data appears. A form part represents a pointer to a page.
PrimaryIndex	Specify the primary index of the view. Only a unique index can be selected. This property is used for database optimization and to indicate which unique index should be used as the caching key. If you don't specify a primary index, the unique index that has the lowest ID is used as the caching key.
Query	Specify the query that is the source of data for the view. You can use this property instead of adding data sources directly to the view.
ReportRef	The name of the default report for the table.

PROPERTY	DESCRIPTION
SaveDataPerCompany	Set this property to **Yes** for company-specific tables. Set it to **No** if the data is related to cross-companies, installation, a database, Application Explorer, tracing, or OLAP. For example, the SysTraceTable or OLAPServerTable table specifies whether data should be saved for that table on a per-company basis, or whether the data should be available without any company affiliation. If the **SaveDataPerCompany** property on a table is set to **Yes**, that table has a **DataAreaId** column that contains the company identifier. If the table property is set to **No**, the **DataAreaId** column is removed from the table.
SaveDataPerPartition	A value that indicates whether the view has a system field that is named **Partition**. This property is intended to be read-only. If the view has a **Partition** field, each record is assigned to one partition. Each record is hidden from data access operations that are run under the context of other partitions.
SearchLinkRefName	The name of the menu item that links to information on a website about a table record that is listed in the Enterprise Portal search results.
SearchLinkRefType	The type of the menu item that links to information on a website about a table record that is listed in the Enterprise Portal search results.
SystemTable	A value that indicates whether a table is a system table. System tables can be filtered during export and import, and are always synchronized when you sign in. Therefore, this property might be useful for tables that you use as soon as you sign in.
TableContents	Specify how setup/parameter data can be reused from one customer to another. The following options are available: • **Not specified** – Use this option for most tables. • **Default Data** – Use this option for customer-independent data, such as postal codes, units, and time intervals. • **Base Data** – Use this option for customer-dependent data, such as calendars, groups, and parameters. • **Default+Base data** – Use this option for data where the local perception varies. For example, Chart of Accounts is customer-independent in Germany but is customer dependent in most other places.
TableGroup	Specify the group that the view belongs to. Table groups categorize tables and views according to the type of data that they contain. Views can belong to the same table groups as a table.

PROPERTY	DESCRIPTION
TitleField1, TitleField2	The information that is shown in the window caption for the view. The caption is constructed from the following elements: • The **TitleField1** label, followed by colon (:) and a space • The value of the current record in the column that is used for **TitleField1**, followed by a comma (,) • The value of the current record in the column that is used for **TitleField2**
ValidTimeStateEnabled	Specify whether the view supports the valid time state feature of the underlying table. The default value is **No**. You can set this property to **Yes** only if both the following conditions are true: • The underlying table is a valid time state table. • The view has **ValidFrom** and **ValidTo** in its **Fields** list.
Visible	Specify the access rights when the table is used as a data source on a page or a report. If the table is used as a data source on a page, the access rights on the page can't exceed the access rights that are defined for the table.

Data set properties

This section contains descriptions of the properties on data set elements in Application Explorer. The **Data Sets** node is a high-level node in Application Explorer. Data sets are used to access data in Enterprise Portal.

Description of properties

The following table describes the properties that are available on data set nodes in Application Explorer.

PROPERTY	DESCRIPTION
Name	Set the name of the data set.

Data Sources properties

The following table describes the properties for the **Data Sources** node of the data set.

PROPERTY	DESCRIPTION
ChangeGroupMode	Specify how changes to the data sources are committed. The following options are available: • **None** – The changes to any data source for the data set are committed independently of changes to the other data sources. • **ImplicitInnerOuter** – All the data sources that are inner-joined or outer-joined work as a single unit. All changes are committed successfully, or they are rolled back if an error occurs.

Data set data source properties

The following table describes the properties that are available for data set data sources.

PROPERTY	DESCRIPTION
AllowCheck	Specify whether security checks occur before the data set is accessed. The following options are available: • **Yes** – The user's read permissions are verified before the data set is accessed. • **No** – The user's read permissions are verified only after the data set is accessed. No data is retrieved if the user lacks sufficient permission for the underlying data sources. **Yes** is the default value and is usually recommended.
AllowCreate	Specify whether users can create new records in the data source (that is, in the table for the data source).
AllowDelete	Specify whether users can delete records in the data source (that is, in the table for the data source).
AllowEdit	Specify whether users can modify the data. **Tip:** You can set the **AllowEdit** property for the whole data source here. The same property also exists on each field in the data source, so that you can prohibit modifications for individual fields.
AutoNotify	This property isn't used for data sets.
AutoQuery	This property isn't used for data sets.
AutoSearch	This property isn't used for data sets.
CounterField	Specify one of the fields in the data source as a counter for the data set. The field must be an index on the underlying table for the data source, and it must be of the **real** type. This property helps guarantee that a record that is inserted in a data set has a line number that corresponds to the actual sequential position in the data. For example, if a new line is inserted between lines 3 and 4, the new line becomes line number 3.5.
CrossCompanyAutoQuery	Specify whether the data source retrieves data from more than one company database.
DelayActive	Use this property to delay execution of the active method for the data source. If you set this property to **Yes**, the active method is activated only after a delay of 20 milliseconds. When a user scrolls through a data source, the active method isn't called on every record. Instead. it's called only on the final record that the user selects. **Tip:** The **DelayActive** property is particularly useful when two data sources are linked (that is, when the **LinkType** property is set to **Delayed**). This property is part of the AutoJoin system.

PROPERTY	DESCRIPTION
Index	Set the index that is used to specify a sorting order. You can choose any of the indexes on the table. If you specify an index in this manner, it's used as an index hint on each query to the database. The index specifies both an access path and a sort order for the records in the data set, based on this data source. The initial sort order for the records is prioritized in this manner: 1. If sort fields are added to the data source query, the sort specification is used. 2. If an index is specified in the **Index** property on the data source, the sort order that is implicitly specified in that index is used. 3. If the data source is autojoined with another data source, the system finds the most appropriate index for this join and then sorts the data according to that index. 4. If nothing else is specified, the sort order that is implicitly specified in the first index (the index that has the lowest ID) on the table that is used in the page data source is used. When no index hints are specified, the database management system locates an applicable access path. This access path is based on the information in the query that is supplied. The user can change the sort order for a page by using the query dialog box.
InsertAtEnd	Specify whether a new record is created when the user moves focus past the last record in the table.
InsertIfEmpty	Specify whether a blank record is inserted if there are no records in the table. If you set this property to **No**, you must manually create a new record.
JoinSource	Use this property to join two data sources. Set this property when two or more tables are used as the data source, and you want to join them.
LinkType	Use this property to maintain an active link between two data sources. When focus changes in the first data source, the corresponding record or records in the second data source are selected. For example, a customer table and a table of transactions is used for each customer. When the user scroll from one customer to the next, the transaction list is automatically updated to show transactions for the current customer. Set this property to **Delayed** for the outer (externally linked) data source. The linked data source is updated only after a delay of 100 milliseconds. This delay helps guarantee that the linked data source isn't updated while the user is scrolling through a data source. The update occurs only after the user finally focuses on a record. This property is part of the AutoJoin system.
Name	Set the name of the data source. This name should be the same as the name of the underlying table.

PROPERTY	DESCRIPTION
OnlyFetchActive	Specify whether to fetch all fields in the data source or only those fields that are used by the data set. When this property is set to **Yes**, records can't be deleted from the data set. This restriction helps preserve data integrity, because it helps guarantee that a delete operation is never tried on incomplete records.
OptionalRecordMode	Specify the create and delete behavior for records on an outer-joined table. The following options are available: • **ImplicitCreate** – When no record is saved in the database, create an outer-joined record and joined tables as soon as the parent record becomes active. If the outer-joined record or its children aren't changed, they will be deleted when the parent record is no longer active. • **ExplicitCreate** – When no record is saved in the database, treat this record as disabled until the user explicitly triggers creation by using the **Optional Record** check box. When the record exists, clearing the check box will delete this record. • **None** – No special create or delete behavior occurs for an outer-joined record.
StartPosition	Specify whether the first record or the last record should be the current record when the data set is accessed.
Table	Set the table that is used as the data source.
ValidTimeStateAutoQuery	Specify the types of queries for date effectivity (**AsOfDate** or **DateRange**).
ValidTimeStateUpdate	Specify the types of updates for an existing date-effective record. The following options are available: • **CreateNewTimePeriod** – On the record that is becoming the previous record, the **ValidTo** date field is set to a date that is no later than the current date. In the same transaction, the new current record has its **ValidFrom** field set to immediately after **ValidTo** date of the previous record. • **Correction** – The **ValidFrom** or **ValidTo** value of existing rows must be modified to keep the date-effective data valid after the record set is updated. • **EffectiveBased** – Records in the past can't be edited. Records that are currently active are edited in a manner that resembles CreateNewTimePeriod mode. Future records are edited in a manner that resembles Correction mode. The default value is **CreateNewTimePeriod**.

Form properties

This section describes the properties that you set on forms in Application Explorer. To provide a uniform application interface, many properties have **Auto** values. You can create forms by using a drag-and-drop operation and then manually setting several properties. To specify the name of a form, you set the **Name** property in the **Properties** window for the form. All other properties on the top-level node for the form are

system properties and are read-only.

Form design properties

Most properties on the **Design** node for a form also exist on the individual controls. Examples include the **Width** and **Height** properties. However, when you set a property on the **Design** node instead of setting it on a control, the setting affects the whole form. A few properties exist only on the **Design** node. The following table describes these properties.

PROPERTY	DESCRIPTION
AlignChild	Specify whether a control within a group follows the **AlignChildren** property setting for the group or for the overall form design. For example, **AlignChildren** is set to **Yes** on the **Design** node for the form, but you don't want a particular group to be arranged together with the other groups. In this case, set **AlignChild** to **No** for that group.
AlignChildren	Align the child controls within a container.
AllowDocking	Specify whether a form can be attached to the client workspace. The default value is **No**.
AllowFormCompanyChange	Specify whether the form supports company changes when it's used as a child form with a cross-company dynamic-link library (DLL). The default value is **No**.
AllowUserSetUp	Specify whether a user can move controls on a form and can change the value of control properties. This property is also found on the design of a form. The following options are available: • **No** – Users can't customize any controls in this container. • **Restricted** – Users can change properties of individual controls, but they can't move controls. • **Yes** – There are no restrictions on the user setup. The default value is **Yes**. **Caution:** Full user setup isn't allowed if any of the parent containers for the control have restrictions on the user setup level. The **AllowAdd** property on form data sources determines whether a user can add a field to a form.
AlwaysOnTop	Specify whether the form always appears on top of other windows in the z-order. The default value is **No**.
ArrangeMethod	Specify whether to arrange child field groups in columns or in rows.
ArrangeWhen	Specify when the controls in the container should be arranged. The following options are available: • Startup • On demand • Never • Default • Auto The default value is **Startup**.

PROPERTY	DESCRIPTION
BackgroundColor	Specify the color that is used for the background of the control. To make the background opaque or transparent, use the **BackStyle** property.
BottomMargin	Set the bottom margin of the form in pixels. The default value is **Auto**.
Caption	Specify the heading for grouped controls. Use a label for this property.
ColorScheme	Specify the color palette for the control. To change the color palette for the whole form, set the **ColorScheme** property for the largest container, and keep the default values for the individual controls.
Columns	Specify the number of columns that show the information. **Caution:** Field groups on the underlying table are never split into more than one column.
ColumnSpace	Set the amount of space between columns in container controls.
DataSource	Specify the table that data in the control comes from. To set a particular field within the table, use the **DataField** property. If the control opens another form, relations between the data source for the control, as specified by this property, and the data source on the other form help guarantee that records in the second form are dynamically selected. For example, a customer is selected in one form, and the control opens a form that shows customer transactions. In this case, the second form shows a range of customer transactions that apply to the current customer. **Caution:** If you set the **DataSource** and **DataField** properties, their settings override any settings for the **DataMethod** or **ExtendedDataType** properties.
Font	Change the font properties for the control by using the **Font** dialog box. Use the dialog font to specify the font, font style, and font size.
Frame	Specify the frame style that the form uses.
Height	Specify the height of the form or control in pixels.
HideIfEmpty	Use this property to hide a container control if it's empty. This property has no affect if the **Width** and **Height** properties of the container are set to **Auto**, because the size of the control is 0 (zero) in this case.
HideToolBar	Hide form-specific buttons on the toolbar.

PROPERTY	DESCRIPTION
ImageMode	Define how the bitmap that is specified by the **ImageName** property appears in a control. The following options are available: • Normal • Size to fit • Side by side • Center The default value is **Normal**.
ImageName	Specify the image that is shown for a control. You can select only .bmp files. To use one of the resource files, use the **ImageResource** property instead.
ImageResource	Use one of the images from the image resource file as the image for a control. Specify the ID of the image. You can select only an image from the integrated resource file. To use another file type, use the **ImageName** property.
LabelFont	Change the font for the text that is supplied in the **Label** property.
Left	Change the position of the upper-left corner of the form. There are several predefined settings. You can also specify an exact position in pixels. The following predefined settings are available: • Auto (left) • Auto (right) • Left edge • Right edge • Center The default value is **Auto (left)**.
LeftMargin	Change the default left margin of the form. The margin is specified in pixels.
MaximizeBox	Specify whether to include the maximize box in the upper-right corner of the enclosing window. The default value is **Yes**.
MinimizeBox	Specify whether to include the minimize box in the upper-right corner of the enclosing window. The default value is **Yes**.
Mode	Specify the data entry mode for the form.
Model	Specify the model that the form is in. A model is a logical grouping of elements in a layer. An element can exist in exactly one model in a layer. The same element can exist in a customized version in a model that is in a higher layer.
RightMargin	Change the default right margin of the form. The margin is specified in pixels.
SaveSize	Set this property to **Yes** to save the size of the form.

PROPERTY	DESCRIPTION
ScrollBars	Specify whether scroll bars are enabled in the form.
SetCompany	Cause the system to change the company when the form receives focus. **Note:** If the **SaveDataPerCompany** property on a table is set to **Yes**, the **SetCompany** property on a form design that uses the table as a data source must also be set to **Yes**.
StatusBarStyle	Specify how the status bar appears in a form. Use this property to hide the status bar, show only Help information, show status bar elements according to the **WindowType** setting, or always show the full status bar. **Note:** Forms that have a WindowType setting of **ListPage**, **ContentPage**, or **Workspace** ignore this property.
Style	Specify the style of the form. This property controls the form design pattern that is used for the form. The following options are available: • Auto • DetailsFormMaster • DetailsFormTransaction • Dialog • DropDialog • FormPart • ListPage • Lookup • SimpleList • SimpleListDetails • TableOfContents The default value is **Auto**.
TitleDataSource	Specify the data source to use in the form caption.
Top	Change the position of the top of the form. There are several predefined settings. You can also specify an exact position in pixels. The following predefined settings are available: • Auto • Top edge • Bottom edge • Center The default value is **Auto**.
TopMargin	Set the top margin of the form in pixels. The default value is **Auto**.
UseCaptionFromMenuItem	Specify whether to replace the form caption with the label from the calling menu item. This property enables the caption of the form to be changed when the form is opened. The default value is **No**.

PROPERTY	DESCRIPTION
ViewEditMode	Specify whether the form opens in read-only mode or as a form that allows you to change fields. The following options are available: • **View** – Open the form as read-only. • **Edit** – Open the form in edit mode. • **Auto** – Open the form in the appropriate mode. The default value is **Auto**.
Visible	Use this property to hide the form. **Caution:** You can't use the **Visible** property to enforce access restrictions. The user can change the visibility for the controls in the **Form Setup** dialog box. To enforce access restrictions, use the **Enabled** and **NeededAccessLevel** properties instead.
Width	Change the width of the form in pixels.
WindowResize	Specify whether the form can be resized.
WindowType	Specify the type of window.
WorkflowDataSource	Set the root data source for the workflow on a form. The root data source that you specify should be the same root data source that is specified in the query that used for the **Document** property on the workflow template.
WorkflowEnabled	Set this property to **Yes** to enable the workflow menu bar on the form. The default value is **No**.
WorkflowType	Specify the workflow type, which determines the following items and behaviors: • The workflow document to use. The workflow document exposes calculated fields and identifies the query that exposes data fields for the workflow. • Whether the user can configure tasks and approvals. • The workflow categories to use when a workflow type is assigned to a specific module. • Menu items and event handlers.

Help document set properties

A document set is a collection of Help documentation that is associated with a workspace. When you publish a content element, you use metadata to add your content element or table of contents information to a document set. To manage the relationship between a workspace and a document set, Application Explorer includes a node that is named **Help Document Sets**. Each document set in the **Help Document Sets** node includes a collection of properties. You edit these properties when you add a new document set or change the relationship between a document set and a workspace. **Caution:** A workspace can be associated with only one document set. Although Application Explorer lets you add a new document set and associate it with a workspace, you will no longer see documentation from the document set that you replaced. Typically, you use **UserDocumentation** as the document set for any content element or table of contents entries that you publish to the Help server. The following table describes the properties for a document set in the **Help Document Sets** node of Application Explorer.

PROPERTY	TYPE	DESCRIPTION
DocumentSetName	String	A name that uniquely identifies the document set. The name is limited to 40 characters and must not contain whitespace. Use the value of this property when you set the value of the **DocumentSets** metadata element in a content element or table of contents file.
DocumentSetDescription	String	The text or label to display for the document set. This value appears in the **Search content from** list of the **Options** menu of the Help viewer.
AddToApplicationHelpMenu	Boolean	Set this property to **Yes** if you want the document set to appear on the **Help** menu of the application workspace.
AddToDeveloperHelpMenu	Boolean	Set this property to **Yes** if you want the document set to appear on the **Help** menu of the developer workspace.
UserDocumentSet	Boolean	Set this property to **Yes** to associate the document set with the application workspace. If you set this property to **No**, you can't view the context-sensitive (F1) Help that was published by Microsoft.
DeveloperDocumentSet	Boolean	Set this property to **Yes** to associate the document set with the development workspace. If you set this property to **No**, you can't view the context-sensitive (F1) Help that was published by Microsoft.
ContentLocation	Enumeration	An enumeration value that specifies where to retrieve documentation. The following table describes the enumeration.

VALUE	LABEL	DESCRIPTION

PROPERTY	TYPE	DESCRIPTION		
		VALUE	LABEL	DESCRIPTION
		1	Help server	The documentation is stored on the Help server. This option is used together with the **UserDocumentation** document set and any document set that files have been published for on the Help server.
		2	World Wide Web	The documentation is stored on MSDN or a similar website. This option is required for the **DeveloperDocumentation** documentation set and should not be used with any other document set.

Menu properties

The following table describes the properties that are available for menus under the **Menus** node in Application Explorer.

PROPERTY	DESCRIPTION
ConfigurationKey	Set the configuration key for the menu.

PROPERTY	DESCRIPTION
CountryRegionCodes	Specify the codes for the countries/regions where the menu is applicable or valid. This property is implemented as a comma-separated list of ISO country codes in a single string. The values must match data in the global address book. The client uses this property to enable or disable country/region-specific features.
DisabledImage	Specify the button image that is used when the menu is disabled. If this property isn't set, the system uses the setting of the **NormalImage** property to generate an image.
DisabledImageLocation	Specify the location of the image that is used for a disabled control. You can use images from a file, the **Resources** node in Application Explorer, or an embedded resource. The value that you select for this property determines the values that are available for the **DisabledImage** property. If the property isn't set, the system uses the setting of the **ImageLocation** property to generate an image.
ImageLocation	Specify the location of the image that is used. You can use images from a file, the **Resources** node in Application Explorer, or an embedded resource. The value that you select for this property determines the values that are available for the **NormalImage** property.
Label	Set the name of the menu that is shown to the user.
MenuItemName	Specify the menu item to include on the menu. The values that are available depend on the value of the **MenuItemType** property.
MenuItemType	Specify the type of the menu item. There are three categories of menu items: • Display • Output • Action The value that you set for this property determines the list of menu item names that appears in the list for the **MenuItemName** property.
Model	Specify the model that the menu is in. A model is a logical grouping of elements in a layer. Examples of elements include a table or class. An element can be located in exactly one model in a layer. The same element can be located in a customized version in a model that is in a higher layer.
NormalImage	Specify the image that is used when the menu is enabled.
Parameters	Specify one or more values that are passed to an object. These values resemble the parameters that are passed to a method. A parameter supplies a value that is then used to perform the task. There is no default value.

PROPERTY	DESCRIPTION
SetCompany	If you set this property to **Yes**, every time that the menu is opened, the company is changed to the company that was specified when the menu was first started.
Shortcut	Specify the keyboard shortcut that opens the menu. For example, you can press Ctrl+F3 to open the menu. There is no default value.
ShowParentModule	Specify whether to update the navigation pane, based on the parent module of the menu item. The following options are available: • **Yes** – Always update the navigation pane, based on the parent module of the menu item. • **No** – Leave the navigation pane unchanged, even if the parent module of the menu item differs from the current module. The default value is **Yes**.

Menu item properties

The following properties are available for all menu items (display, output, and action), even menu items that are used for web menus.

PROPERTY	DESCRIPTION
ConfigurationKey	Select the configuration key that is required in order to enable the menu item. Use the key for the module that the object belongs to.
CopyCallerQuery	Specify whether to copy the query from the calling form to the target form. This property enables the target form to show the same data that was viewed in the original form. The default value is **Auto**.
CorrectPermissions	Specify whether correct permission should be available for selection when privileges are assigned to the menu item. The following options are available: • **Auto** – The permission will be available for selection as a privilege on this menu item's **Privileges** node under the **Entry Points** node. • **No** – The permission won't be available for selection as a privilege on the menu item. The default value is **Auto**.
CountryConfigurationKey	Optional: Set a country/region-specific key in addition to or instead of a standard configuration key.
CountryRegionCodes	Specify the codes for the countries/regions where the menu item is valid. This property is implemented as a comma-separated list of ISO country codes in a single string. The values must match data in the global address book. The client uses this property to enable or disable country/region-specific features.

PROPERTY	DESCRIPTION
CreatePermissions	Specify whether create permission should be available for selection when privileges are assigned to the menu item. The following options are available: • **Auto** – The permission will be available for selection as a privilege on this menu item's **Privileges** node under the **Entry Points** node. • **No** – The permission won't be available for selection as a privilege on the menu item. The default value is **Auto**.
DeletePermissions	Specify whether delete permission should be available for selection when privileges are assigned to the menu item. The following options are available: • **Auto** – The permission will be available for selection as a privilege on this menu item's **Privileges** node under the **Entry Points** node. • **No** – The permission won't be available for selection as a privilege on the menu item. The default value is **Auto**.
DisabledImage	Specify the image that is used when the menu item is disabled. If this property isn't set, the system uses the setting of the **NormalImage** property to generate an image.
DisabledImageLocation	Specify the location of the image that is used for a disabled control. You can use images from a file, the **Resources** node in Application Explorer, or an embedded resource. The value that you select for this property determines the values that are available for the **DisabledImage** property. If the property isn't set, the system uses the setting of the **ImageLocation** property to generate an image.
EnumTypeParameter and EnumParameter	Optional: Select an enumerated type as a parameter for the object, and then select an enumeration value as the value of the **EnumParameter** property. Typically, these properties are used when one form is used in several situations. You can change the behavior of the form, depending on the **EnumParameter** value. For example, the **PriceDiscGroup** form is used by three display menu items (**PriceDiscGroup_***), each of which has a different **EnumParameter** value. In the form's **init** method, a switch construct validates the value, and then the form is created.
ExtendedDataSecurity	Specify whether the menu item appears under all companies instead of in the context of a single company. The default value is **No**.
FormViewOption	Specify the form mode to use. The following options are available: • Auto • Grid • Details The default value is **Auto**.

PROPERTY	DESCRIPTION
HelpText	Create a Help string for the menu item. The text appears on the status bar when you select the object that is opened by the menu item (for example, a form). **Note:** To write a Help topic for the menu item, in Application Explorer, in the **Application Documentation/Menu Items** node, find the topic that has the same name as your menu item. This topic will then be shown instead of any Help topic that was written about the object that is opened by the menu item.
ImageLocation	Specify the location of the image that is used for a control. You can use images from a file, the **Resources** node in Application Explorer, or an embedded resource. The value that you select for this property determines the values that are available for the **NormalImage** property.
Label	Select the label to use as the name that appears for the item on menus and buttons.
LinkedPermissionObject	If the permissions of another object (for example, a form or report) should be applied to this menu item, select the object. Typically, this property is used for action menu items.
LinkedPermissionType	Specify the type of the object that is specified by the **LinkedPermissionObject** property.
MultiSelect	Select whether the menu item can be used on multiple record selections in forms.
Model	Specify the model that the table is in. A model is a logical grouping of elements in a layer. Examples of elements include a table or class. An element can be located in exactly one model in a layer. The same element can be located in a customized version in a model that is in a higher layer.
Name	The name of the menu item.
NeededAccessLevel	Define the minimum access that is required for the menu item to appear on a menu or a button. This property is used to set access to the menu item for different user groups.
NeedsRecord	Specify whether a button that represents the menu item is enabled if no record is present. The default value is **No**. You can use this property to help guarantee that an action can be completed. For example, you have a menu item button that opens a detail form. You might want to disable the button if there are no records on the list page.
NormalImage	Specify the image that is used when the menu item is associated with an enabled button control.
Object	Select an object of the object type that is specified in the **Class** property.

PROPERTY	DESCRIPTION
ObjectType	Select the type of object that the menu item opens. **Caution:** You should use **SSRSReport** for a menu item for an SSRS report. Don't use **SQLReportLibraryReport** for new menu items. The **SQLReportLibraryReport** option is obsolete and will be removed in a future version.
OpenMode	Specify the view mode of the target form. You use this property to specify whether the target form opens in edit mode or read-only mode. The following options are available: • Auto • View • Edit • New The default value is **Auto**.
Parameters	Optional: Specify the arguments that are passed to the object.
Query	Select the query that is passed to the target form for the **InitialQuery** method.
ReadPermissions	Specify whether read permission should be available for section when privileges are assigned to the menu item. The following options are available: • **Auto** – The permission will be available for selection as a privilege on this menu item's **Privileges** node under the **Entry Points** node. • **No** – The permission won't be available for selection as a privilege on the menu item. The default value is **Auto**.
ReportDesign	Select the report design to use for a specific SSRS report model.
RunOn	Select whether to run the menu item on the client, the server, or the location that it's called from. This property is mainly used for menu items that open reports. This property determines where the application object is run from only if the **RunOn** property of the object is set to **Called from**. • A form is instantiated and run on the client, because the **FormRun** class always runs on the client. • A report is instantiated and run as specified by the menu item's **RunOn** property, because the **ReportRun** class always runs where it was called from. You should set the property to **Called from**. If you set the report to run on the client, and the report is run in a batch, the report will fail. If you set the report to run on the server, and the report is shown on the screen, the report will fail. • A class's **main** method is run as specified by its modifier. The class itself is instantiated as specified by its **RunOn** property. The instantiation might occur in the **main** method.

PROPERTY	DESCRIPTION
UpdatePermissions	Specify whether update permission should be available for section when privileges are assigned to the menu item. The following options are available: • **Auto** – The permission will be available for section as a privilege on this menu item's **Privileges** node under the **Entry Points** node. • **No** – The permission won't be available for section as a privilege on the menu item. The default value is **Auto**.
Web	Specify the URL that is opened when you run the menu item. The value of this property is no longer used. Don't use this property.
WebConfigurationKey	Optional: Select a web-specific configuration key in addition to a standard configuration key. This property applies to web menu items only.
WebMenuItemName	Specify the menu item to include on a web menu. The available values depend on the setting of the **WebMenuItemType** property.
WebMenuItemType	Specify the type of the web menu item. There are two categories of web menu items: • URL • Action The value that you select determines the web menu item names that are available for the **WebMenuItemName** property.
WebPage	Specify the webpage that is linked to the menu item. The value of this property is no longer used. Don't use this property.
WebSecureTransaction	Select whether the menu item requires secure transactions (SSL). This property applies to web menu items only.

NOTE

When you use the **Parameters** or **EnumParameter** property, errors such as type mismatches can be found only at run time, not at compile time.

Query properties

Within a query, you can set properties on the query itself, the data sources, the fields that are used for sorting, and the ranges that are used to delimit the query.

Query properties

Query properties determine the overall behavior of the query. For example, you can specify the form that is shown to users so that they can interact with the query.

PROPERTY	DESCRIPTION
AllowCheck	This property is ignored for queries. It's effective on forms and reports.
AllowCrossCompany	Specify whether data is retrieved for all companies that the user has authority to read from. If the property is set to **false**, which is the default value, data is retrieved only for the current session company.
Description	Optional: Describe the query, what it returns, and so on. This property is useful in Microsoft Office Add-in scenarios.
Form	Specify that query form that MorphX should show when users interact with the query. The default value is **SysQueryForm**.
Interactive	Specify whether users can interact with the report by delimiting queries, setting printer options, and so on.
Literals	Specify how literals are represented in SQL statements. The **forceLiterals** option instructs the kernel to reveal the actual values that are used in **where** clauses to the Microsoft SQL Server database at the time of optimization. The **forcePlaceholders** option instructs the kernel not to reveal the actual values. **Note:** We don't recommend that you use the **forceLiterals** option, because it could expose code to an SQL injection security threat.
Model	Specify the model that the query is in. A model is a logical grouping of elements in a layer. Examples of elements include a table or class. An element can exist in exactly one model in a layer. The same element can exist in a customized version in a model that is in a higher layer.
QueryType	Specify the type of the query. The following options are available: • Join • Union The default value is **Join**.
Searchable	Specify whether the query can be part of a set of queries that is used to search the Microsoft SharePoint Business Catalog. This property is useful when you use the Enterprise Search feature. The default value is **No**.
Title	The heading for the query.
UserUpdate	Specify whether the query form should retain its state when it's reopened. If you set this property to **Yes**, the previous settings are restored. If you set it to **No**, the data can be viewed but not edited.
Version	The version is increased every time that the query is updated. This property is read-only.

Data source properties

The following properties control the characteristics of a data source. Additional properties are available on embedded data sources and relations between data sources. You can also set one property on fields in the data source.

PROPERTY	WHERE IT'S AVAILABLE	DESCRIPTION
AllowAdd	Data source	Specify whether users can add fields to sorting and ranges at run time.
Company	Data source	Specify the company to retrieve data from.
Dynamic	Fields node in a data source	Specify whether all fields in the table in the data source are used. If you set this property to **Yes**, all the fields in the data source are used. If you set it to **No**, you can remove some of the fields. When the data source is a base table, a value of **Yes** means that all fields from the derived tables are used.
Enabled	Data source	If you set this property to **No**, the data source (and all embedded data sources) are ignored.
FetchMode	Embedded data source	Specify whether the data sources should be related through a 1:1 relation or a 1:n relation. **Note:** For data sources that are used on reports, use a join relation that uses 1:1 fetch mode.
Field, RelatedField	Relations on an embedded data source	The name of the fields from the parent data source and related data source that are used in the relation.
FirstFast	Data source	If you set this property to **Yes**, the database receives a hint that the first record from the query should be retrieved before the other records. This setting enables some database systems to optimize record retrieval and therefore helps improve performance.
FirstOnly	Data source	If you set this property to **Yes**, the database receives a hint that only the first record from the query is required. This setting enables some database systems to optimize record retrieval and therefore helps improve performance.
JoinMode	Embedded data source	Specify the strategy that is used to join the output from a data source.
Name	Data source	Specify the name of the data source.

PROPERTY	WHERE IT'S AVAILABLE	DESCRIPTION
Relations	Embedded data source	Specify whether the query system should use the relations that are defined for tables and EDTs. If you set this property to **Yes**, the query is automatically updated if a relation is changed.
Table	Data source	Specify the table, map, or view that is used as a data source. This property can't be modified after a sorting order or a range has been defined.
Table, RelatedTable	Relations on an embedded data source	The name of the parent data source and the related data source.
UniqueId	Data source	The unique number of the data source. This property is read-only.
Update	Data source	Specify whether the query can update records in the database.

Range properties

The following properties determine the characteristics of the range specification. For example, you can specify whether users are allowed to modify the range at run time.

PROPERTY	DESCRIPTION
Enabled	Use this property to disable a field in a range specification.
Field	Specify the field to define a range on.
Label	Enter a label for the range.
Status	Specify whether users are allowed to modify the range in the query dialog box at run time. The following options are available: • **Open** – Users can view and edit the range. • **Lock** – Users can only view the range. • **Hide** – Users can't view or edit the range.
Value	Specify the range for the records that are retrieved. If you use enumerations, don't use text strings. The enumeration ID must be used.

Report properties

Most of the properties for a report are set on the design, design section, and control nodes in Application Explorer. For information about system properties that are available on reports, see the "System and common properties" section. The following table describes the properties of a report.

PROPERTY	DESCRIPTION
AllowCheck	Specify whether a message is shown when users try to run reports that they don't have permission to view. Select **Yes** to specify that a message is shown.
AutoJoin	Specify whether a record that is returned by the **element.args** method is used to set the range on the report query.
Interactive	Specify whether users can select which records to show by modifying the query that is associated with a report.
Model	Specify the model that the report is in. A model is a logical grouping of elements in a layer. An element can exist in exactly one model in a layer. The same element can exist in a customized version in a model that is in another layer.

Report control properties

The following table describes report control properties. For information about additional properties that are available for controls, see the "Form control properties" section.

PROPERTY	DESCRIPTION
Alignment	Specify the alignment of a value that is shown in a control.
AllowNegative	Specify whether the control accepts negative values. This property is available only for integer and real controls.
ArrayIndex	Specify the array element that is shown in a control. The control is based on an extended data type that has array elements. This property isn't available for text and shape controls.
AutoDeclaration	Specify whether a variable is created that has the same name as the control. When you set this property to **Yes**, it's easier to access the report controls from X++ code, and you can find errors at compile time.
AutoInsSeparator	Specify whether a decimal separator is shown. This property is available only for real controls.
BackgroundColor	Specify the background color for a control. The setting of this property can be overridden by using the **BackStyle** property.
BackStyle	Specify whether the control background is opaque or transparent. When you set this property to **Transparent**, the behavior depends on the type of control: • For bitmap controls, pixels that have the same color are transparent. • For all other controls, the foreground color is used as the background color.
Bold	Specify bold text formatting.

PROPERTY	DESCRIPTION
BottomMargin	Specify the margin for a control.
ChangeCase	Specify the case of text that a user enters. This property is available only for string, enum, text, and prompt controls.
ChangeLabelCase	Specify whether the label for the control should be modified when the report is printed. The following options are available: • Auto • None • UPPER CASE • lower case • Title Case The default value is **Auto**.
ColorScheme	Specify the color palette for a control.
ConfigurationKey	Specify a configuration key for the control.
CSSClass	Specify the Cascading Style Sheet (CSS) to use to render the value in HTML.
DataField	Specify a table field for the control. This property isn't available for text, shape, box, and bitmap controls.
DataMethod	Specify a display method that shows data in a control. This property isn't available for text, shape, and box controls.
DateDay	Specify the format for the day. This property is available only for date controls.
DateFormat	Specify the format for a date. This property is available only for date controls.
DateMonth	Specify the format for the month. This property is available only for date controls.
DateSeparator	Specify the separator that appears between the month, day, and year. This property is available only for date controls.
DateYear	Specify the format for the year. This property is available only for date controls.
DecimalSeparator	Specify the symbol that is used to separate decimal values. This property is available only for real controls.
DisplaceNegative	Specify a new position for a value in a grid control when the value is a negative number. This property is available only for integer and real controls.

PROPERTY	DESCRIPTION
DynamicHeight	Specify whether the control is resized to show additional lines of text. When you set this property to **Yes**, page headers, page footers, and repeating column headings are automatically added as required. This property is available only for string controls.
ExtendedDataType	Specify the EDT that the field that is associated with the control should be based on.
ExtraSumWidth	Modify the default width that is allowed for sums. This property is available only for integer and real controls.
Font	Specify the font.
FontSize	Specify the font size.
ForegroundColor	Specify the foreground color for a control.
FormatMST	Specify whether values are formatted by using standard currency format. This property is available only for real controls.
Height	Specify the height of a control. When a control is associated with an EDT, the **Height** property of the control overrides the **DisplayLength** property of the EDT. If you set the **Height** property to **Auto** for a bitmap control, the size of the control is based on the size of the graphic.
ImageName	Specify the file name for an image. This property is available only for bitmap controls.
ImageResource	Specify the ID for a system resource to show. The resource macro provides a list of these IDs. Macros are located under the **Macros** node in Application Explorer. This property is available only for bitmap controls.
Italic	Specify italic text formatting.
Label	Specify a title for the control. If a label isn't specified here, it's inherited from the field.
LabelBold	Set or return a value that indicates the bold setting for the label in the control.
LabelCSSClass	Specify the CSS to use to render the label in HTML.
LabelFont	Set or return a string data type value that indicates the font for the label text in a form combo box control.
LabelFontSize	Set or return the font size, in points, for the label text in a form combo box control.
LabelItalic	Set or return a value that indicates whether the text in the control label should be italic.

PROPERTY	DESCRIPTION
LabelLineBelow	Specify the format of the underline for the control title.
LabelLineThickness	Specify the format of the line below column headings.
LabelPosition	Set or return the position of the label for the control. Valid values are **Left** and **Above**.
LabelTabLeader	Specify whether to append a series of dots to control labels. The following options are available: • Auto • Do not append • Do append The default value is **Auto**.
LabelUnderline	Set or return a value that indicates whether the text in the control label should be underlined.
LabelWidth	Specify the width of the label for the control.
Left	Specify the left alignment of a control.
LeftMargin	Specify the left margin for a control.
Line	Specify the appearance of the lines that form a shape. This property is available only for shape controls.
LineAbove	Specify the type of line for the top border of a control. If your report has many lines or boxes, consider using a shape control inside the individual sections.
LineBelow	Specify the type of line for the bottom border of a control. If your report has many lines or boxes, consider using a shape control inside the individual sections.
LineLeft	Specify the type of line for the left border of a control. If your report has many lines or boxes, consider using a shape control inside the individual sections.
LineRight	Specify the type of line for the right border of a control. If your report has many lines or boxes, consider using a shape control inside the individual sections.
MenuItemLabel	Specify the label for a menu item.
MenuItemName	Specify the name of the menu item. The available menu items vary, depending on the setting of the **MenuItemType** property.
MenuItemType	Specify whether the menu item is an action, display, or output menu item. A display menu item is for a form, and an output menu item is for a report. An output menu item is for a class, job, or query.

PROPERTY	DESCRIPTION
MinNoOfDecimals	Specify the minimum number of decimal places that are shown. Trailing zeros aren't shown.
ModelFieldName	Specify a field that is used to determine the left alignment and width of a control.
NoOfDecimals	Specify the number of decimal places that are shown. The default value is **20**. This property is available only for real controls.
ResizeBitmap	Specify whether an image can be resized to fit the dimensions of a control. This property is available only for bitmap controls.
RightMargin	Specify the margin for a control.
RotateSign	Specify whether the sign for the control is inverted. This property is available only for integer and real controls.
ShowLabel	Set or return a value that indicates whether the label for the control is shown in the form. A value of **True** indicates that the label will be shown.
ShowPicAsText	Specify whether the file name for an image is shown instead of the image. This property is available only for bitmap controls.
ShowZero	Specify whether a 0 (zero) value is shown. This property is available only for integer and real controls.
SignDisplay	Specify how the sign of a number is shown. This property is available only for integer and real controls.
SumAll	Specify whether the sum of all values is calculated. This property is available only for integer and real controls.
SumNeg	Specify whether the sum of all negative values is calculated. This property is available only for integer and real controls.
SumPos	Specify whether the sum of all positive values is calculated. This property is available only for integer and real controls.
Table	Specify a data source for the control. This property isn't available for text, shape, box, and bitmap controls.
Text	Specify the text string that is shown in a control. This property is available only for text controls.
TimeFormat	Specify whether times are shown in 24-hour format or AM/PM format. This property is available only for time controls.
TimeHours	Specify whether hours are shown. This property is available only for time controls.

PROPERTY	DESCRIPTION
TimeMinutes	Specify whether minutes are shown. This property is available only for time controls.
TimeSeconds	Specify whether seconds are shown. This property is available only for time controls.
TimeSeparator	Specify the symbol that is used to separate hours, minutes, and seconds. This property is available only for time controls.
Thickness	Specify the thickness of a control border.
ThousandSeparator	Specify the symbol that is used to separate thousands. This property is available only for real controls.
Top	Specify the top alignment of a control.
TopMargin	Specify the margin for a control.
Type	Specify the type of shape that is shown. This property is available only for shape controls.
TypeHeaderPrompt	Specify whether a line of dots is added to fill the space between the control title and the control value. This property is available only for text and prompt controls.
Underline	Specify underline text formatting.
Visible	Set or return a value that indicates whether the control is visible. A value of **True** indicates that the control is visible.
WarnIfMissing	Specify whether a message is shown if an image is missing from the report. This property is available only for bitmap controls.
WebMenuItemName	Specify the menu item to include on a web menu. The available values depend on the setting of the **WebMenuItemType** property.
WebMenuItemType	Specify the type of the menu item. There are two categories of web menu items: • URL • Action The value that you select determines the web menu item names that are available for the **WebMenuItemName** property.
WebTarget	Specify the location of the control on a web report.
Width	Specify the width of a control. When a control is associated with an EDT, the **Width** property of the control overrides the **DisplayLength** property of the EDT. If you set the **Width** property to **Auto** for a bitmap control, the size of the control is based on the size of the graphic.

Report design properties

The following table describes the report design properties.

PROPERTY	DESCRIPTION
ArrangeMethod	Specify the layout for the controls in a report section.
ArrangeWhen	Specify when report controls are arranged.
BottomMargin	Specify the bottom margin. If you set this property to **Auto**, the default value that is stored in the system table is used.
Caption	Specify the name that is shown for the report in the user interface.
ColorScheme	Specify the color palette.
Columns	Specify the number of columns.
ColumnSpace	Specify the space between columns.
Font, FontSize, Italic, Underline and Bold	Specify the text formatting. The settings of the **Font** and **FontSize** properties override the values that you can set by clicking **Options** > **Fonts** on the **Tools** menu.
ForegroundColor	Specify the foreground color.
Height	Specify the height.
LeftMargin	Specify the left margin. If you set this property to **Auto**, the default value that is stored in the system table is used.
LineAbove	Specify the type of line for the top border of a section. If a report has many lines and boxes, consider using the shape control inside a section.
LineBelow	Specify the type of line for the bottom border of a section. If a report has many lines and boxes, consider using the shape control inside a section.
LineLeft	Specify the type of line for the left border of a section. If a report has many lines and boxes, consider using the shape control inside a section.
LineRight	Specify the type of line for the right border of a section. If a report has many lines and boxes, consider using the shape control inside a section.
ResolutionX, ResolutionY	Specify the distance between gridlines.
RightMargin	Specify the right margin. If you set this property to **Auto**, the default value that is stored in the system table is used.

PROPERTY	DESCRIPTION
Ruler	Specify the unit for the ruler that is shown when you edit a design. To edit a design, right-click **AutoDesignSpecs** or **Generated Design**, and then click **Edit**.
Thickness	Specify the thickness of a section border.
TopMargin	Specify the top margin. If you set this property to **Auto**, the default value that is stored in the system table is used.

Report design section properties

The following table describes properties for report design sections. For information about additional properties that are available for report designs, see the "Report design properties" section.

PROPERTY	DESCRIPTION
ArrangeMethod	Specify the layout for the controls in a report section.
ArrangeWhen	Specify when the controls in the container should be arranged. The available options are **Startup**, **On demand**, and **Never**.
Bold	Get or set the weight of the font that was used to show text in the control.
Bottom	Change the position of the bottom of the report.
BottomMargin	Specify the bottom margin. If you set this property to **Auto**, the default value that is stored in the UserInfo system table is used.
ColorScheme	Specify the color palette.
ColumnHeadingsStrategy	Specify the layout of column headings. If you set this property to **WordWrap**, a heading wraps when it's longer than the longest field in the column. Headings can wrap to a maximum of eight lines. Headings that are longer than eight lines are truncated. **Note:** The heading length varies, depending on the language.
Columns	Specify the number of columns.
Columnspace	Specify the space between columns.
Font	Specify the text formatting. The settings of the **Font** and **FontSize** properties override the values that you can set by clicking **Options** > **Fonts** on the **Tools** menu.
FontSize	Specify the text formatting. The settings of the **Font** and **FontSize** properties override the values that you can set by clicking **Options** > **Fonts** on the **Tools** menu.
ForegroundColor	Specify the foreground color.

PROPERTY	DESCRIPTION
GrandHeader	Specify whether the value of the **HeaderText** property is shown. The **GrandHeader** property is available only when a report has multiple data sources that aren't nested.
GrandTotal	Specify whether the value of the **FooterText** property is shown. The **GrandTotal** property is available only when a report has multiple data sources that aren't nested.
HeaderText	Specify the text that is shown above the first record in a section when the **GrandHeader** property is set to **Yes**. This property is available only when a report has multiple data sources that aren't nested.
Height	Specify the height.
Italic	Specify the text formatting. The settings of the **Font** and **FontSize** properties override the values that you can set by clicking **Options** > **Fonts** on the **Tools** menu.
LabelTopMargin, LabelBottomMargin	Specify the margins above and below column headings.
LeftMargin	Specify the left margin. If you set this property to **Auto**, the default value that is stored in the UserInfo system table is used.
LineAbove, LineBelow, LineLeft, LineRight	Specify the type of line for a section border. If a report has many lines and boxes, consider using the shape control inside a section.
Map	Specify the map to use to show data. You can associate a map field with a field in one or more tables. This property lets you use the same field name to access fields that have different names in different tables.
NoOfHeadingLines	Specify the number of lines that are used to show column headings. If you set the property to **0** (zero), column headings aren't displayed. For reports that include several fields, increase the number of lines to make sure that all fields are shown.
RightMargin	Specify the right margin. If you set this property to **Auto**, the default value that is stored in the UserInfo system table is used.
ResolutionX	Specify the distance between gridlines.
ResolutionY	Specify the distance between gridlines.
Ruler	Specify the unit for the ruler that is shown when you edit a design. To edit a design, right-click **AutoDesignSpecs** or **Generated** Design, and then click **Edit**.
Table	Specify the data source for a section.
Thickness	Specify the thickness of a section border.

PROPERTY	DESCRIPTION
Top	Change the position of the top of the report.
TopMargin	Specify the top margin. If you set this property to **Auto**, the default value that is stored in the UserInfo system table is used.
Underline	Specify the text formatting. The settings of the **Font** and **FontSize** properties override the values that you can set by clicking **Options** > **Fonts** on the **Tools** menu.

Report query properties

The following table describes report query properties. For information about additional report properties, see the "Report properties" and "System and common properties" sections.

PROPERTY	DESCRIPTION
AllowCheck	Get or set the Allow check flag.
AllowCrossCompany	Get or set the Allow cross-company flag. This flag indicates whether query execution will be across companies.
Description	A textual explanation of the query. This optional property is often used in Office Add-ins scenarios.
Form	Specify the form that is used for user interaction.
Interactive	Specify whether users can interact with the report by delimiting queries, setting printer options, and so on.
Literals	Specify how literals are represented in SQL statements.
Model	Specify the model that the report query is in. A model is a logical grouping of elements in a layer. Examples of elements include a table or class. An element can exist in exactly one model in a layer. The same element can exist in a customized version in a model that is in another layer.
QueryType	Specify the type of the query. The following options are available: • Join • Union The default value is **Join**.
Searchable	Specify whether the query can be part of a set of queries that can be used to search the SharePoint Business Catalog. This property is useful when you use the Enterprise Search feature. The default value is **No**.
Title	Specify the title of the query.

PROPERTY	DESCRIPTION
UserUpdate	Specify whether users can update a query.
Version	This is a read-only internal property.

Security code permission properties

A code permission is a group of permissions that are associated with a menu item or a service operation. When a security role has access to a menu item, it also has access to other Application Explorer items that are mentioned within the code permission for that menu item. The degree of access is controlled by the specific permissions that are defined under the code permission node.

Securable objects

Code permissions are used to give access to securable objects. The following list shows the hierarchy of code permission nodes in Application Explorer:

- Security
  - Code Permissions
    - YourCodePermission
      - Tables
      - Server Methods
      - Associated Objects
        - Forms
        - Web Controls
        - Reports

Code permissions can also override the access levels to securable objects under the **Associated Objects** node.

Code permission properties

This following table describes the properties for the node at **Security** > **Code Permissions** > **YourCodePermission** in Application Explorer.

PROPERTY	REQUIRED	DESCRIPTION
Name	Yes	The name of the code permission. The code permission lets users run the class method that is specified in the **Method** property.
Class	Optional	The class that is associated with this code permission.
Method	Optional	The method that is associated with this code permission.

Table properties

The following table describes the properties for the node at **Security** > **Code Permissions** > **YourCodePermission** > **Tables** > **YourTable** in Application Explorer.

PROPERTY	REQUIRED	DESCRIPTION
Table	Yes	The name of the table.
EffectiveAccess	Yes	The permission value. The following options are available: • Read • Update • Create • Correct • Delete • NoAccess The permission values for the **EffectiveAccess** property represent a hierarchy. Read is the weakest permission, and Delete is the strongest. Delete permission includes every other permission. Create permission includes Update and Read. You can set the permission value to **NoAccess** to prevent all access to the table.
ManagedBy	Optional	This property is used by automation tools.

Server method properties

The following tables describes the properties for the node at **Security** > **Code Permissions** > **YourCodePermission** > **Server Methods** > **YourServerMethod** in Application Explorer.

PROPERTY	REQUIRED	DESCRIPTION
Class	Yes	The name of the server class.
Method	Yes	The secure server method that is tagged with the **SysEntryPointAttribute** attribute.
EffectiveAccess	Yes	The permission value. The following options are available: • **Invoke** – The server method can be called. • **NoAccess** – The server method can't be called.
ManagedBy	Optional	This property is used by automation tools.

Form properties

The following table describes the properties for the node at **Security** > **Code Permissions** > **YourCodePermission** > **Associated Objects** > **Forms** > **YourForm** in Application Explorer.

PROPERTY	REQUIRED	DESCRIPTION
Form	Yes	The name of the form.

PROPERTY	REQUIRED	DESCRIPTION
AccessLevel	Yes	The permission value. The following options are available: • Read • Update • Create • Correct • Delete • NoAccess The permission values for the **EffectiveAccess** property represent a hierarchy. Read is the weakest permission, and Delete is the strongest. Delete permission includes every other permission. Create permission includes Update and Read. You can set the permission value to **NoAccess** to prevent all access to the form.
ManagedBy	Optional	This property is used by automation tools.

Web control properties

The following table describes the properties for the node at **Security** > **Code Permissions** > **YourCodePermission** > **Associated Objects** > **Web Controls** > **YourWebControl** in Application Explorer.

PROPERTY	REQUIRED	DESCRIPTION
WebControl	Yes	The name of the web control.
AccessLevel	Yes	The permission value. The following options are available: • Read • Update • Create • Correct • Delete • NoAccess The permission values for the **EffectiveAccess** property represent a hierarchy. Read is the weakest permission, and Delete is the strongest. Delete permission includes every other permission. Create permission includes Update and Read. You can set the permission value to **NoAccess** to prevent all access to the web control.
ManagedBy	Optional	This property is used by automation tools.

Report properties

The following table describes the properties for the node at **Security** > **Code Permissions** > **YourCodePermission** > **Associated Objects** > **Reports** > **YourReport** in Application Explorer.

PROPERTY	REQUIRED	DESCRIPTION
Name	Yes	The name of the report design.
Report	Yes	The full name of the report.
ManagedBy	Optional	This property is used by automation tools.

Security duty properties

Security permissions are combined into privileges, and privileges are combined into duties. Duties are defined as groups of related privileges that provide a user with access to a specific business function. In Application Explorer, these privileges are organized into the nodes of a duty.

Best practices

This section describes the best practice rules for duties.

- All duties should be assigned to a role.
- All duties should be part of a process cycle.
- Because a duty represents a specific business function, the name of the duty should rarely or never change. For example, your company pays bills. Although the details of how you pay bills might change, the essential function of paying bills won't change. Instead of creating a new duty, you should change the privilege subnodes of the duty.
- The name of a process cycle should rarely or never change.

Duty hierarchy in Application Explorer

The following list shows the hierarchy of duty nodes in Application Explorer:

- Security
  - Duties
    - YourDuty
      - Privileges

Duty properties

The following table describes the properties for the node at **Security** > **Duties** > **YourDuty** in Application Explorer.

PROPERTY	REQUIRED	DESCRIPTION
Name	Yes	The name of the duty.
Label	Yes	Text that is shown for the duty in the user interface.
Description	Yes	A description of the duty.
Enabled	Yes	A value that indicates whether the duty is enabled. The following options are available: • **Yes** – Enable the duty. • **No** – Disable the duty.

Privilege properties

The following table describes the properties for the node at **Security** > **Duties** > **YourDuty** > **Privileges** > **YourPrivilege** in Application Explorer.

PROPERTY	REQUIRED	DESCRIPTION
Name	Yes	The name of the privilege.
Enabled	Yes	A value that indicates whether the duty is enabled. The following options are available: • **Yes** – Enable the privilege. • **No** – Disable the privilege.

Security privilege properties

A privilege is a group of permissions. The nodes that are below each privilege node identify the securable objects that a user can access and set the level of access for each object.

Best practices

This section describes the best practice rules for privileges.

- You can use privileges to specify the access that is required in order to perform a job.
- You can use privileges to group the permissions for related securable objects. For example, menu items and their controls are closely related.
- You can assign privileges directly to security roles. However, security settings are easier to maintain if you assign duties or process cycles instead of privileges.

Securable objects

Privileges are used to give access to securable objects. The following list shows the hierarchy under the **Security** > **Privileges** node in Application Explorer:

- Security
  - Privileges
    - YourPrivilege
      - Entry Points
      - Permissions
        - Tables
        - Server Methods
        - Forms

Privileges can also override the access levels to securable objects as they are defined elsewhere in Application Explorer. For example, a privilege can override a permission that is defined by the **EffectiveAccess** property under **Forms** > **YourForm** > **Permissions** > **Update** > **Tables** > **YourTable** in Application Explorer.

Privilege properties

The following table describes the properties for the node at **Security** > **Privileges** > **YourPrivilege** in Application Explorer.

PROPERTY	REQUIRED	DESCRIPTION
Name	Yes	The name of the privilege.

PROPERTY	REQUIRED	DESCRIPTION
Label	Yes	Text that is shown for the privilege in the user interface.
Description	Yes	A description of the privilege.
Enabled	Yes	A value that indicates whether the duty is enabled. The following options are available: • **Yes** – Enable the privilege. • **No** – Disable the privilege.

Entry point properties

The following table describes the properties for the node at **Security** > **Privileges** > **YourPrivilege** > **Entry Points** > **YourEntryPoint** in Application Explorer.

PROPERTY	REQUIRED	DESCRIPTION
Name	Yes	The name of the entry point.
ObjectType	Yes	The object type of the entry point. The following options are available: • MenuItemDisplay • MenuItemOutput • MenuItemAction • ServiceOperation • WebActionItem • WebURLItem • WebManagedContent
ObjectName	Yes	The object name of the entry point.
ObjectChildName	Optional	A value that represents the service method name. **Note:** Specify a value for this property only if the **ObjectType** property is set to **ServiceOperation**.

PROPERTY	REQUIRED	DESCRIPTION
AccessLevel	Yes	The permission value. For all object types except **ServiceOperation**, the following options are available: • Read • Update • Create • Correct • Delete • NoAccess The permission values for the **AccessLevel** property represent a hierarchy. Read is the weakest permission, and Delete is the strongest. Delete permission includes every other permission. Create permission includes Update and Read. You can set the permission value to **NoAccess** to prevent all access to the entry point. The Correct permission applies only when a time state table is involved. This permission authorizes you to issue update records in a time state table. For the **ServiceOperation** object type, the following options are available: • **Invoke** – The server method can be called. • **NoAccess** – The server method can't be called.

Table properties

The following table describes the properties for the node at **Security** > **Privileges** > **YourPrivilege** > **Permissions** > **Tables** > **YourTable** in Application Explorer.

PROPERTY	REQUIRED	DESCRIPTION
Table	Yes	The name of the table.

PROPERTY	REQUIRED	DESCRIPTION
EffectiveAccess	Yes	The permission value. The following options are available: • Read • Update • Create • Correct • Delete • NoAccess The permission values for the **EffectiveAccess** property represent a hierarchy. Read is the weakest permission, and Delete is the strongest. Delete permission includes every other permission. Create permission includes Update and Read. The Correct permission applies only when a time state table is involved. This permission authorizes you to update records in a time state table. You can set the permission value to **NoAccess** to prevent all access to the table.
ManagedBy	Optional	This property is used by automation tools.

Server method properties

The following table describes the properties for the node at **Security** > **Privileges** > **YourPrivilege** > **Permissions** > **Server Methods** > **YourServerMethod** in Application Explorer.

PROPERTY	REQUIRED	DESCRIPTION
Class	Yes	The name of the server class.
Method	Yes	The name of the secure server method that is tagged with the **SysEntryPointAttribute** attribute.
EffectiveAccess	Yes	The permission value. The following options are available: • **Invoke** – The server method can be called. • **NoAccess** – The server method can't be called.
ManagedBy	Optional	This property is used by automation tools.

Form properties

The following table describes the properties for the node at **Security** > **Privileges** > **YourPrivilege** > **Permissions** > **Forms** > **YourForm** in Application Explorer.

PROPERTY	REQUIRED	DESCRIPTION
Form	Yes	The name of the form.

Security process cycle properties

A process cycle is a group of duties. A process cycle represents a high-level job function. Although the details of how a given job function is run might change over time, the concept and name of that job function probably don't change.

Best practices

This section describes the best practice rules for process cycles.

- Each duty should be part of a process cycle.
- Use a process cycle to organize a group of duties for a job function.

Process cycle hierarchy in Application Explorer

The following list shows the hierarchy of process cycles nodes in Application Explorer:

- Security
  - Process Cycles
    - YourProcessCycle
      - Duties

Process cycle properties

The following table describes the properties for the node at **Security** > **Process Cycles** > **YourProcessCycle** in Application Explorer.

PROPERTY	REQUIRED	DESCRIPTION
Name	Yes	The name of the process cycle.
Label	Yes	Text that is shown for the process cycle in the user interface.
Description	Yes	A description of the process cycle.
Enabled	Yes	A value that indicates whether the duty is enabled. The following options are available: • **Yes** – Enable the process cycle. • **No** – Disable the process cycle.

Duty properties

The following table describes the properties for the node at **Security** > **Process Cycles** > **YourProcessCycle** > **Duties** > **YourDuty** in Application Explorer.

PROPERTY	REQUIRED	DESCRIPTION
Name	Yes	The name of the duty.

PROPERTY	REQUIRED	DESCRIPTION
Enabled	Yes	A value that indicates whether the duty is enabled. The following options are available: • **Yes** – Enable the duty. • **No** – Disable the duty.

Security policy properties

Developers and system administrators can create security policies to deny access to a subset of data records in tables.

Constrained tables of a policy

You can add tables and views under the **Constrained Tables** node of a security policy in Application Explorer. These tables and views are related to the data source table of the query that is named in the **Query** property of the policy. The following list shows the hierarchy of security policy nodes in Application Explorer:

- Security
  - Policies
    - YourPolicy
      - Constrained Tables
        - YourConstrainedTable
          - YourConstrainedSubTable
        - YourConstrainedView

Each **Constrained Tables** node can contain any number of constrained tables and views. Additionally, each constrained table can contain any number of constrained sub-tables.

Security policy properties

The following table describes the properties for the node at **Security** > **Policies** > **YourPolicy** in Application Explorer.

PROPERTY	REQUIRED	DESCRIPTION
Name	Yes	The name of the security policy.
Label	Yes	The text that is shown for the security policy in the user interface.
PrimaryTable	Yes	The table that is specified in the data source for the security policy query.
Query	Yes	The query that the policy uses to filter data from the constrained tables that are specified in the policy.
UseNotExistJoin	Yes	A value that indicates whether the security query must be applied as a **not exists** join or an **exists** join.

PROPERTY	REQUIRED	DESCRIPTION
PolicyGroup	No	Administrators and developers can use this property to quickly identify groups of related security policies. The available options are the names of the security policy groups that the system administrator or developer has created. The system doesn't use this property at run time.
ConstrainedTable	Yes	A value that controls whether the security policy restricts the data values in records that are returned from the primary table. The following options are available: • **Yes** – The security policy is enforced on the primary table. • **No** – The security policy isn't enforced on the primary table.
Enabled	Yes	A value that controls whether the system enforces the policy at run time. The following options are available: • **Yes** – Enable the security policy. • **No** – Disable the security policy.
Operation	Yes	A value that controls which data operations the policy is enforced for. The following options are available: • Select • Insert • Update • Delete • Insert, Update and Delete • All operations
ContextType	Yes	A value that controls the context type of the security policy. The following options are available: • **ContextString** – You must specify a value for the **ContextString** property. The security policy uses a specific application context for the policy. • **RoleName** – The security policy is applied only to the application user who is assigned to the value of **RoleName**. • **RoleProperty** – This value is used in combination with the **ContextString** property to specify multiple roles context.

PROPERTY	REQUIRED	DESCRIPTION
ContextString	Yes	This property is used in combination with **ContextType** property. It can be used to specify an application or multiple roles context.

Security role properties

Roles represent a collection of permissions that can be granted to users. The nodes that are nested below each role node identify various securable objects that a user can access and specify the level of access.

Role node in Application Explorer

Roles are used to give access to securable objects. The following list shows the hierarchy of role nodes in Application Explorer:

- Security
  - Roles
    - YourRole
      - Duties
      - Privileges
      - Permissions
        - Tables
        - Forms
        - Server Methods
      - Sub Roles

Roles are typically associated with security duties, and sometimes with security privileges. Access levels to securable objects within a role are derived from the duties, privileges, or both. Roles can also override the access levels to securable objects under the **Permissions** node.

Role properties

The following table describes the properties for the node at **Security** > **Roles** > **YourRole** in Application Explorer.

PROPERTY	REQUIRED	DESCRIPTION
Name	Yes	The name of the role.
Label	Yes	Text that is shown for the role in the user interface.
Description	Yes	A description of the role.
Enabled	Yes	A value that indicates whether the duty is enabled. The following options are available: • **Yes** – Enable the role. • **No** – Disable the role.

PROPERTY	REQUIRED	DESCRIPTION
PastDataAccess	Yes	The past data access for the tables that have date-effective fields. The following options are available: • Read • Update • Create • Correct • Delete • NoAccess The permission values for the **PastDataAccess** property represent a hierarchy. Read is the weakest permission, and Delete is the strongest. Delete permission includes every other permission. Create permission includes Update and Read. You can set the permission value to **NoAccess** to prevent all access to the table.
CurrentDataAccess	Yes	The current data access for the tables that have date-effective fields.
FutureDataAccess	Yes	The future data access for the tables that have date-effective fields.
ContextString	Optional	A user-defined string that can be used by security policies.

Duty properties

The following table describes the properties for the node at **Security** > **Roles** > **Duties** > **YourDuty** in Application Explorer.

PROPERTY	REQUIRED	DESCRIPTION
Name	Yes	The name of the duty.
Enabled	Yes	A value that indicates whether the duty is enabled. The following options are available: • **Yes** – Enable the duty. • **No** – Disable the duty.

Privilege properties

The following table describes the properties for the node at **Security** > **Roles** > **Privileges** > **YourPrivilege** in Application Explorer.

PROPERTY	REQUIRED	DESCRIPTION
Name	Yes	The name of the privilege.

PROPERTY	REQUIRED	DESCRIPTION
Enabled	Yes	A value that indicates whether the duty is enabled. The following options are available: • **Yes** – Enable the privilege. • **No** – Disable the privilege.

Table properties

The following table describes the properties for the node at **Security** > **Roles** > **Permissions** > **Tables** > **YourTable** in Application Explorer.

PROPERTY	REQUIRED	DESCRIPTION
Table	Yes	The name of the table.
EffectiveAccess	Yes	The permission value. The following options are available: • Read • Update • Create • Correct • Delete • NoAccess The permission values for the **EffectiveAccess** property represent a hierarchy. Read is the weakest permission, and Delete is the strongest. Delete permission includes every other permission. Create permission includes Update and Read. You can set the permission value to **NoAccess** to prevent all access to the table.
ManagedBy	Optional	This property is used by automation tools.

Form properties

The following table describes the properties for the node at **Security** > **Roles** > **Permissions** > **Form** > **YourForm** in Application Explorer.

PROPERTY	REQUIRED	DESCRIPTION
Form	Yes	The name of the form.

Server method properties

The following table describes the properties for the node at **Security** > **Roles** > **Permissions** > **Server Methods** > **YourServerMethod** in Application Explorer.

PROPERTY	REQUIRED	DESCRIPTION
Class	Yes	The name of the server class.

PROPERTY	REQUIRED	DESCRIPTION
Method	Yes	The name of the secure server method that is tagged with the **SysEntryPointAttribute** attribute.
EffectiveAccess	Yes	The permission value. The following options are available: • **Invoke** – The server method can be called. • **NoAccess** – The server method can't be called.
ManagedBy	Optional	This property is used by automation tools.

Subrole properties

The following table describes the properties for the node at **Security** > **Roles** > **Sub Roles** > **YourSubRole** in Application Explorer.

PROPERTY	REQUIRED	DESCRIPTION
Name	Yes	The name of the subrole.
Enabled	Yes	A value that indicates whether the duty is enabled. The following options are available: • **Yes** – Enable the subrole • **No** – Disable the subrole.

Web menu properties

The following table describes the properties that are specific to web menus and submenus.

PROPERTY	DESCRIPTION
ConfigurationKey	Specify the configuration key that is used to control display of this menu. If a user doesn't have access to the configuration key, the menu won't be visible.
HighlightSelected	This property isn't supported.
Label	Specify the text that is shown for the top-level node of the web menu or submenu. The value can't exceed 250 characters.
MenuItemName	Specify the menu item to access when the top-level node for the menu or submenu is clicked. The options that are available depend on the setting of the **MenuItemType** property.
MenuItemType	Specify the type of menu item that is accessed by the top-level node of the menu or submenu. The available options are **Action** and **URL**.

PROPERTY	DESCRIPTION
Model	Specify the model. A model is a logical grouping of elements in a layer. Examples of elements include a table or class. An element can exist in exactly one model in a layer. The same element can exist in a customized version in a model that is in a higher layer.
SetCompany	This property causes the system to change the company when the form receives focus. If the **SaveDataPerCompany** property on a table is set to **Yes**, the **SetCompany** property on a form design that uses the table as a data source must also be set to **Yes**.
ShowParentModule	Specify whether to update the QuickLaunch, based on the parent module of the menu item. The following options are available: • **Yes** – Always update the QuickLaunch, based on the parent module of the menu item. • **No** – Leave the QuickLaunch unchanged, even if the parent module of the menu item differs from the current module. The default value is **Yes**.

Web menu item properties

The following table describes the properties that are specific to web menu items.

PROPERTY	DESCRIPTION
Big	Specify the size of the button when it's used on an Action Pane. The following options are available: • **Yes** – The button is shown at full size and is located at the start of the group. • **No** – The button is shown at the smaller size and is located on the right side of the group.
CloseDialogBehavior	Specify the action that is performed on the parent window when the dialog box closes. The following options are available: • **Auto** – Depending on how the dialog box was used, the appropriate update actions are performed when the dialog box is closed. • **RefreshDataSource** – The read-only data source on the parent form is updated. This option preserves the current selection and performs a **Research()** operation on the data source. • **RefreshPage** – Refresh the page. • **Submit** – Refresh the parent page. • **None** – No action is performed. The default value is **Auto**.
HideActionPane	Specify whether the Action Pane is visible on the page that is being opened.

PROPERTY	DESCRIPTION
HomePage	Specify whether the page is a Role Center page and is deployed to the main Enterprise Portal site.
NeedsRecord	When you set this property to **Yes**, the menu item is shown when there are no records in the data set.
PageDefinition	The page that the web menu item points to.
Parameters	Specify the arguments that are passed to the page that is being opened. Each parameter must have the following form: *name*= *value* If multiple parameters must be passed, they must be separated by an ampersand (&), as shown in the following example: mode=2&category=1
URL	Specify the URL to navigate to.
WebConfigurationKey	Select the configuration key that is required in order to enable the web menu item. Use the key for the module that the object belongs to.
WindowMode	Specify the type of window to use for the page that is being opened. The following options are available: • **Inline** – The page that is being opened will replace the existing content in the browser. If the web menu item is being accessed from a dialog box, the page that is being opened will open in a new browser window. • **Modal** – If no dialog box is open, a new dialog box will be created. If the web menu item is being accessed from a dialog box, the page that is being opened will replace the content of the current dialog box. • **NewModal** – The page that is being opened will always open in a new dialog box. • **NewWindow** – The page that is being opened will open in a new browser window.
WindowParameters	Specify additional parameters to control the appearance of the SharePoint dialog box. The parameters must be enclosed in braces ({}) and separated by commas. The following example shows how to set the **WindowParameters** property so that the dialog box has a size of 400 × 300 pixels, and so that it has no **Close** or **Maximize** button: {width:400, height:300, showClose:false, allowMaximize:false}
WindowSize	Specify the size of the window to use for the page that is being opened. The following options are available: • **Smallest** – 330 × 200 pixels • **Small** – 550 × 450 pixels • **Medium** – 800 × 630 pixels • **Large** – 930 × 630 pixels • **Maximum** – The largest size that will fit in the boundaries of the main browser window

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

X++ language reference

2/18/2021 • 2 minutes to read • Edit Online

X++ is an object-oriented, application-aware, and data-aware programming language used in enterprise resource planning (ERP) programming and in database applications. It provides system classes for a broad range of system programming areas, highlighted in the following table.

X++ LANGUAGE FEATURE	DESCRIPTION
Classes	In addition to system classes, there are also application classes for managing many types of business processes. Reflection on classes is supported.
Tables	X++ programmers can access the relational tables. X++ includes keywords that match most of the keywords in standard SQL. Reflection on tables is supported.
User interface	Manipulation of user interface items, such as forms and reports.
Best practice checks	X++ code is checked for syntax errors during compile time. The compile process also performs best practice checks. Violations of best practices can generate compiler messages.
Garbage collection	The X++ runtime execution engines have automatic mechanisms to discard objects that are no longer referenced, so that memory space can be reused.
Interoperability	Interoperability between classes written in X++ and in C# (or other .NET Framework languages) is supported.
File manipulation	File input and output is supported, including XML building and parsing.
Collections	Dynamic arrays are supported and the X++ includes several collection objects.

The X++ language programming guide is divided into these sections:

- Variables and data types
- Statements, loops, and exception handling
- Operators
- Classes and methods
- Data selection and manipulation
- Macros
- Attribute classes

Additional resources

- X++ Syntax
- X++ and C# Comparison

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

X++ variables

2/18/2021 • 9 minutes to read • Edit Online

This topic describes variables in X++.

- A *variable* is an identifier that points to a memory location where information of a specific data type is stored. The size, precision, default value, implicit and explicit conversion functions, and range depend on the variable's data type.
- The *scope* of a variable defines the area in the code where an item can be accessed.
- *Instance variables* are declared in class declarations, and can be accessed from any methods in the class or from methods that extend the class.
- *Local variables* can be accessed only in the block where they were defined.
- When a variable is declared, memory is allocated, and the variable is initialized to the default value.
- You can assign values to both static fields and instance fields as part of the declaration statement.
- Variables can be declared anywhere in a code block in a method. They don't have to be declared at the beginning of a method.
- *Constants* are variables where the value can't be changed when the variable is declared. They use the **const** or **readonly** keyword.
- Constants differ from *read-only fields* in only one way. Read-only fields can be assigned a value only one time, and that value never changes. The field can be assigned its value either inline, at the place where the field is declared, or in the constructor.

When you declare variables of managed types that aren't authored in X++, you have two options. You can fully qualify the type names in the declaration by including the full namespace, or you can add a **using** statement to your file and then omit the namespace from the type name.

Variable examples

```
// An example of two valid variable names.
str variableName;
CustInfo custNumber;

// An example of simultaneously declaring and initializing a variable.
real pi = 3.14159265359; // Assigns value of pi to 12 significant digits.

// An example of initializing an object by using the new method on the class.
Access accessObject = new Access(); // Simple call to the new method in the Access class.

// An example of multiple declarations using integers.
int i,j; // Declares 2 integers, i and j.

// An example of multiple declarations using an array.
int a[100,5], b=1; // Declares array with 100 integers with 5 in memory and b as an integer with value 1.

// An example of how variable scopes work.
class ScopeExample
{
    // The variable a is declared within the class.
    int a;

    // Because the method below is declared within the class,
    // it can access all the variables defined within the class.
    void aNewMethod()
    {
```

```
        // The variable b is declared within the method.
        // It can only be accessed by this method.
        int b;
    }
}

// An example of an assignment of field members inline.
public class FieldAssignmentExample
{
    int field1 = 1;
    str field2 = "Banana";
    void new()
    {
        // ...
    }
}

class ConstantExample
{
    // An example of a constant being declared at the class level.
    public const str MyContent = 'SomeValue';
    public int ResultSoFar()
    {
        return 1;
    }
}

// The constants can then be referenced by using the double-colon syntax.
str value = ConstantExample::MyContent;
// If you're in the scope of the class where the const is defined,
// you can omit the type name prefix (ConstantExample in this example).

// An example of the using clause where the alias can denote
// namespaces and classes.
using System;
using IONS=System.IO; // Namespace alias.
using Alist=System.Collections.ArrayList; // Class alias.
public class NamespaceExample
{
    public static void Main(Args a)
    {
        Int32 I; // Alternative to System.Int32.
        Alist al; // Using a class alias.
        al = new Alist();
        str s;
        al.Add(1);
        IONS.Path::ChangeExtension(@"c:\tmp\test.xml", ".txt");
    }
}
```

var

You can declare a variable without explicitly providing the type of the variable, if the compiler can determine the type from the initialization expression. The variable is still strongly-typed into one, unambiguous type.

You can use **var** only on declarations where initialization expressions are provided. (The compiler will infer the type from the initialization expression.) In some cases, this approach can make code easier to read.

You should use **var** to declare local variables in these situations:

- When the type of the variable is obvious from the right side of the assignment
- When the exact type isn't important
- For the declarations of **for** loop counters
- For disposable objects inside **using** statements

Don't use **var** when the type isn't clear from the initialization expression.

var examples

```
// When the type of a variable is clear from
// the context, use var in the declaration.
var var1 = "This is clearly a string.";
var var2 = 27; // This is an integer (not a real).
var i = System.Convert::ToInt32(3.4);

// Don't use var when the type of the variable is not clear
// from the context. Use an explicit type instead.
int var4 = myObject.ResultSoFar();
```

Declare anywhere

Declarations can now be provided wherever statements can be provided. A declaration is syntactically a statement, a *declaration statement*.

You can provide declarations immediately before the variable is used, and you don't have to declare all the variables in one place. Therefore, you have precise control over the scope of your variables.

You can give variables smaller scopes, outside which the variables can't be referenced. The lifetime of the variable is the scope that it's declared in. Scopes can be started at the block level (inside compound statements), in **for** statements, and in **using** statements.

There are several advantages to making scopes small:

- Readability is enhanced.
- You reduce the risk that a variable will be reused in an inappropriate manner during long-term maintenance of the code.
- It's easier to refactor code. You can copy in code without having to worry that variables might be reused in contexts where they shouldn't be reused.

In the following example, we declare the loop counter inside the **for** statement that it's used in.

```
void MyMethod()
{
    for (int i = 0; i < 10; i++)
    {
        info(strfmt("i is %1", i));
    }
}
```

The scope of the variable is the **for** statement itself, and includes the condition expression and the loop update parts. The value can't be used outside this scope.

In the following example, when the compiler reaches the **info** statement, it will issue the following error message: "'i' isn't declared."

```
void MyMethod()
{
    for (int i = 0; i < 10; i++)
    {
        if (i == 7)
        {
            break;
        }
    }
    // The next statement causes a compiler error.
    info(strfmt("Found: %1", i));
}
```

You can also scope variables to a **using** statement, as shown in the following example.

```
static void AnotherMethod()
{
    str textFromFile;
    using (System.IO.StreamReader sr = new System.IO.StreamReader("c:\\test.txt"))
    {
        textFromFile = sr.ReadToEnd();
    }
}
```

When you use an object that implements **IDisposable**, you should declare and instantiate that object in a **using** statement. The **using** statement calls the **Dispose** method on the object correctly, even if an exception occurs while you're calling methods on the object. You can achieve the same result by putting the object inside a **try** block and then explicitly calling **Dispose** in a **finally** block. In fact, the compiler translates the **using** statement in just this manner.

The following example shows some of the features that we have been describing.

```
// loop variable declared within the loop: It will
// not be misused outside the loop
for(int i = 1; i < 10; i++)
{
    // Because this value is not used from outside the loop,
    // its declaration belongs in this smaller scope.
    str s = int2str(i);
    info(s);
}
```

To prevent confusion, the compiler issues an error message if you try to introduce a variable that will hide another variable in an enclosing scope, or even in the same scope. For example, the following code will cause the compiler to issue the following diagnostic message: "A local variable named 'i' can't be declared in this scope because it would give a different meaning to 'i', which is already used in a parent or current scope to denote something else."

```
{
    int i;
    {
        int i;
    }
}
```

Constants, read-only variables, and macros

The concept of macros is fully supported. Constants have the following advantages over macros:

- You can add a documentation comment to a constant but not to the value of a macro. Eventually, the language service will pick up this comment and provide useful information to the user.
- A constant is known by IntelliSense.
- A constant is cross-referenced. Therefore, you can find all references for a specific constant but not for a macro.
- A constant is subject to access modifiers. You can use the **private**, **protected**, and **public** modifiers. The accessibility of macros isn't rigorously defined.
- Constant variables have scope, whereas macros don't have scope.
- You can see the value of a constant or a read-only variable in the debugger.

Macros that are defined in class scopes (that is, in class declarations) are effectively available in all methods of all derived classes. This feature was originally a bug in the implementation of the legacy compiler macro, however, many application programmers often take advantage of it now. The X++ compiler still honors this feature, but no new code that uses it should be written. This feature also has a significant effect on the performance of the compiler.

Constants can be declared at the class level, as shown in the following example.

```
private const str MyConstant = 'SomeValue';
```

The constants can then be referenced by using the double colon (::) syntax.

```
str value = MyClass::MyConstant;
```

If you're in the scope of the class where the constant (**const**) is defined, you can omit the type name prefix (**MyClass** in the preceding example). Therefore, you can easily implement the concept of a macro library. The list of macro symbols becomes a class that has public **const** definitions.

You can also define constants as variables only. The compiler will maintain the invariant so that the value can't be modified.

```
{
    const int Blue = 0x0000FF;
    const int Green = 0x00FF00;
    const int Red = 0xFF0000;
}
```

Null values for data types

The concept of **null** values that is available in many other database management systems (DBMSs) is not supported. A variable in X++ always has a type and a value, however, for each data type, one value is considered **null** (for example, when the **validateField** table method is run).

TYPE	VALUE THAT IS TREATED AS NULL
Date	1900-01-01
Enum	An element that has its value set to **0**
Integer	0

TYPE	VALUE THAT IS TREATED AS NULL
Real	0.0
String	An empty string
Time	00:00:00
Utcdatetime	Any value that has its date portion set to **1900-01-01**, regardless of the value of the time portion. For example, the value **1900-01-01T22:33:44** is treated as **null**. Note that any **utcDateTime** value that has its date portion set to **1900-01-01** is shown as blank by the X++ **print** statement. Only the value **1900-01-01T00:00:00** is shown as blank by the **Global::info** method. That value is the value from the **DateTimeUtil::MinValue** method.

When the **validateField** method checks whether a user has entered a value in a mandatory field, **0** isn't accepted in an **integer** type field, the first entry isn't accepted in an **enum** type field, and so on. Additionally, in SQL X++ statements, the values that are listed in the previous table yield **false** in a Boolean comparison. However, in non-SQL X++ statements, the equal and relational operators work with these values, just as they work with other values. Variables of the **container** type, and classes and variables of the **table** type can be **null** in the traditional DBMS sense. A **table** type is **null** if all its fields have their **null** value.

X++ Primitive data types

2/18/2021 • 17 minutes to read • Edit Online

This topic describes primitive data types in X++. The primitive data types in X++ are **anytype**, **boolean**, **date**, **enum**, **guid**, **int**, **int64**, **real**, **str**, **timeOfDay**, and **utcdatetime**.

anytype

The **anytype** data type is a placeholder for any data type. You should use variables of this type only as arguments and return values.

To use **anytype** as a variable, you must first assign a value to it, otherwise, a run-time error occurs. After you've assigned a value to **anytype**, you can't convert it to another data type.

Although you can use **anytype** variables in expressions, they're usually used as arguments and return types. The size, precision, scope, default value, and range of **anytype** depend on the conversion type that you assign to it. You can use **anytype** just as you use the data type that you convert it to. For example, if you assign an integer, you can then apply relational and arithmetic operators to the variable.

An **anytype** variable is automatically converted to a date, enumeration (enum), integer, real, string, extended data type (EDT) (record), class, or container when a value is assigned to the type. Additionally, the following explicit conversion functions can be used: **any2date**, **any2enum**, **any2int**, **any2real**, and **any2str**. You can't change the variable to another data type after you've converted it to **anytype**.

anytype examples

```
// An example of using anytype variables.
public static str range(anytype _from, anytype _to)
{
    return queryValue(_from) + '..' + queryValue(_to);
}

// Another example of using anytype variables.
void put(int position, anytype data)
{
    record = conPoke (record, position, data);
}

public void AnytypeMethod()
{
    // An example of automatic conversion for anytype.
    anytype a;
    a = "text"; // Automatically assigns a string literal.
}
```

boolean

The **boolean** data type contains a value that is evaluated as either **true** or **false**. You can use the reserved literal keywords **true** and **false** wherever a **boolean** expression is expected. The size of a **boolean** is 1 byte. The default value is **false**, and the internal representation is a short number.

A **boolean** is automatically converted to an **int**, **date**, or **real**. It has no explicit conversion functions. The internal representation of a **boolean** is an integer. You can assign any integer value to a variable that is declared as the **boolean** type. The integer value **0** (zero) is evaluated as **false**, and all other integer values are evaluated

as **true**. Because the internal representation of a **boolean** is an integer, **boolean** values are automatically converted to integers and reals.

boolean examples

```
public void BooleanMethod()
{
    // Simple declaration of a boolean variable, b.
    boolean b;

    // Multiple declarations of booleans.
    boolean b1, b2;

    // Boolean variable is initialized to true.
    boolean b3 = true;

    // Declares a dynamic array of booleans.
    boolean b4[];

    // This example shows the most common usage of a boolean: a boolean in
    // a conditional statement and as a result of a logical expression.
    void main()
    {
        // Declares a boolean called exprValue.
        boolean exprValue;

        // Assigns ExprValue the value of (7*6 == 42), which equates to true.
        exprValue = (7*6 == 42);

        // If the conditional statement is true, print "OK".
        if (exprValue)
        {
            print "OK";  //"OK" is printed because the expression is true.
        }
    }
}
```

date

The **date** data type contains the day, month, and year. Dates can be written as literals by using the following syntax: **Date literal = day \ month \ year**. You must use four digits for the year.

The **date** data type can hold dates between January 1, 1900, and December 31, 2154. The size of a **date** is 32-bits. The default value is **null**, and the internal representation is a date.

A **date** has no implicit conversions, however, the following explicit conversion functions can be used: **str2date**, **date2str**, **date2num**, and **int2date**.

You can add and subtract integers from dates, which moves the date some days into the future and past respectively. Subtracting dates from each other will calculate the difference in days, however, adding two dates together is not possible and will lead to a compiler error.

date examples

```
public void DateMethod()
{
    // Simple declaration of a date variable, d.
    date d;

    // Multiple declaration of two date variables.
    date d1, d2;

    // A date variable, d3, is initialized to the 21st of November 1998.
    date d3 = 21\11\1998;

    // Declaration of a dynamic array of dates.
    date d4[];

    // Using arithmetic operators with integer variables and dates.
    void myMethod()
    {
        int anInteger;
        date aDate;
        // Sets the date variable aDate to January 1, 1998.
        aDate = 1\1\1998;
        // Sets the integer variable anInteger to 30.
        anInteger = 30;
        // Uses an integer value in the computation of dates.
        // This sets aDate to aDate + 30; that is the 31st of January 1998.
        aDate = aDate + anInteger;

        // Create 2 variables, set bDate, and then subtract from that date.
        date bDate;
        int dateDifference;
        bDate = 2\10\1998;
        dateDifference = bDate - aDate; // dateDifference will equal 244.
    }
}
```

enum

An **enum** is a list of literals. Before you can use an **enum**, you must declare it in Application Explorer.

The literal values are represented internally as integers. The first literal has the number 0, the next literal has the number 1, the next literal has the number 2, and so on. You can use **enum** values as integers in expressions. The default value for the first entry is **0**, and the internal representation is a short number.

An **enum** value is automatically converted to a **boolean**, **int**, or **real**. Additionally, the following explicit conversion functions can be used: **enum2str** and **str2enum**.

Hundreds of enumerable types are built into the standard application. For example, the **NoYes** enum has two associated literals: **No** has the value **0**, and **Yes** has the value **1**. You can create as many enum types as you want, and you can declare up to 251 (0 to 250) literals in a single enum type. To reference an **enum** value, enter the name of the enum, two colons, and then the name of the literal. For example, to use the literal **No** in the **NoYes** enum, enter **NoYes::No**.

Create an enum

1. In Solution Explorer, right-click the project, point to **Add**, and then click **New Item**.
2. Under **Artifacts**, select **Data Types**.
3. Click **Base Enum** to select the new item type.
4. In the **Name** field, enter a name for the enum, and then click **Add**. A new enum is added to the project, and the enum designer for the new element is opened.
5. In the enum designer, right-click the enum name, and then click **New Element**.
6. In the **Properties** window, enter the name of the enum element.

enum examples

```
public void EnumMethod()
{
    // Declare the enum (a NoYes enum) in the Application Explorer.
    NoYes done;

    // An array of Criteria enums.
    Criteria crit[100];
}
```

guid

The **guid** data type holds a *globally unique identifier* (GUID) value. A GUID is an integer that can be used across all computers and networks, wherever a unique identifier is required. It's unlikely that the number will be duplicated. A valid GUID meets all the following specifications:

- It must have 32 hexadecimal digits.
- It must have four dash characters that are embedded at the following locations: 8-4-4-4-12.
- Braces ({}) at the beginning and end of a string are optional. For example, both "12345678-BBBb-cCCCC-0000-123456789012" and "{12345678-BBBb-cCCCC-0000-123456789012}" are valid GUID strings.
- It must have a total of either 36 or 38 characters, depending on whether braces are added.
- The hexadecimal digits a–f (or A–F) can be uppercase, lowercase, or mixed.

The size of a **guid** is 16 bytes or 128-bits. The following explicit conversion functions can be used: **any2guid**, **guid2str**, **newGuid**, **str2guid**, **Global::guidFromString**, and **Global::stringFromGuid**.

guid examples

The following set of examples shows how to use the **guid** functions. The code output of these examples follows.

```
// An example of how to use the GUID functions.
static void GuidRoundTripJob(Args _args)
{
    guid guid2;
    str string3;

    // Convert a guid to a string, and back to a guid.
    guid2 = newGuid();
    info(strFmt("Info_a1:  guid2 == %1", guid2));
    string3 = guid2str(guid2);
    info(strFmt("Info_a2:  string3 == %1", string3));
    guid2 = str2guid(string3);
    info(strFmt("Info_a3:  guid2 == %1", guid2));

    // Test string representations of a guid. Mixing upper and lower case letters does not affect the guid.
    guid2 = str2guid("BB345678-abcd-ABCD-0000-bbbbffff9012");
    string3 = guid2str(guid2);
    info(strFmt("Info_b1:  8-4-4-4-12 format for dashes works (%1)", string3));
    info(strFmt("Info_b2:  Mixed upper and lower case works."));

    // Test invalid dash locations, see output is all zeros. Dash locations must be exact.
    guid2 = str2guid("CC2345678abcd-ABCD-0000-cccc9012");
    string3 = guid2str(guid2);
    info(strFmt("Info_c1:  These embedded dash locations are required.  %1", string3));

    // Braces {} are optional.
    guid2 = str2guid("{DD345678-abcd-ABCD-0000-ddddaaaa9012}");
    string3 = guid2str(guid2);
    info(strFmt("Info_d1:  Braces {} are optional (%1)", string3));
}
```

guid code output

The following output appears in the Infolog. Note that the string includes the optional braces.

```
Message (02:26:46 pm)
Info_a1:  guid2 == {93945629-734B-475E-99CE-6AA7AFA43259}
Info_a2:  string3 == {93945629-734B-475E-99CE-6AA7AFA43259}
Info_a3:  guid2 == {93945629-734B-475E-99CE-6AA7AFA43259}
Info_b1:  8-4-4-4-12 format for dashes works ({BB345678-ABCD-ABCD-0000-BBBBFFFF9012})
Info_b2:  Mixed upper and lower case works.
Info_c1:  These embedded dash locations are required.  {00000000-0000-0000-0000-000000000000}
Info_d1:  Braces {} are optional ({DD345678-ABCD-ABCD-0000-DDDDAAAA9012})
```

int and int64

Integers are numbers that have no decimal places. There are two integer types: **int** and **int64**. Integers are used as control variables in repetitive statements or as indexes in arrays.

You can also use *integer literals* anywhere that an integer expression is expected, and both relational and arithmetic operators can be applied. An integer literal is the integer as it's entered directly in the code, such as **32768**. An **int** is 32-bits wide, and an **int64** is 64-bits wide. The default value is **0**, and the internal representation is a long number. An integer is automatically converted to a **boolean**, **enum**, or **real**.

Additionally, the following explicit conversion functions can be used: **str2int**, **int2str**, **str2int64**, and **int642str**. The range of an **int** is [-2,147,483,647 : 2,147,483,647], and the range of an **int64** is [-9,223,372,036,854,775,808 : 9,223,372,036,854,775,808]. All integers in either of these ranges can be used as literals.

int and int64 examples

The following example shows how to declare integers and use them in expressions. If you try to assign the largest integer plus 1 to an **int64**, you get the wrong result, because the number is interpreted as a 32-bit number. Therefore, the number is wrapped around and stored instead as -2,147,483,647. To prevent this behavior, add "u" to the end of the number. For example, enter **int64 i = 0x8000 0000u** (0x8000 0000 is 2,147,483,648).

```
public void IntegerMethod()
{
    // Declaration of an integer variable, i.
    int i;

    // Declaration of two int64 variables.
    int64 i1, i2;

    // An integer variable is initialized to the value 100.
    int i3 = 100;

    // Declaration of a dynamic array of integers.
    int i4[];
    void element()
    {
        // Two integer variables are declared and initialized.
        int k = 1, j = 2;

        // j is assigned the result of j + ((i + i) DIV 2).
        j +=(i + i) div 2;

        // This results in: j=3.

        if (j > 2 )
        {
            print "J is greater than 2";
        }
        else
        {
            print "J is NOT greater than 2";
        }
    }
}
```

real

A **real** variable can hold decimal values in addition to integers. You can use *decimal literals* anywhere that a **real** is expected. A decimal literal is the decimal as it's entered directly in the code, such as **2.123876**. Real literals can also be written by using exponential notation, such as **1.0e3**.

Reals can be used in all expressions, and they can be used with both relational and arithmetic operators. A **real** has a precision of 16 significant digits. The default value for a **real** is **0.0**, and the internal representation is a binary-coded digital (BCD) number. The BCD encoding enables exact representations of values that are multiples of 0.1. The range of a **real** variable is $-(10)^{127}$ through $(10)^{127}$. All reals in this range can be used as literals in X++.

A **real** variable is automatically converted to a **boolean**, **enum**, or **int**. If the result is an integer, or if the operator is an integer operator, the **real** is converted to an integer. If the result is a **boolean**, the **real** is converted to a **boolean**, and so on. Additionally, the following explicit conversion functions can be used: **str2num** and **num2str**.

Direct assignments between X++ **real** and the Microsoft .NET Framework **System.Decimal** convert the value correctly. A call to a conversion function isn't required. A *decimal number* is a floating-point value that consists of a sign, a numeric value where each digit is in the range 0 through 9, and a scaling factor that indicates the position of a floating decimal point that separates the integral and fractional parts of the numeric value. The binary representation of a **real** value consists of a 1-bit sign, a 96-bit integer number, and a scaling factor. The scaling factor is used to divide the 96-bit integer and specify what part of it is a decimal fraction. The scaling factor is implicitly the number 10 raised to an exponent in the range 0 through 28. Therefore, the binary representation of a decimal value represents ([-2^{96} through 2^{96}] ÷ 10(0\ through\ 28)), where $-(2^{96}-1)$ is the minimum value that can be expressed and $2^{96}-1$ is the maximum value.

> **NOTE**
>
> The type that is used to represent **real** values in Finance and Operations applications has changed from the interpreted X++ of Microsoft Dynamics AX 2012. However, you don't have to rewrite any code, because the new type can express all the values that the old type could express. We provide this material in the interest of full disclosure only.

All instances of the **real** type are now implemented as instances of the .NET decimal type (**System.Decimal**). Just as the **real** type in previous versions, the decimal type in a binary-coded decimal type is resilient to rounding errors. The range and resolution of the decimal type differ from previous versions. The original X++ **real** type supported 16 digits and an exponent that defined the position of the decimal point. However, the X++ **real** type for Finance and Operations applications can represent decimal numbers in the range 79,228,162,514,264,337,593,543,950,335 (2^{96}-1) through -79,228,162,514,264,337,593,543,950,335 (-[2^{96}-1]).

Rounding is still required for the new **real** type. For example, the following code produces a result of 0.99999999999999999999999999 instead of 1. No number of decimals will suffice to represent the value of 1/3 accurately. The discrepancy obtained here is due to the fact that only a finite number of decimals are provided. You should use the **round** function to round to the number of decimals required.

```
// An example of using the debugger to show the value of the variables.
public static void UseTheDebugger(Args a)
{
    real dividend = 1.0;
    real divisor = 3.0;
    str stringvalue;
    System.Decimal valueAsDecimal;
    real value = dividend/divisor * divisor;
    valueAsDecimal = value;
    info(valueAsDecimal.ToString("G28"));
    // An example of using the Round function to round to the number of decimals required.
    value  = round(value, 2);
}
```

real examples

```
public void RealMethod()
{
    // Simple declaration of a real variable, r.
    real r;

    // Multiple declaration of two real variables.
    real r1, r2;

    // A real variable is initialized to the approximate value of pi.
    real r3 = 3.1415;

    // Declaration of a dynamic array of reals.
    real r4[];

    // An example of a real literal written using exponential notation.
    real r;
    r = 1.000e3;
    r = 1.2345e+3;
    r = 1.2345e+03;
    r = 1234.5e4;
    r = 1.0e1; // Means 1.0E1
}

// An example of automatic conversions.
void main()
{
    // Declares a variable of type integer with the name exprValue.
```

```
    int exprValue;

    // Declares a real variable with the name area.
    real area = 3.141528;
    exprValue = Area/3;

    // The expression Area/3 is a real expression because
    // division is a real operator, and the result is 1.047176. This result is
    // automatically converted (actually truncated) to an integer with the value 1,
    // because exprValue is an integer.
}

// An example of a real being converted to .NET System.Decimal.
void AnotherMain(Args _args)
{
    real real9;
    System.Decimal sysdec1;

    // Direct assignments supported between these types.
    sysdec1 = 2.3456;
    real9 = sysdec1;
    info(strFmt("strFmt says real9 == %1", real9));
}

/***
Message (05:48:43 pm)
strFmt says real9 == 2.35
***/

// An example of using reals in expressions.
void myMethod()
{
    // Two real variables are declared and initialized.
    real i = 2.5, j = 2.5;

    // j is assigned the result of j * i, so j=6.25.
    j = j * i;
    if (j > (i * 2)) // If j > 5
    {
        print "Great"; // "Great" is printed.
    }
    else
    {
        print "Oops"; // else "Oops" is printed.
    }
}
```

str

A **str** variable (a *string*) is a sequence of characters that are used as texts, help lines, addresses, telephone numbers, and so on.

To declare a string, use the **str** keyword.

String literals are characters that are enclosed in quotation marks (""). String literals can be used wherever string expressions are expected. Examples of string literals include "StringLit" and "Hello World". If you want the string to span more than one line, precede it with an at sign (@). You can use strings in logical expressions, such as comparisons. You can also concatenate strings by using the + operator.

The default value for a string is **empty**, and the internal representation is a list of characters. There are no automatic conversions for strings, however, the following explicit conversion functions can be used: **str2int**, **str2int64**, **int2str**, **str2num**, **num2str**, **str2date**, and **date2str**.

A string can hold an unlimited number of characters, however, you can specify the maximum length of a string

in the variable declaration. The string is then truncated to that maximum length. An example is shown in the next section.

str examples

```
void StringMethod()
{
    // Declare a dynamic string of unlimited length.
    str unlimitedString;

    // Declare a string with a maximum of 64 characters
    // in order to force a truncation, initialized to "A".
    str 64 maxLengthString = "A";

    // Declare an array of 100 strings.
    str 30 hundredStrings[100];

    // Using strings in expressions.
    void myMethod()
    {
        // Two strings are declared and initialized.
        str a="Hello", b="World";

        // The concatenation of a, " " and b is printed in a window.
        print a+" "+b;
    }
}
```

timeOfDay

The **timeOfDay** (time) data type is an integer value that represents the number of seconds that have passed since midnight. Like integers, **timeOfDay** variables can be used as literals. Relational and arithmetic operators can be applied to **timeOfDay** variables. A **timeOfDay** variable can also be used in expressions. The range of a **timeOfDay** data type is in the closed interval [0; 86,400]. Values above 86,400 (23:59:59) can't be interpreted. A **timeOfDay** variable is automatically converted to a **boolean**, **enum**, or **real**. Additionally, the following explicit conversion functions can be used: **str2time** and **time2str**.

timeOfDay examples

```
public void TimeofdayMethod()
{
    // Declaration of a timeOfDay variable, time1.
    timeOfDay time1;

    // Declaration and initialization of a timeOfDay variable to 00:21:35.
    timeOfDay time2 = 1295;
}
```

utcdatetime

The **utcdatetime** data type combines the **date** type and the **timeOfDay** type. A **utcdatetime** variable also holds information about the time zone, however, this information can't be accessed in code.

The format for a **utcdatetime** literal is **yyyy-mm-ddThh:mm:ss**. The uppercase "T" is required. This format can be written without quotation marks. The minimum value is **1900-01-01T00:00:00**, and the maximum value is **1900-01-01T00:00:00**. This maximum value matches the upper range of **date** and **timeOfDay**. The smallest unit of time in **utcdatetime** is one second.

A **utcdatetime** variable that has been declared but hasn't been initialized has the default value **1900-01-**

01T00:00:00. This value is the value that is returned by **DateTimeUtil::minValue()**. Some functions treat an input parameter of this minimum value as **null**. For example, the **DateTimeUtil::toStr** method returns an empty string, however, the **DateTimeUtil::addSeconds** method returns a usable **utcdatetime** value.

There are no implicit conversions for the **utcdatetime** data type. The **DateTimeUtil** class provides many methods that you can use to manipulate **utcdatetime** values.

The following explicit conversion functions can also be used: **str2datetime** and **datetime2str**.

Additionally, the **Global** class provides the **utcDateTime2SystemDateTime** and **CLRSystemDateTime2UtcDateTime** conversion methods to support common language runtime (CLR) interop.

Comparison operators are the only type of operators that can be used with the **utcdatetime** data type. The following operators can be used to compare two **utcdatetime** values: !=, <, <=, ==, >, and >=. When you add a **utcdatetime** field to a table, we recommend that you base the field on an EDT.

utcdatetime examples

```
public void UtcdatetimeMethod()
{
    // Declaring a utcdatetime literal.
    utcdatetime myUtc2 = 1988-07-20T13:34:45;

    // Another example of declaring a utcdatetime literal.
    int iDay = DateTimeUtil::day(1988-07-20T13:34:45);

    // utcdatetime using a quoted string parameter into the DateTimeUtil::parse method.
    utcdatetime myUtc4 = DateTimeUtil::parse("1988-07-20T13:34:45");
}
```

X++ composite data types

2/18/2021 • 21 minutes to read • Edit Online

This topic describes composite data types in X++. The composite data types in X++ are arrays, containers, classes as data types, delegates as data types, and tables as data types.

Array

An *array* is a variable that contains a list of items that have the same data type. The elements of an array are accessed by using integer indexes. You use a separate statement to initialize each element in an array. When you use a container data type or an array object to create a collection, you can initialize multiple elements by using a single statement. By default, all the items in an array have the default value of the data type in the array. There are three kinds of arrays: *dynamic arrays*, *fixed-length arrays*, and *partly on disk arrays*.

- **Dynamic arrays** – These arrays are declared by using an empty array option. In other words, they have only brackets ([]).
- **Fixed-length arrays** – These arrays can hold the number of items that is specified in the declaration. Fixed-length arrays are declared like dynamic arrays, but a length option is included in the brackets.
- **Partly on disk arrays** – These arrays are declared as either dynamic arrays or fixed-length arrays that have an extra option that declares how many items should be held in memory. The other items are stored on disk and are automatically loaded when they are referenced.

X++ supports only one-dimensional arrays. However, you can mimic the behavior of multiple array indexes. (For more information, see the Multiple array indexes section). Variables in objects and tables can be declared as arrays. For example, this functionality is used in address lines in the standard application. An array collection class lets you store objects in an array.

Array indexes begin at 1. The first item in the array is referenced as [1], the second item is referenced as [2], and so on. The following syntax is used to access an array element: **ArrayItemReference = ArrayVariable [Index]**. In this syntax, **ArrayVariable** is the identifier of the array, and **Index** is the number of the array element. **Index** can be an integer expression. Item zero [0] is used to clear the array. If a value is assigned to index 0 in an array, all elements in the array are reset to their default value.

Array examples

```
public void ArrayMethod()
{
    int myArray[10]; // Fixed-length array with 10 integers.
    myArray[4] = 1; // Accessing the 4th element in the array.
    myArray[0] = 0; // Resets all elements in intArray.

    // Dynamic array of integers.
    int intArray[];

    // Dynamic array of variables of type Datatype.
    //Datatype arrayVariable[];

    // Fixed-length arrays.
    boolean boolArray[100]; // Fixed-length array of booleans with 100 items.

    // Two examples of Partly On Disk Arrays.
    // Dynamic integer array with only 100 elements in memory.
    int arrayVariable [ ,100];
    // Fixed-length string array with 1000 elements, and only 200 in memory.
    str arrayVariable2 [1000,200];

    // A dynamic array of integers.
    int i[];
    // A fixed-length real array with 100 elements.
    real r[100];
    // A dynamic array of dates with only 10 elements in memory.
    date d[,10];
    // A fixed length array of NoYes variables with 100 elements and 10 in memory.
    NoYes ny[100,10];
}
```

Multiple array indexes

Some languages, such as C++ and C#, let you declare arrays that have more than one index. In other words, you can define "arrays of arrays." In X++, you can't directly create multiple array indexes because only one-dimensional arrays are supported. However, you can implement multiple indexes by using the method that is described in this section. For example, you want to declare an array that has two dimensions, to hold an amount that is earned by country by dimension. There are 10 countries and three dimensions. In C++ and C#, you declare the following array.

```
// This is C# or C++ code, not X++ code.
real earning[10, 3];
```

However, X++ doesn't support this declaration. Instead, you can define a one-dimensional array where the number of elements is the product of the elements in each dimension. Here is an example.

```
public void MultipleArrayMethod()
{
    // Step 1: define a one-dimensional array with the number
    // of elements that is the product of the elements in each dimension.
    real earnings[10*3];

    // Step 2: to refer to a specific element, such as earnings[i,j], write the following:
    // declare i and j (maybe) and assign the value to something
    int i = 1;
    int j = 2;
    real element = earnings[(i-1)*3 + j];
}

// This can be written into a macro like this:
#localmacro.earningIndex
(%1-1)*3+%2
#endmacro

public void CallTheMacro()
{
    // Next, call the specific element within the macro like this:
    int i = 1;
    int j = 2;
    real element = earnings[#earningIndex(i,j)];

    // The previous scheme can be extended to any number of dimensions.
    // The element a[i1, i2, ..., ik] can be accessed by computing the
    // offset into an array containing (d1*d2*...*dk) elements.
    //(i1 - 1)*d2*d3*..*dk +
    //(i2 - 1)*d3*d4*...*dk + .... +
    //(ik-1 -1)*dk +
    //(ik-1)
}
```

Container

A *container* object is a dynamic list of items that contains primitive data types or composite data types. A container is useful when you must pass various types of values between the client and server tiers. However, if you plan to repeatedly add to a list in a loop, a container isn't a good choice. Containers are most suitable for processes that don't involve excessive modification to the size or contents of the container. When a container undergoes excessive additions of data, overall system performance can decrease because container data must be repeatedly copied and new space must be repeatedly allocated.

A container isn't a class. A container contains an ordered sequence of primitive values or other containers. Because of the flexibility of **anytype**, a container offers a good way to store values of different types together. A container can be stored in the database. A container is one of the column types that you can select when you use Application Explorer to add a column to a table. A container slightly resembles an array, or collections, such as the **List** or **Stack** classes. However, you can never change the size or content of a container after the container is created.

X++ statements that appear to modify a container are internally building a new container and copying values as required. Even an assignment of a container to another container variable creates a new copy of the container. All these operations can affect performance. In the functions that provide access to a container (such as **conPeek**), the container is 1-based, not 0-based. Indexing is 1-based for arrays. The default value of a container is **empty**. The container doesn't contain any values. Some statements that use containers might appear to modify a container, however, inside the system, containers are never modified. Instead, the data from the original container is combined with data from the command to build a new container. You can create a new container by using one of the following functions: **conDel**, **conIns**, or **conPoke**.

Additionally, the **Global** class has static methods for handling containers. These methods include

con2ArraySource, **con2Buf**, **con2List**, **con2Str**, **containerFromXmlNode**, **conView**, and **str2Con**. There are several intrinsic functions for handling a container, such as **conIns** and **conPeek**. The X++ **conPeek** function returns an **anytype** type, therefore, it's easier to read the values from a container when you don't know the type of each value. An **anytype** can be assigned to any X++ value type, provided that the value can be converted. Your code is easier to read when it avoids explicit data type conversions. Therefore, assign values from a container to the same data type that was used to put the value into the container. You must not assign a container to an **anytype** because the system might not be able to determine the correct conversions. In these cases, unexpected behavior or errors might occur.

Comparing container to other options

The **container** type resembles other constructs, such as arrays and collection classes, such as **List** and **Stack**. The difference between a **container** and **List** is that an instance of the **List** class is mutable. A **List** object first allocates more space than its data consumes. Then, as data is added, the space is filled. This behavior is more efficient than allocating more space every time that an element is added. An update of a **List** performs faster than similar operations on a container.

When you construct a **List** object, you determine the one type of data that the **List** object can store. This restriction is less flexible for a **List** than it is for a container. However, you can store objects in a **List**, whereas a container can store only value types. The difference between a container and an array is that an array can hold only items of its declared type. You can allocate memory space for an array and fill that space with values later. For example, you can fill in values in a loop. This behavior is efficient and performs well. When you want to build a new container by appending new data, you can use either the **+=** operator or the **conIns** function. The **+=** operator is the faster alternative. Use the **conIns** function only when you want to add new data before the last index of the original data.

In Dynamics AX 2012, although you could use the X++ compiler to store object references in containers, the result would fail at run time. However, in Finance and Operations applications, when the compiler sees an attempt to store an object reference in a container, it issues an error message. If the type of the element that is added to the container is **anytype**, the compiler can't determine whether the value is a reference type. In this case, the compiler allows the attempt. Although the compiler doesn't diagnose the code as erroneous, an error will be thrown at run time.

Container examples

```
public void ContainerExample()
{
    // First, declare the variables you are using.
    container myContainer;
    container myContainer4;
    container myContainer5;
    // Three ways to declare a container.
    myContainer = [1];
    myContainer += [2];
    myContainer4 = myContainer5;

    // Declare a container.
    container cr3;

    // Assign a literal container to a container variable.
    cr3 = [22, "blue"];

    // Declare and assign a container.
    container cr2 = [1, "blue", true];

    // Mimic container modification (implicitly creates a copy).
    cr3 += [16, strMyColorString];
    cr3 = conIns(cr3, 1, 3.14);
    cr3 = conPoke(cr3, 2, "violet");

    // Assignment of a container (implicitly creates a copy).
```

```
    cr2 = cr3;

    // Read a value from the container.
    str  myStr = conPeek(cr2, 1);

    // One statement that does multiple assignments from a container.
    str myStr;
    int myInt;
    container cr4 = ["Hello", 22, 20\07\1988];
    [myStr, myInt] = cr4; // "Hello", 22

    // Example of applying the = operator to a container. The example
    // initializes myContainer2 and myContainer33.
    myContainer2 = [2, "apple"];

    // Next, you make a copy of myContainer33 and assign the copy to myContainer2.
    myContainer33 = [33, "grape"];
    myContainer2 = myContainer33;  // The container that myContainer2 had been holding is no longer
available and cannot be recovered.
    // An example of building a new container by
    // assigning a new value to myContainer33 through the += operator.
    myContainer33 += [34, "banana"];
}

// Container example. In this example, variable2 and variable33 hold different containers.
static void JobC(Args _args)
{
    container variable2, variable33;
    variable2 += [98];
    variable33 = variable2;
    variable2 += [97];
}

// List class example. In this example, variable2 and variable33 refer to the same List object.
static void JobL(Args _args)
{
    List variable2,variable33;
    variable2 = new List(Types::Integer);
    variable2.addEnd(98);
    variable33 = variable2;
    variable2.addEnd(97);
}

// The automatic type conversion by anytype also applies to the special syntax for making multiple
// assignments from a container in one statement. This is shown in the following code example,
// which assigns a str to an int, and an int to a str.
static void JobContainerMultiAssignmentUsesAnytype(Args _args)
{
    container con2;
    int int4;
    str str7;
    con2 = ["11", 222];
    [int4, str7] = con2;
    info(strfmt("int4==11==(%1), str7==222==(%2)", int4, str7));
}

/***  Output:
Message (10:36:22 am)
int4==11==(11), str7==222==(222)
***/

static void UseQuery()
{
    // An example of how the compiler diagnoses attempts to store object in containers
    container c = [new Query()];   // This statement will cause the error message shown below.
    /*** Instance of type 'Query' cannot be added to a container. ***/

    // An example of a code that won't cause an error message, but will
    // cause an error message to be thrown at runtime.
```

```
    anytype a = new Query();
    container d = [a];
}
```

Classes as data types

A *class* is a type definition that describes both variables and methods for instances of the class. (The instances of a class are also known as *objects*.) A class is only a definition for objects, and all objects are **null** when they are declared. In Application Explorer, every application class under the **Classes** node is a data type. You can declare variables of these types in your code. You can construct instances of a class and assign the instances to variables.

Classes can be nested in source code. Nested classes are available only inside forms (such as a class that extends **FormRun**), and are used to represent controls, data sources, or data fields. An attribute decoration, such as the attribute decoration on a class or a method, can omit the suffix of the attribute name if the suffix is **Attribute**. Therefore, X++ allows **[MyFavorite]** instead of requiring **[MyFavoriteAttribute]**. Additionally, attributes are now applied to the handlers of delegates and methods, to map the handlers to those targets.

In AX 2012 and earlier versions, you could designate a method to run on either the client or the server. However, in Finance and Operations applications, all compiled X++ code is run as .NET Common Intermediate Language (CIL) on the server. There is no longer any code that is evaluated at the client site or in the browser. Therefore, the **client** and **server** keywords are now ignored. Although these keywords don't cause a compile error if they are used, they should not be used in any new code.

Private and protected member variables

Previously, all member variables that were defined in a class were protected. You can now make the visibility of member variables explicit by adding the **private**, **protected**, and **public** keywords. The interpretation of these modifiers is obvious and is aligned with the semantics for methods:

- **private** – The member variable can be used only within the class where it's defined.
- **protected** – The member variable can be used in the class where it's defined and all subclasses of that class.
- **public** – The member variable can be used anywhere. It's visible outside the confines of the class hierarchy where it's defined.

By default, member variables that aren't adorned with an explicit modifier are still protected. However, as a best practice, you should explicitly specify the visibility. As we described earlier, when a member variable is defined as **public**, it can be accessed outside the class where it's defined. In this case, you must specify a qualifier that designates the object that is hosting the variable. To specify the qualifier, use the dot notation, as you do for method calls.

In the following example, **field1** is accessed by using the explicit **this** qualifier. In this case, it might not be a good idea to make a member variable public, because that approach exposes the internal workings of the class to its consumers, and therefore creates a strong dependency between the class implementation and its consumers. You should always try to depend only on a contract, not an implementation.

```
public class AnotherClass3
{
    int field1;
    str field2;
    void new()
    {
        this.field1 = 1;   // Explicit object designated.
        field2 = "Banana";  // 'this' assumed, as usual.
    }
}
```

Static constructors and static fields

Static fields are fields that are declared by using the **static** keyword. Conceptually, static fields apply to the class, not to instances of the class. Static constructors are guaranteed to run before any static calls or instance calls are made to the class. The execution of the static constructor is relative to the user's session. You never call the static constructor explicitly. Instead, the compiler will generate code to make sure that the constructor is called exactly one time, before any other method on the class is called. A static constructor is used to initialize any static data or perform an action that must be performed only one time. You can't provide parameters for the static constructor, and it must be marked with the **static** keyword.

```
// An example of how a singleton (call instance in the example below)
// can be created using the static constructor.
public class Singleton
{
    private static Singleton instance;
    private void new()
    {
    }
    static void TypeNew()    // This is the static constructor.
    {
        instance = new Singleton();
    }

    public static Singleton Instance()
    {
        return Singleton::instance;
    }
}

// The singleton ensures that only one instance of the class
// will be called, which is consumed by the following.
{
    // Your code here.
    Singleton i = Singleton::Instance();
}
```

Class elements in Application Explorer

Under most class nodes in Application Explorer, there are two special nodes: a **classDeclaration** node and a **new** node. A **classDeclaration** always contains the X++ **class** keyword. Additional keywords, such as **extends**, can be included to modify the class. This node can also contain declarations of member variables.

In the following example, the variables **m_priority** and **m_rectangle** are members of the class.

```
// An example of a classDeclaration.
public class YourDerivedClass extends YourBaseClass
{
    int m_priority;
    Rectangle m_rectangle;
    void new(int _length, int _width)
    {
        this.m_rectangle = new Rectangle(_length, _width);
    }
}
```

A **new** operator contains logic that is run when the **new** operator is used to create an instance of the class. The logic in the **new** method might construct an object and assign that object to a variable that is declared in the **classDeclaration**. Each class can have only one **new** method. However, in the **new** method, you often should call the **new** method of the base class. To call the **new** method of the base class, call **super()**.

The following example shows the **new** method for the **YourDerivedClass** class in the previous **classDeclaration** example. In this **new** method, the code constructs an instance of the **Rectangle** class. The instance is assigned to the **m_rectangle** variable. The **this** keyword that is used in the example is optional,

however, if you include it, IntelliSense might be more helpful.

```
// An example of the new method from the previous classDeclaration example.
void new(int _length, int _width)
{
    this.m_rectangle = new Rectangle(_length, _width);
}
```

Garbage collection

Eventually during run time, most objects no longer have any variable that points to them. The system scans for these objects and erases them from memory. The memory space then becomes available for other uses. The **Object** class has a method that is named **finalize**. However, the **finalize** method isn't a destructor. The runtime never calls the **finalize** method, even when an object is collected as garbage.

System classes

In Application Explorer, under **System Documentation** > **Classes**, there is a list of the kernel classes or system classes. System classes aren't written in X++, and you can't see their source code. You can't add system classes. System classes usually have a **new** method, but they don't have a **classDeclaration** node. Every application class implicitly extends the **Object** system class. Some system classes are extended by an application class that has a similar name. For instance, **xClassFactory** is extended by **ClassFactory**. In these cases, you should not use the system class. For more information, see "Substitute application classes for system classes" in Classes and methods.

Extension methods

The extension method feature lets you add extension methods to a target class by writing the methods in a separate extension class. The following rules apply:

- The extension class must be static.
- The name of the extension class must end with the ten-character suffix **_Extension**, however, there's no restriction on the part of the name that precedes the suffix.
- Every extension method in the extension class must be declared as **public static**.
- The first parameter in every extension method is the type that the extension method extends. However, when the extension method is called, the caller must not pass in anything for the first parameter. Instead, the system automatically passes in the required object for the first parameter.
- The target of an extension method must be a class, table, view, or map application object type.

An extension class can contain private or protected static methods. These methods are typically used for implementation details and aren't exposed as extensions. The extension method technique doesn't affect the source code of the class that it extends, therefore, the addition to the class doesn't require over-layering.

Upgrades to the target class are never affected by any existing extension methods. If an upgrade to the target class adds a method that has the same name as your extension method, your extension method can no longer be reached through objects of the target class. The extension method technique uses the same dot-delimited syntax that you often use to call regular instance methods. Extension methods can access all public artifacts of the target class, but they can't access anything that is protected or private. Therefore, extension methods can be considered a type of syntactic sugar. Regardless of the target type, an extension class is used to add extension methods to the type. For example, an extension table isn't used to add methods to a table, and there's no such thing as an extension table.

```
// An example of an extension class holding a few extension methods.
public static class AtlInventLocation_Extension
{
    public static InventLocation refillEnabled(
        InventLocation _warehouse,
        boolean _isRefillEnabled = true)
    {
        _warehouse.ReqRefill = _isRefillEnabled;
        return _warehouse;
    }

    public static InventLocation save(InventLocation _warehouse)
    {
        _warehouse.write();
        return _warehouse;
    }
}
```

Delegates as data types

A *delegate* collects methods that subscribe to it. The delegate specifies the parameter signature that all its subscriber methods must share. When the delegate is called, the delegate calls each of its subscribers. A delegate never returns a value and **can't have a default value**. At first, every delegate has no subscribed methods. There is no limit on the number of parameters that a delegate can declare, and there is no limitation on the type of those parameters. The delegate body is always empty, because the delegate's only purpose is to define the contract that subscribers must conform to. A delegate doesn't have to be defined in a class. Delegates can also be defined in a table, form, or query.

Delegate examples

```
abstract class VarDatClass
{
    // delegatemethod examples
    // An example of declaring a delegate.
    delegate void notifyChange(utcdatetime _dateTime, str _changeDescription)
    {
    }

    // An example of subscribing an event handler to a delegate.
    public static void notifyStatic(utcDateTime _dateTime, str _changeDescription)
    {
        info("A notification has occurred calling static handler:" +
            DateTimeUtil::toStr(_dateTime) +
            " Message:" +
            _changeDescription);
    }
}
```

Tables as data types

All tables can be treated as class definitions. A table variable can be considered an instance (object) of the table (class) definition. For every field in a table variable, the default value is **empty**. You can address fields and create methods on tables. The methods can be invoked on instances of the table. To manipulate (that is, read, update, insert, and delete) records in tables, you must declare at least one table variable that can hold the record in focus. As a best practice, you should use the name of the table as the name of the variable but use an initial lowercase letter. Here are a few important differences between tables and objects:

- You can't allocate space for table variables. Allocation is done implicitly.
- Fields in table variables are public. You can reference them anywhere.

- Fields in table variables can be referenced by using expressions.

There is no automatic conversion, but table variables that are declared as **Common** can hold data from any table.

Scope of table variables

In most respects, table variables can be considered objects, however, unlike objects, they aren't explicitly allocated. Only a variable declaration is required. All tables are compatible with the **Common** table, just as all objects are compatible with the **Object** class. Table variables are declared as common buffers and can be used to hold data from any table. You can't access tables that don't have table variables. The principles for declaring table variables and objects are the same, except with regard to the allocation of space.

Table examples

The syntax enables various possibilities for referencing fields in records. For example, you can use the **TableName.(FieldId)** syntax.

The following example prints the contents of the fields in the current record in the Customer table.

```
// Declares and allocates space for one CustTable record.
public void myMethod()
{
    CustomerTable custTable;
}

// An example of referencing table variables.
public void printAccountNo()
{
    CustomerTable custTable;
    print custTable.AccountNo;  // Prints the field reference.
}
```

The following example uses the **fieldCnt** and **fieldCnt2Id** methods. The **fieldCnt** method counts the number of fields in a table, whereas **fieldCnt2Id** returns the ID for a field number. For example, you can use the **fieldCnt2Id** method to learn that field number 6 in a table has the ID 54. This conversion is required, because there is no guarantee that the IDs of the fields in a table are consecutive.

```
// An example of the various possibilities for referencing fields in records.
public void printCust()
{
    int i, n, k;
    CustomerTable custTable;
    DictTable dictTable;
    dictTable = new DictTable(custTable.TableId);
    n = dictTable.fieldCnt();
    print "Number of fields in table: ", n;
    for(i=1; i<=n; i++)
    {
        k = dictTable.fieldCnt2Id(i);
        print "The ", dictTable.fieldName(k),
        " field with Id=",k, " contains '",
        custTable.(k), "'";
    }
}
```

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

X++ collection classes

2/18/2021 • 2 minutes to read • Edit Online

This topic describes collection classes in X++.

The X++ language syntax provides two composite types: arrays and containers. These composite types are useful for aggregating values of primitive types. However, you can't store class objects in arrays or containers.

Collection classes are used to store objects. They let you create arrays, lists, sets, maps, and structs that can hold any data type, even objects. For maximum performance, the classes are implemented in C++ (they are system classes). Collection classes were previously known as *foundation classes*. The collection classes are **Array**, **List**, **Map**, **Set**, and **Struct**.

- **Array** – This class resembles the **array** type in the X++ language, but it can hold values of any single type, even objects and records. Objects are accessed in a specific order.
- **List** – This class contains elements that are accessed sequentially. Unlike an array, the **List** class provides an **addStart** method. Like the **Set** class, the **List** class provides the **getEnumerator** and **getIterator** methods. You can use an iterator to insert and delete items from a **List** object.
- **Map** – This class associates a key value with another value.
- **Set** – This class holds values of any single type. Values aren't stored in the sequence in which they are added. Instead, the **Set** object stores the value in a manner that optimizes performance for the **in** method. A **Set** object ignores any attempt to add a value that the **Set** object is already storing. Unlike the **Array** class, the **Set** class provides the **in** and **remove** methods.
- **Struct** – This class can contain values of more than one type. It's used to group information about a specific entity.

The constructor for every collection class except **Struct** takes a type parameter that is an element of the **Types** system enum. The collection instance can store items of that type only. The **Types::AnyType** enum element is a special case that can't be used to construct a collection object, such as a **Set** object. The **null** value can't be stored as an element in a **Set** object. Additionally, **null** can't be a key in a **Map** object. You can iterate through a collection object by using an iterator or enumerator. Here are typical examples that show how you can obtain an iterator.

```
new MapIterator(myMap)
myMap.getEnumerator()
```

For **Set** objects, if any elements are added or removed after an iterator is created, the iterator instance can no longer be used to read from or step through the collection.

For **Map** objects, as for **Set** objects, if any elements are removed, the iterator is no longer valid. However, a **MapIterator** object remains valid even after a call to the **Map.insert** method, regardless of whether the key is new, or whether the key already exists and only the value is being updated in the **Map** element. Code that calls **Map.insert** and depends on the iterator object remaining valid might fail if it's run as .NET Framework CIL.

You can use the collection classes to form more complex classes. For example, you can easily implement a stack by using a list where elements are always added to the beginning of the list. The newest element then occupies the top of the stack.

You can also extend the collection classes. For example, you can extend the **List** class to create a list of customer records where the operations are type-safe. In this case, the derived collection class will accept only customer records.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

X++ extended data types

2/18/2021 • 2 minutes to read • Edit Online

This topic describes extended data types in X++.

Extended data types are user-defined types that are based on the **boolean**, **int**, **int64**, **real**, **str**, and **date** primitive data types, and on the **container** composite type. An EDT is a primitive data type or container that has a supplementary name and additional properties. For example, you can create a new EDT that is named **Name** and base it on a string. You can then use the new EDT in variable and field declarations in the development environment.

You can also base EDTs on other EDTs. EDTs are standard data types, but they have a specific name and additional properties. EDTs undergo the same value and type conversions as the standard data types that they are based on. Here are the benefits of EDTs:

- Code is easier to read, because variables have a meaningful data type. For example, the data type is **Name** instead of **str**.
- The properties that you set for an EDT are used by all instances of that type. Therefore, EDTs help reduce work and promote consistency. For example, account numbers (**AccountNum** data type) have the same properties throughout the system.
- You can create hierarchies of EDTs. The EDTs can inherit the appropriate properties from the parent, and you can change other properties. For example, the **ItemCode** data type is used as the basis for the **MarkupItemCode** and **PriceDiscItemCode** data types.

Create an EDT

This feature isn't implemented as a language construct. To create an EDT, follow these steps.

1. In Solution Explorer, right-click on the project, point to **Add**, and then click **New item**.
2. In the **Add New Item** dialog box, select **Installed** and then **Artifacts** in the left pane.
3. In the middle pane, select the EDT type to create.
4. Enter a name, and then click **Add**.

EDT example

```
public void EdtMethod()
{
    // Example of declaring EDT variables where
    // a UserGroupID (integer) variable is declared and initialized to 1.
    UserGroupID groupID = 1;

    // An Amount (real) variable is declared.
    Amount currency;
}
```

Comments, using, and print statements

2/18/2021 • 5 minutes to read • Edit Online

This topic describes statements in X++.

Comments

It's a good practice to add comments to your code. Comments make a program easier to read and understand. Comments are ignored when the program is compiled. Your comments can use either the **//** style or the **/*** style. However, a best practice is to use the **//** style for comments, and even for multiline comments.

```
// This is an example of a comment.
/* Here is another example of a comment. */
```

print statements

You use the **print** statement to output text through **System.Diagnostics.WriteLine** to the Visual Studio **Output** window. During testing, the **print** statement is an alternative to the **Global::info** method, which shows text in the **Infolog** window. The following table compares the **print** statement and the **info** method.

FEATURE	PRINT STATEMENT	INFO METHOD
Ease of invocation	The **print** statement automatically converts various data types into strings. It can convert multiple data types in one invocation.	The **info** method requires that the input parameter be a string.
Ability to copy contents to the clipboard	Text is easily copied from the **Output** window to the clipboard.	Text is easily easily copied from the **Infolog** window to the clipboard.
Typical usage	The **print** statement is used for convenience during testing. It can help you debug small issues without having to run a formal debugger.	The **info** method is appropriate for use in production.

Example of a print statement

The following code example demonstrates the print statement automatically converting any date type to a string. You do not need to prefix **info** with **Global::** when you call it.

```
str hello = "Hello";
int fortytwo = 42;
utcDateTime now = DateTimeUtil::utcNow();
Dialog dialog = new Dialog();

print "The print statement automatically converts data types to strings.";
print hello, " -- ", fortytwo, " -- ", now, " -- ", dialog;
// Output to the Print window:
// The print statement automatically converts data types to strings.
// Hello -- 42 -- 10/3/2011 09:18:10 pm -- 1

// int2Str converter is needed when using info().
info("Hello");
info(int2Str(fortytwo));

// Output to Infolog window:
// Hello
// 42
```

TODO comments

The compiler recognizes the string **TODO** when it occurs at the start of a comment. The **TODO** string prompts the compiler to report the rest of the comment text in the **Task List** window in Microsoft Visual Studio. To open the **Task List** window, select **View**, and then select **Task Window**. The **Task Window** reports the line number where the **TODO** comment can be found in the code.

Here are the rules for using **TODO** in comments:

- The **TODO** string can appear in a comment that uses either the **//** style or the **/*** style.
- The **TODO** string must be the very first non–white space string in the comment. A carriage return, a line feed, a tab, and a space are all considered white space.
- No white space is required between the start of the comment and the **TODO**.
- The **TODO** string is case-insensitive. However, the convention is to type **TODO** in all uppercase letters, instead of **ToDo** or another variation.
- The **TODO** string can have any characters appended to it. However, the convention is either to append a colon to the **TODO** string or to follow it with a white space.
- The rest of the comment after the **TODO** string is reported as the task description. If the comment is longer than 200 characters, it might appear truncated on the **Tasks** tab.
- The **TODO** task description can be spread over multiple lines when the **/*** comment style is used.

Examples of TODO comments

The following examples show **TODO** comments.

```
// An example of using TODO in the // style of comment.
public boolean isLate()
{
    // TODO: Finish this stub.
    return true;
}

// An example of using TODO in the /* */ style of comment.
public boolean isLate()
{
    /* TODO Finish this stub */
    return true;
}
```

Unsupported statements: changeSite, pause, and window

The **changeSite**, **pause**, and **window** keywords are no longer a part of the X++ language. These keywords will cause compilation errors if you use them.

using clauses

You use **using** clauses so that you don't have to provide the fully qualified name of a type. The **using** clause must precede the class that it applies, and it's required in every source file that you want it to apply to. Typically, all **using** clauses are put at the beginning of the source file. You can also provide aliases that introduce a short name for a fully qualified name. Aliases can denote namespaces or classes.

The following example shows a **using** clause, a namespace alias, and a class alias.

```
using System;
using IONS=System.IO; // Namespace alias
using Alist=System.Collections.ArrayList; // Class alias

class UsingClass
{
    public static void test()
    {
        Int32 I;                  // Alternative to System.Int32
        Alist al = new Alist();   // Using a class alias
        al.Add(1);
        str s = IONS.Path::ChangeExtension(@"c:\tmp\test.xml", ".txt"); // Using a namespace alias
    }
}
```

using statements

The **using** statement helps guarantee that objects that implement **IDisposable** are disposed of correctly. When you use an **IDisposable** object, you should declare and instantiate it in a **using** statement. The **using** statement calls the **Dispose** method on the object in the correct way, even if an exception occurs while you're calling methods on the object. You can achieve the same result by putting the object inside a **try** block and then explicitly calling **Dispose** in a **finally** block. The **using** statement simplifies the syntax and disposes of the object correctly. Here is the syntax for a **using** statement:

using (*expression* **) { *statement* }**

In this syntax, *statement* can be a block of statements, and *expression* declares and instantiates an object that implements **IDisposable**. The following example creates and uses a **StreamReader** object.

```
static void AnotherMethod()
{
    str textFromFile;
    using (System.IO.StreamReader sr = new System.IO.StreamReader("c:\\test.txt"))
    {
        textFromFile = sr.ReadToEnd();
    }
}
```

X++ conditional statements

2/18/2021 • 4 minutes to read • Edit Online

This topic describes conditional statements in X++. The conditional statements are **if**, **if...else**, **switch**, and the ternary operator (**?**). You use conditional statements to specify whether a block of code is executed. Different conditional statements offer advantages in different situations.

if and if...else statements

The **if** statement evaluates a conditional expression, and then executes a statement or a set of statements if the conditional expression is evaluated as **true**. You can use the **else** clause to provide an alternative statement or set of statements that is executed if the condition is evaluated as **false**. The syntax for an **if...else** statement is:

if (*expression* **)** *statement* **[else** *statement* **]**

In this syntax, both occurrences of *statement* can be compound statements (statements enclosed in braces). The *expression* in the parentheses (the conditional expression) can be any valid expression that is evaluated as **true** or **false**. All numbers except 0 (zero) are **true**. All non-empty strings are **true**. You can nest **if** statements. However, if the nesting of **if** statements becomes too deep, you should consider using a **switch** statement instead.

Examples of if and if...else statements

```
// if statement
if (a > 4)
{
   info("a is greater than 4");
}

// if... else statement
if (a > 4)
{
   info("a is greater than 4");
}
else
{
   info("a is less than or equal to 4");
}
```

switch statements

The **switch** statement is a multibranch language construct that has the same behavior as nested **if**. The expression of the **switch** statement is evaluated and checked against each case value. The case values must be constants that the compiler can evaluate.

- If a case constant matches the **switch** expression, the **case** statement is executed.
- If the case contains a **break** statement, the program then jumps out of the switch.
- If the case doesn't contain a **break** statement, the program continues and executes the next **case** statements.
- If no matches are found, the **default** statement is executed.
- If there are no matches and no **default** statement, none of the statements inside the **switch** statement are executed.

Here is the syntax for a **switch** statement:

switch (*expression*) { { **case** } [**default**: *statement*] }

The syntax for a **case** statement is:

case *expression* { , *expression* } : *statement*

In the syntax for both a **switch** statement and a **case** statement, every occurrence of *statement* can be replaced with a block of statements by enclosing the block in braces ({}).

Examples of switch statements

When you include the **break** keyword in a switch statement, the execution of the case branch terminates, and the statement following the switch is executed. As shown in the following example, if the Debtor account number is 1000, the program executes "do something", and then continues execution after the switch statement.

```
switch (Debtor.AccountNo)
{
    case "1000":
        // do something
        break;
    case "2000":
        // do something else
        break;
    default:
        // default statement
        break;
}
```

The following code examples makes the execution drop through the first case branch by omitting a break statement. If x is 10, b is assigned to a, and d is assigned to c. If x is 11, d is assigned to c. If x is 12, f is assigned to e.

```
 switch (x)
 {
     case 10:
         a = b;
     case 11:
         c = d;
         break;
     case 12:
         e = f;
         break;
 }
```

If you do not use the break statement, the program flow in the switch statement continues into the next case. Code segments A and B have the same behavior.

```
// Code segment A (break omitted)
case 13:
case 17:
case 21:
case 500:
    info("g");
    break;

// Code segment B (the values are comma-delimited)
case 13, 17, 21, 500;
    info("g");
    break;
```

Ternary operator (?)

The ternary operator (**?**) is a conditional statement that is resolved to one of two expressions. The result can be assigned to a variable. By contrast, an **if** statement provides conditional branching of the program flow but can't be assigned to a variable. Here is the syntax for the ternary operator:

expression1 **?** *expression2* **:** *expression3*

In this syntax, *expression1* must return a value of **true** or **false**. If *expression1* is **true**, the whole ternary statement returns *expression2*. Otherwise, the statement returns *expression3*. Both *expression2* and *expression3* must have the same type.

Examples of the ternary operator (?)

The following code example returns one of two strings based on a Boolean return value from a method call. The Boolean expression indicates whether the CustTable table has a row with a RecId field value of 1. If this Boolean expression is true (meaning RecId != 0), found is assigned to result. Otherwise, the alternative not found is assigned to result.

```
result = (custTable::find("1").RecId) ? "found" : "not found";
```

You can nest statements with the ternary operator. The following example assigns one of three values to **level** based on the value of **x**.

```
int x = 1001;
str level = x <= 1000 ? "A" : (x <= 2000 ? "B" : "C");
info(level);
// Output is "B".
```

X++ loop statements

2/18/2021 • 3 minutes to read • Edit Online

This topic describes loop statements in X++.

There are three loop statements: **for**, **while**, and **do...while**. A loop repeats its statement until the condition that is set for the loop is **false**. Within the loop statements, you can use **break** and **continue** statements.

for loops

The syntax of a **for** loop is:

for (*initialization* **;** *test* **;** *increment* **) {** *statement* **}**

The **for** loop repeatedly executes **statement** for as long as the conditional expression *test* is **true**. *statement* can be a block of statements. The body of the **for** loop (*statement*) might be executed zero or more times, depending on the results of *test*.

A **for** loop differs from other loops because an initial value can be assigned to a control variable, and because there is a statement for incrementing or decrementing the variable. These additions make a **for** loop especially useful for traversing lists, containers, and arrays because they have a fixed number of elements.

You can also apply a statement to each element and increment your way through the elements, setting the condition to test for the last element.

Example of a for loop

In the following code example, the items in an array of integers are printed.

```
int integers[10];
for (int i = 0; i < 10; i++)
{
    info(int2str(integers[i]));
}
// The output is a series of 0's.
```

while loops

The syntax of a **while** loop is:

while (*expression* **)** *statement*

A **while** loop repeatedly executes *statement* for as long as the conditional *expression* is **true**. *statement* can be replaced by a block of statements. *statement* is executed as many times as the condition is met (zero to many).

Example of a while loop

The following code example demonstrates a **while** loop that traverses a container and prints out the contents of the container.

```
container cont = ["one", "two", "three"];
int no = 0;
while (no <= conlen(cont))
{
    info(conPeek(cont, no));
    no++;
}
// The output is "one", "two", "three".
```

do...while loops

The syntax of the **do...while** loop is:

do { *statement* } **while** (*expression*) ;

The **do**...**while** loop is similar to the **while** loop, but the condition appears after the *statement* that must be executed. *statement* can be a block of statements. The *statement* is always executed at least one time, because the condition is tested after *statement* is executed. The **do**...**while** loop is well-suited to tasks that must always be done at least one time, such as getting parameters for a report.

Example of a do...while loop

The following code example finds the smallest power of 10 that is larger than `realNumber`.

```
int FindPower(real realNumber)
{
    int exponent = -1;
    real curVal;

    do
    {
        exponent++;
        curVal = power(10, exponent);
    }
    while (realNumber > curVal);

    return exponent;
}
```

continue statement

The **continue** statement causes execution to move directly to the next iteration of a **for**, **while**, or **do**...**while** loop. For **do** or **while**, the test is executed immediately. For a **for** statement, the increment step is executed.

Example of a continue statement

In the following code example, if `Iarray[i] <= 0`, the remaining statements in the loop are not executed, and `i` is incremented before the **if** statement is tried again.

```
int Iarray[100];
for (int i = 0; i < 100; i++)
{
    if (Iarray[i] <= 0)
    {
        Info("Will continue.");
        continue;
    }

    info("Did not continue.");
}
// The output is "Will continue." for all 100 interations.
```

break statement

The **break** statement within a loop is used to terminate that loop. Execution then moves to the first statement after the loop.

Example of a break statement

This example is uses a **break** statement within a **while** loop. When used within a loop, the loop is terminated and execution continues from the statement following the loop. This works for **do... while** and **for** loops as well.

```
var mainMenu = SysDictMenu::newMainMenu();
var enum = mainMenu.getEnumerator();
var found = false;
while (enum.moveNext())
{
    var menuItem = enum.current();
    if (menuItem.label() == "StringOfInterest")
    {
        found = true;
        break;
    }
}
if (found)
{
    // do something
}
```

X++ exception handling

2/18/2021 • 13 minutes to read • Edit Online

This topic describes exception handling in X++. You handle errors by using the **throw**, **try**...**catch**, **finally**, and **retry** statements to generate and handle exceptions.

An *exception* is a regulated jump away from the sequence of program execution. The instruction where program execution resumes is determined by **try**...**catch** blocks and the type of exception that is thrown. An exception is represented by a value of the **Exception** enumeration, or an instance of .NET's System.Exception class or a derived class. One exception that is often thrown is the **Exception::error** enum value. A common practice is to write diagnostic information to the Infolog before the exception is thrown.

The **Global::error** method is often the best way to write diagnostic information to the Infolog. For example, your method might receive an input parameter value that isn't valid. In this case, the method can throw an exception to immediately transfer control to a **catch** code block that contains logic for handling this error situation. You don't necessarily have to know the location of the **catch** block that will receive control when the exception is thrown.

throw statements

You use the **throw** keyword to throw an **Exception** enum value. For example, the following statement throws an error exception.

```
throw Exception::error;
```

Instead of throwing an enum value, a best practice is to use the output of the **Global::error** method as the operand for **throw**.

```
throw Global::error("The parameter value is invalid.");
```

The **Global::error** method can automatically convert a label into the corresponding text. This functionality helps you write code that can be localized more easily.

```
throw Global::error("@SYS98765");
```

The static methods on the **Global** class can be called without the **Global::** prefix. For example, the **Global::error** method can be called like this.

```
error("My message.");
```

In Platform update 31 or later versions, the **throw** keyword can be used to throw .NET exceptions.

```
throw new InvalidOperationException("This function is not allowed");
```

Also in Platform update 31 or later, the **throw** keyword can be used by itself inside a catch block. In such a case, **throw** will behave like the **rethrow** statement in C#. The original exception, exception message and its context such as call stack will be rethrown and be available to any catch statements in calling code.

```
try
{
    throw Exception::error;
}
catch
{
    // locally handle exception
    // then rethrow for caller
    throw;
}
```

try, catch, finally, and retry statements

When an exception is thrown, it's first processed through the **catch** list of the innermost **try** block. If a **catch** block is found that handles the kind of exception that is being thrown, program control jumps to that **catch** block. If the **catch** list has no block that specifies the exception, the system passes the exception to the **catch** list of the next-innermost **try** block. The **catch** statements are processed in the same sequence as they appear in the code.

It's a common practice to have the first **catch** statement handle the **Exception::Error** enum value. One strategy is to have the last **catch** statement leave the exception type unspecified. In this case, the last **catch** statement handles all exceptions that aren't handled by any earlier **catch** statement. This strategy is appropriate for the outermost **try**...**catch** blocks.

An optional ***finally*** clause can be included in **try**...**catch** statements. The semantics of a **finally** clause are the same as they are in C#. The statements in the **finally** clause are executed when control leaves the **try** block, either normally or through an exception.

The **retry** statement can be written only in a **catch** block. The **retry** statement causes control to jump up to the first line of code in the associated **try** block. The **retry** statement is used when the cause of the exception can be fixed by the code in the **catch** block. The **retry** statement gives the code in the **try** block another opportunity to succeed. The **retry** statement erases all messages that have been written to the Infolog since program control entered the **try** block.

> **NOTE**
> You must make sure that your **retry** statements don't cause an infinite loop. As a best practice, the **try** block should include a variable that you can test to find out whether you're in a loop.

```
try
{
    // Code here.
}
catch (Exception::Numeric)
{
    info("Caught a Numeric exception.");
}
catch
{
    info("Caught an exception.");
}
finally
{
    // Executed no matter how the try block exits.
}
```

The system exception handler

If no **catch** statement handles the exception, it's handled by the system exception handler. The system exception handler doesn't write to the Infolog. Therefore, an unhandled exception can be hard to diagnose. We recommended that you follow all these guidelines to provide effective exception handling:

- Have a **try** block that contains all your statements in the outermost frame on the call stack.
- Have an unqualified **catch** block at the end of your outermost **catch** list.
- Avoid throwing an **Exception** enum value directly.
- Throw the enum value that is returned from one of the following methods on the **Global** class: **Global::error**, **Global::warning**, or **Global::info**. (You can omit the implicit **Global::** prefix).
- When you catch an exception that hasn't been shown in the Infolog, call the **Global::info** function to show it.

Exception::CLRError, **Exception::UpdateConflictNotRecovered**, and system kernel exceptions are examples of exceptions that aren't automatically shown in the Infolog.

Exceptions and CLR interop

You can call Microsoft .NET Framework classes and methods that reside in assemblies that are managed by the common language runtime (CLR). When a .NET Framework **System.Exception** instance is thrown, your code can catch it by declaring a variable of type **System.Exception** to catch any .NET exception, or one of its derived classes to catch a specific .NET exception type as shown in the following example.

```
System.ArgumentException ex;
try
{
    throw new System.ArgumentException("Invalid argument specified");
}
catch(ex)
{
    error(ex.Message);
}
```

In releases prior to Platform update 31, .NET exceptions can be caught by referencing **Exception::CLRError**. Your code can obtain a reference to the **System.Exception** instance by calling the **CLRInterop::getLastException** method.

```
try
{
    // call to .NET code which throws exception
}
catch(Exception::CLRError)
{
    System.Exception ex = CLRInterop::getLastException();
    error(ex.Message);
}
```

Ensuring that exceptions are shown

Exceptions of the **Exception::CLRError** type aren't shown in the Infolog, because these exceptions aren't issued by a call to a method such as **Global::error**. In your **catch** block, your code can call **Global::error** to report the specific exception.

Global class methods

This section describes some **Global** class methods in more detail. These class methods include **Global::error**, **Global::info**, and **Global::exceptionTextFallThrough**.

Global::error method

The following code shows how the **error** method is declared.

```
static Exception error
    (SysInfoLogStr txt,
    URL helpURL = '',
    SysInfoAction _sysInfoAction = null)
```

The return type is the **Exception::Error** enum value. The **error** method doesn't throw an exception. It just provides an enum value that can be used in a **throw** statement. The **throw** statement throws the exception. Here are descriptions of the parameters for the **error** method. Only the first parameter is required.

- **SysInfoLogStr** txt is a **str** of the message text. It can also be a label reference, such as **strFmt("@SYS12345", strThingName)**.
- The **URL** helpUrl is a reference to the location of a Help topic in Application Explorer, such as **"KernDoc:\\\\Functions\\substr"**. The parameter value is ignored if _sysInfoAction is supplied.
- The **SysInfoAction** is an instance of a class that extends the **SysInfoAction** class. The method overrides that we recommend for the child class are the **description** method, the **run** method, the **pack** method, and the **unpack** method.

Global::info method

The **Global::info** method is often used to show text in the Infolog. In programs, it's often written as **info("My message.");**. Although the **info** method returns an **Exception::Info** enum value, you will rarely want to throw **Exception::Info**, because nothing unexpected has occurred.

Global::exceptionTextFallThrough method

Occasionally, you want to do nothing inside your **catch** block. However, the X++ compiler generates a warning if you have an empty **catch** block. To avoid this warning, call the **Global::exceptionTextFallThrough** method in the **catch** block. The method does nothing, but it satisfies the compiler and explicitly states the intention.

Exceptions inside transactions

If an exception is thrown inside a transaction, the transaction is automatically canceled (that is, a **ttsAbort** operation occurs). This behavior applies for both exceptions that are thrown manually and exceptions that the system throws. When an exception is thrown inside a **ttsBegin**-**ttsCommit** transaction block, no **catch** statement inside that transaction block can process the exception, (unless it is a **UpdateConflict** or a **DuplicateKeyException**). Instead, the innermost **catch** statements that are outside the transaction block are the first **catch** statements that are tested.

Examples of exception handling

Showing exceptions in the Infolog

The following code example shows exceptions in the Infolog.

```
// This example shows that a direct throw of Exception::Error does not
// display a message in the Infolog. This is why we recommend the
// Global::error method.
static void TryCatchThrowError1Job(Args _args)
{
/***
    The 'throw' does not directly add a message to the Infolog.
    The exception is caught.
***/
    try
    {
        info("In the 'try' block. (j1)");
        throw Exception::Error;
    }
    catch (Exception::Error)
    {
        info("Caught 'Exception::Error'.");
    }

/**********  Actual Infolog output
Message (03:43:45 pm)
In the 'try' block. (j1)
Caught 'Exception::Error'.
**********/
}
```

Using the error method to write exception information to the Infolog

The following code example uses the **error** method to write exception information to the Infolog.

```
// This example shows that the use of the Global::error method
// is a reliable way to display exceptions in the Infolog.
static void TryCatchGlobalError2Job(Args _args)
{
    /***
    The 'Global::error()' does directly add a message to the Infolog.
    The exception is caught.
    ***/
    try
    {
        info("In the 'try' block. (j2)");
        throw Global::error("Written to the Infolog.");
    }
    catch (Exception::Error)
    {
        info("Caught 'Exception::Error'.");
    }

/***  Infolog output
Message (03:51:44 pm)
In the 'try' block. (j2)
Written to the Infolog.
Caught 'Exception::Error'.
***/
}
```

Handling a CLRError

The following code example handles a **CLRError** exception.

```
// This example shows that a CLRError exception is not displayed
// in the Infolog unless you catch the exception and manually
// call the info method. The use of the CLRInterop::getLastException
// method is also demonstrated.
static void TryCatchCauseCLRError3Job(Args _args)
{
    /***
    The 'netString.Substring(-2)' causes a CLRError,
    but it does not directly add a message to the Infolog.
    The exception is caught.
    ***/
    System.String netString = "Net string.";
    System.Exception netExcepn;
    try
    {
        info("In the 'try' block. (j3)");
        netString.Substring(-2); // Causes CLR Exception.
    }
    catch (Exception::Error)
    {
        info("Caught 'Exception::Error'.");
    }
    catch (Exception::CLRError)
    {
        info("Caught 'Exception::CLRError'.");
        netExcepn = CLRInterop::getLastException();
        info(netExcepn.ToString());
    }

/**********  Actual Infolog output (truncated for display)
Message (03:55:10 pm)
In the 'try' block. (j3)
Caught 'Exception::CLRError'.
System.Reflection.TargetInvocationException: Exception has been thrown by the target of an invocation. --->
    System.ArgumentOutOfRangeException: StartIndex cannot be less than zero.
Parameter name: startIndex
    at System.String.InternalSubStringWithChecks(Int32 startIndex, Int32 length, Boolean fAlwaysCopy)
    at System.String.Substring(Int32 startIndex)
    at ClrBridgeImpl.InvokeClrInstanceMethod(ClrBridgeImpl* , ObjectWrapper* objectWrapper, Char*
pszMethodName,
    Int32 argsLength, ObjectWrapper** arguments, Boolean* argsAreByRef, Boolean* isException)
**********/
}
```

Using a retry statement

The following code example uses a **retry** statement.

```
// This example shows how to use the retry statement. The print
// statements are included because retry causes earlier Infolog
// messages to be erased.
static void TryCatchRetry4Job(Args _args)
{
    /***
    Demonstration of 'retry'. The Infolog output is partially erased
    by 'retry', but the Print window is fully displayed.
    ***/
    Exception excepnEnum;
    int nCounter = 0;
    try
    {
        info("        .");
        print("        .");
        info("In the 'try' block, [" + int2str(nCounter) + "]. (j4)");
        print("In the 'try' block, [" + int2str(nCounter) + "]. (j4)");
        nCounter++;
        if (nCounter >= 3) // Prevent infinite loop.
        {
            info("---- Will now throw a warning, which is not caught.");
            print("---- Will now throw a warning, which is not caught.");
            throw Global::warning("This warning will not be caught. [" + int2str(nCounter) + "]");
        }
        else
        {
            info("Did not throw a warning this loop. [" + int2str(nCounter) + "]");
            print("Did not throw a warning this loop. [" + int2str(nCounter) + "]");
        }
        excepnEnum = Global::error("This error message is written to the Infolog.");
        throw excepnEnum;
    }
    catch (Exception::Error)
    {
        info("Caught 'Exception::Error'.");
        print("Caught 'Exception::Error'.");
        retry;
    }
    info("End of job.");
    print("End of job.");

/**********  Actual Infolog output
Message (04:33:56 pm)
            .
In the 'try' block, [2]. (j4)
---- Will now throw a warning, which is not caught.
This warning will not be caught. [3]
**********/
}
```

Throwing an exception inside a transaction

The following code example throws an exception in a transaction block.

```
// This examples uses three levels of try nesting to illustrate
// where an exception is caught when the exception is thrown inside
// a ttsBegin-ttsCommit transaction block.
static void TryCatchTransaction5Job(Args _args)
{
    /***
    Shows an exception that is thrown inside a ttsBegin - ttsCommit
    transaction block cannot be caught inside that block.
    ***/
    try
    {
        try
        {
            ttsbegin;
            try
            {
                throw error("Throwing exception inside transaction.");
            }
            catch (Exception::Error)
            {
                info("Catch_1: Unexpected, caught in 'catch' inside the transaction block.");
            }
            ttscommit;
        }
        catch (Exception::Error)
        {
            info("Catch_2: Expected, caught in the innermost 'catch' that is outside of the transaction
block.");
        }
    }
    catch (Exception::Error)
    {
        info("Catch_3: Unexpected, caught in 'catch' far outside the transaction block.");
    }
    info("End of job.");

/**********  Actual Infolog output
Message (04:12:34 pm)
Throwing exception inside transaction.
Catch_2: Expected, caught in the innermost 'catch' that is outside of the transaction block.
End of job.
**********/
}
```

Using Global::error with a SysInfoAction parameter

When your code throws an exception, it can write messages to the Infolog. You can make those Infolog messages more helpful by using the **SysInfoAction** class.

In the following example, a **SysInfoAction** parameter is passed in to the **Global::error** method. The **error** method writes the message to the Infolog. When the user double-clicks the Infolog message, the **SysInfoAction.run** method is run.

In the **run** method, you can write code that helps diagnose or fix the issue that caused the exception. The object that is passed in to the **Global::error** method is constructed from a class that you write that extends **SysInfoAction**.

The following code sample is shown in two parts.

- The first part shows a job that calls the **Global::error** method and then throws the returned value. An instance of the **SysInfoAction_PrintWindow_Demo** class is passed in to the **error** method.
- The second part shows the **SysInfoAction_PrintWindow_Demo** class.

Part 1: Calling Global::error

```
static void Job_SysInfoAction(Args _args)
{
    try
    {
        throw Global::error
            ("Click me to make the Print window display."
            ,""
            ,new SysInfoAction_PrintWindow_Demo()
            );
    }
    catch
    {
        warning("Issuing a warning from the catch block.");
    }
}
```

Part 2: The SysInfoAction_PrintWindow_Demo class

```
public class SysInfoAction_PrintWindow_Demo extends SysInfoAction
{
    str m_sGreeting; // In classDeclaration.
    public str description()
    {
        return "Starts the Print Window for demonstration.";
    }
    public void run()
    {
        print("This appears in the Print window.");
        print(m_sGreeting);

        /********** Actual Infolog output
        Message (03:19:28 pm)
        Click me to make the Print window display.
        Issuing a warning from the catch block.
            ***************/
    }
    public container pack()
    {
        return ["Packed greeting."]; // Literal container.
    }
    public boolean unpack(container packedClass, Object object = null)
    {
        [m_sGreeting] = packedClass;
        return true;
    }
}
```

List of exceptions

The following table shows the exception literals that are the values of the **Exception** enumeration.

EXCEPTION LITERAL	DESCRIPTION
Break	The user pressed Break or Ctrl+C.
CLRError	An error occurred while the CLR functionality was being used.
CodeAccessSecurity	An error occurred while the **CodeAccessPermission.demand** method was being used.

EXCEPTION LITERAL	DESCRIPTION
DDEerror	An error occurred while the **DDE** system class was being used.
Deadlock	A database deadlock occurred, because several transactions are waiting for each other.
DuplicateKeyException	An error occurred in a transaction that is using Optimistic Concurrency Control. The transaction can be retried (use a **retry** statement in the **catch** block).
DuplicateKeyExceptionNotRecovered	An error occurred in a transaction that is using Optimistic Concurrency Control. The code won't be retried. This exception can't be caught inside a transaction.
Error	A fatal error occurred. The transaction has been stopped.
Info	This exception literal holds a message for the user. Don't throw an **info** exception.
Internal	An internal error occurred in the development system.
Numeric	An error occurred while the **str2int**, **str2int64**, or **str2num** function was being used.
Sequence	
UpdateConflict	An error occurred in a transaction that is using Optimistic Concurrency Control. The transaction can be retried (use a **retry** statement in the **catch** block).
UpdateConflictNotRecovered	An error occurred in a transaction that is using Optimistic Concurrency Control. The code won't be retried. This exception can't be caught within a transaction.
Warning	An exceptional event has occurred. Although the user might have to take action, the event isn't fatal. Don't throw a **warning** exception.
SQL connection error X++ exception	An error occured when during the query execution. The transaction will be canceled. This exception can't be caught within a transaction.

SQL connection error X++ exception

2/18/2021 • 2 minutes to read • Edit Online

This topic describes the SQL connection error exception types in X++.

TransientSqlConnectionError X++ exception

During an X++ SQL query execution, when a transient SQL connection error occurs on the server side, a TransientSqlConnectionError X++ exception will occur. Depending on the application requirements, the application should catch and handle the exception.

This exception usually occurs during a large transaction or when the database is under a lot of processing pressure.

The TransientSqlConnectionError exception is not catchable within the transaction. The X++ transaction that encounters this exception is canceled (calling **ttsAbort**) before the exception occurs. This means that you need to use the catch block to identify the transient SQL connection error instead of a generic X++ error exception, and then retry the outermost transaction or retry application code logic in a new session. This exception allows the application to be designed for transient server failures.

If an application transaction takes a long time to process, you can use multiple incremental delays to catch the TransientSqlConnectionError exception. Retrying your application code in a new session is most likely to succeed after you have caught the exception.

Example

```
public static void LargeTransactionWrapper()
{
    try
    {
        LargeTransaction();
    }
    catch (Exception::TransientSqlConnectionError)
    {
        info("Caught transient SQL connection error, ttslevel=" + int2Str(appl.ttsLevel()));
        // At this point, transaction is canceled
        // Code that indicates retry is possible
    }
    finally
    {
        // Do clean up
    }
}
```

X++ operators

2/18/2021 • 14 minutes to read • Edit Online

This topic describes the operators supported in X++.

Assignment operators

An assignment changes the value of a variable or field. The following table shows the X++ assignment operators. There is no difference between prefix and postfix operators.

OPERATOR	DESCRIPTION
`=`	Assign the expression on the right of the equal sign to the variable on the left.
`+=`	Assign the current variable value plus the expression on the right to the variable on the left.
`++`	Increment the variable by 1.
`-=`	Assign the current variable value minus the expression on the right to the variable on the left.
`--`	Decrement the variable by 1.

Code examples for assignment operators

```
// An example of assignment operators and their output.
static void Example1()
{
    int i = 1;
    // Using the = operator. i is assigned the value of i, plus 1. i = 2.
    i = i + 1;
    info(strFmt("Example 1: The result is "), i); // The result is 2.
}

static void Example2()
{
    int i = 1;
    // Using the += operator. i is assigned the value of i, plus 1.
    // i = 2 (i = i + 1).
    i += 1;
    info(strFmt("Example 2: The result is "), i); // The result is 2.
}

static void Example3()
{
    int i = 1;
    // Using the ++ operator. i is incremented by 1, and then
    // by 1 again in the second statement. The final value of i is 3.
    i++;
    ++i;
    info(strFmt("Example 3: The result is "), i); // The result is 3.
}

static void Example4()
{
    int i = 1;
    // Using the -= operator. i is assigned the value of i minus 1.
    // i = 0 (i = i - 1).
    i -= 1;
    info(strFmt("Example 4: The result is "), i); // The result is 0.
}

static void Example5()
{
    int i = 1;
    // Using the -- operator. i is decremented by 1, and then by
    // 1 again in the second statement. The final value of i is -1.
    i--;
    --i;
    info(strFmt("Example 5: The result is "), i); // The result is -1.
}
```

Arithmetic operators

You use arithmetic operators to perform numeric calculations. Most of the operators are binary and take two operands. However, the **not** (`~`) operator is unary and takes only one operand. Syntax for binary operators: *expression1 ArithmeticOperator expression2* Syntax for unary operators: *ArithmeticOperator expression1*

OPERATOR	DESCRIPTION
`<<`	The **left shift** operator performs *expression2* left shift (multiplication by 2) on *expression1*.
`>>`	The **right shift** operator performs *expression2* right shift (division by 2) on *expression1*.

OPERATOR	DESCRIPTION
`\*`	The **multiply** operator multiplies *expression1* by *expression2*.
`/`	The **divide** operator divides *expression1* by *expression2*.
`DIV`	The **integer division** operator performs an integer division of *expression1* by *expression2*.
`MOD`	The **integer remainder** operator returns the remainder of an integer division of *expression1* by *expression2*.
`~`	The **not** operator, or unary operator, performs a binary not operation.
`&`	The **binary AND** operator performs a binary and operation on *expression1* and *expression2*.
`^`	The **binary XOR** operator performs a binary XOR-operation on *expression1* and *expression2*.
`|`	The **binary OR** operator performs a binary or operation on *expression1* and *expression2*.
`+`	The **plus** operator adds *expression1* to *expression2*.
`-`	The **minus** operator subtracts *expression2* from *expression1*.
`?`	The **ternary** operator takes three expressions: *expression1* ? *expression2* : *expression3*. If *expression1* is true, *expression2* is returned. Otherwise, *expression3* is returned.

Code examples for arithmetic operators

```
int a = 1 << 4;      // Perform four left shifts on 1 (1*2*2*2*2). a=16.
int b = 16 >> 4;     // Perform four right shifts on 16 (16/2/2/2/2). b=1.
int c = 4 * 5;       // Multiply 4 by 5. c=20.
int d = 20 / 5;      // Divide 20 by 5. d=4.
int e = 100 div 21;  // Return the integer division of 100 by 21. e=4 (4*21 = 84, remainder 16).
int f = 100 mod 21;  // Return the remainder of the integer division of 100 by 21. f=16.
int g = ~1;          // Binary negate 1 (all bits are reversed). g=-2.
int h = 1 & 3;       // Binary AND. Return the bits that are in common in the two integers. h=1.
int i = 1 | 3;       // Binary OR. Return the bits that are set in either 1 or 3. i=3.
int j = 1 ^ 3;       // Binary XOR. Return the bits that are set in 1 and NOT set in 3, and vice versa. j=2.
int k = 1 + 3;       // Add 1 and 3. k=4.
int l = 3 - 1;       // Subtract 1 from 3. l=2.
int m = (400 > 4) ? 1 : 5;  // If 400>4, 1 is returned. Otherwise, 5 is returned. Because 400>4, 1 is
returned. m=1.
```

Expression operators

The `as` and `is` expression operators control downcast assignments. Downcast assignments involve class or table inheritance. Assignment statements that implicitly downcast can cause errors that are difficult to predict and diagnose. You can use the `as` keyword to make your downcasts explicit. You can use the `is` keyword to

test whether a downcast is valid at run time.

The as keyword

Use the `as` keyword for assignments that downcast from a base class variable to a derived class variable. The `as` keyword tells other programmers and the compiler that you believe that the downcast will be valid during run time.

- The compiler reports an error for downcast assignment statements that lack the `as` keyword.
- At run time, the `as` keyword causes the downcast assignment statement to assign `null` if the downcast isn't valid.
- This `is` keyword is often used to safely test whether the `as` keyword will work.

Code example for the as keyword

In the following code example, the **DerivedClass** class extends the **BaseClass** class. The code example contains two valid assignments between its **basec** and **derivedc** variables. The upcast assignment to **basec** doesn't require the `as` keyword, but the downcast assignment to **derivedc** does require the `as` keyword. The following code will compile and run without errors.

```
static void AsKeywordExample()
{
    // DerivedClass extends BaseClass.
    BaseClass basec;
    DerivedClass derivedc;
    // BottomClass extends DerivedClass.
    BottomClass bottomc;
    derivedc = new DerivedClass();
    // AS is not required for an upcast assignment like this.
    basec = derivedc;
    // AS is required for a downcast assignment like this.
    derivedc = basec as DerivedClass;
    bottomc = new BottomClass();
    // AS causes this invalid downcast to assign null.
    bottomc = basec as DerivedClass;
}
```

The is keyword

The `is` keyword verifies whether an object is a subtype of a specified class. The `is` expression returns **true** if the object is a subtype of the class, or if the object is the same type as the class. The compiler reports an error if an `is` keyword expression compares two types, but neither type is a subtype of the other, and they aren't of the same type. The compiler reports a similar error for any plain assignment statement between two types, where neither type is a subtype of the other, and they aren't of the same type. At run time, the type of variable that references the underlying object is irrelevant to the `is` keyword. The `is` keyword causes the system to verify the object that the variable references, not the declared type of the variable that references the object.

Code examples for the is keyword

The following code examples illustrate the conditions that control whether an `is` expression returns **true** or **false**. The code examples depend on the fact that the **Form** class and the **Query** class both extend the **TreeNode** class.

```
// The compiler issues an error for the following code.
// The compiler ascertains that the Form class and the Query class are not
// part of the same inheritance hierarchy. Both the Form class and the Query class
// extend the TreeNode class, but neither Form nor Query is a subtype of the other.
Form myForm = new Form();
info(strFmt("%1", (myForm is Query)));

// The Infolog displays 0 during run time, where 0 means false. No supertype
// object can be considered to also be of its subtype class.
TreeNode myTreeNode = new TreeNode();
info(strFmt("%1", (myTreeNode is Form)));

// The Infolog displays 0 during run time, where 0 means false. A null
// reference causes the is expression to return false.
Form myForm;
info(strFmt("%1", (myForm is Form)));

// The Infolog displays 1 during run time, where 1 means true.
// An object is an instance of its own class type.
Form myForm = new Form();
info(strFmt("%1", (myForm is Form)));

// The Infolog displays 1 during run time, where 1 means true.
// Every subtype is also of its supertype.
Form myForm = new Form();
info(strFmt("%1", (myForm is TreeNode)));

// The Infolog displays 1 during run time, where 1 means true.
// The type of the underlying object matters in the is expression,
// not the type of the variable that references the object.
Form myForm = new Form();
TreeNode myTreeNode;
myTreeNode = myForm; // Upcast.
info(strFmt("%1", (myTreeNode is Form)));
```

Code example for the is and as keywords

The following code example contains a typical use of the `is` keyword. The `as` keyword is used after the `is` keyword verifies that the `as` keyword will succeed. In this example, the `is` and `as` keywords are uppercase to make them more visible.

```
static void IsKeywordExample()
{
    DerivedClass derivedc;
    BaseClass basec;
    basec = new DerivedClass();  // An upcast.
    if (basec IS DerivedClass)
    {
        info("Test 1: (basec IS DerivedClass) is true. Good.");
        derivedc = basec AS DerivedClass;
    }
    basec = new BaseClass();
    if (!(basec IS DerivedClass))
    {
        info("Test 2: !(basec IS DerivedClass) is true. Good.");
    }
}

//Output to the Infolog
Test 1: (basec IS DerivedClass) is true. Good.
Test 2: (!(basec IS DerivedClass)) is true. Good.
```

Object class as a special case

The **Object** class can appear as a special case in inheritance functionality. The compiler bypasses type checking

for assignments to and from variables that are declared to be of type **Object**. Some classes inherit from the **Object** class, some classes inherit from another class, and some classes don't inherit from any class. Although the **Dialog** class doesn't inherit from any class, the assignment and call statements in the following code example work. However, if the assignment is `bank4 = dlog3;`, it will fail at compile time, because the **Bank** and **Dialog** classes have no inheritance relationship to each other. The compiler performs only one small validation on assignments to a variable that is declared to be of the **Object** class. The compiler verifies that the item that is being assigned to the **Object** variable is an instance of a class. The compiler doesn't allow an instance of a table buffer to be assigned to the **Object** variable. Additionally, the compiler doesn't allow primitive data types, such as `int` or `str`, to be assigned to the **Object** variable.

```
static void ObjectExample()
{
    Bank bank4;
    Object obj2;
    Dialog dlog3 = new Dialog("Test 4.");
    obj2 = dlog3;  // The assignment does work.
    obj2.run(false);  // The call causes the dialog to appear.
    info("Test 4a is finished.");
}
```

Tables

All tables inherit directly from the Common system table, unless they explicitly inherit from a different table. The Common table can't be instantiated. It doesn't exist in the underlying physical database. The Common table inherits from the **xRecord** class, but in a special way that isn't appropriate for the `is` keyword or the `as` keyword. When the `as` keyword is used to perform an invalid downcast among tables, the target variable references an unusable non-null entity. Any attempt to de-reference the target variable will cause an error that stops the program.

The is and as keywords and extended data types

Each extended data type has an **Extends** property. The style of inheritance that this property controls differs from the style of inheritance that the `is` and `as` keywords are designed for.

Relational operators

The following table lists the relational operators that can be used in X++. Most of the operators are binary and take two operands. However, the **not** (`!`) operator is unary and takes only one operand. Syntax for binary operators: *expression1 relationalOperator expression2* Syntax for unary operators: *relationalOperator expression1*

OPERATOR	DESCRIPTION
`like`	The **like** relational operator returns **true** if *expression1* is like *expression2*.
`==`	The **equal** relational operator returns **true** if both expressions are equal.
`>=`	The **greater than or equal to** relational operator returns **true** if *expression1* is greater than or equal to *expression2*.
`<=`	The **less than or equal to** relational operator returns **true** if *expression1* is less than or equal to *expression2*.

OPERATOR	DESCRIPTION
`>`	The **greater than** relational operator returns **true** if *expression1* is greater than *expression2*.
`<`	The **less than** relational operator returns **true** if *expression1* is less than *expression2*.
`!=`	The **not equal** relational operator returns **true** if *expression1* differs from (that is, if it isn't equal to) *expression2*.
`&&`	The **and** relational operator returns **true** if both *expression1* and *expression2* are true.
`||`	The **or** relational operator returns **true** if *expression1* or *expression2* is true, or if both are true.
`!`	The **not** or **unary** relational operator negates the expression. It returns **true** if the expression is false and **false** if the expression is true.

The like operator

The `like` operator can use `*` as a wildcard character for zero or more characters, and `?` as a wildcard character for one character. The maximum length of the operand is 1,000 characters. The `like` operator is evaluated by the underlying SQL, so the result might differ on different installations. If the expressions that you're comparing contain a file path, you must include four backslashes between each element, as shown in the following example.

```
select * from xRefpaths
where xRefPaths.Path like "\\\\Classes\\\\AddressSelectForm"
```

The equal (==) operator

When you use the **equal** (`==`) operator to compare objects, the object references are compared, not the objects themselves. This behavior might cause issues if you compare two objects, one of which is located on the server, and the other of which is located on the client. In these cases, you should use the **equal** method in the **Object** class. You can override this method to specify what it means for two objects to be equal. If you don't override the **equal** method, the comparison is identical to the comparison that is done by the **equal** (`==`) operator.

Code examples for relational operators

```
"Jones" like "Jo?es"  // Returns true, because the ? is equal to any single character.
"Fabrikam, Inc." like "Fa*"  // Returns true, because the * is equal to zero or more characters.
(( 42 * 2) == 84)  // Returns true, because 42*2 is equal to 84.
today() >= 1\1\1980  // Returns true, because today is later than January 1, 1980.
((11 div 10) >= 1)  // Returns true, because 11 div 10 is 1 (therefore, >= 1 is true).
(11<= 12)  // Returns true, because 11 is less than 12.
((11 div 10) > 1)  // Returns false, because 11 div 10 is 1.
(11 div 10) < 1)  // Returns false, because 11 div 10 is 1.
(11 != 12)  // Returns true, because 11 is not equal to 12.
(1 == 1) && (3 > 1)  // Returns true, because both expressions are true.
```

Operator precedence

The order that a compound expression is evaluated in can be important. For example, `(x + y / 100)` gives a

different result, depending on whether the addition or the division is done first. You can use parentheses (`()`) to explicitly tell the compiler how it should evaluate an expression. For example, you can specify `(x + y) / 100`. If you don't explicitly tell the compiler the order that you want operations to be done in, the order is based on the precedence that is assigned to the operators. For example, the division operator has higher precedence than the addition operator. Therefore, for the expression `x + y / 100`, the compiler evaluates `y / 100` first. In other words, `x + y / 100` is equivalent to `x + (y / 100)`. To make your code easy to read and maintain, be explicit. Use parentheses to indicate which operators should be evaluated first. The following table lists the operators in order of precedence. The higher an operator appears in the table, the higher its precedence. Operators that have higher precedence are evaluated before operators that have lower precedence. Note that the operator precedence of X++ isn't the same as the operator precedence of other languages, such as C# and Java.

OPERATOR GROUPS, IN ORDER OF PRECEDENCE	OPERATORS
Unary	`- ~ !`
Multiplicative, shift, bitwise **AND**, bitwise exclusive **OR**	`* / % DIV << >> & ^`
Additive, bitwise inclusive **OR**	`+ -`
Relational, equality	`< <= == != > >= like as is`
Logical (**AND**, **OR**)	`&&` `||`
Conditional	`? :`

Operators on the same line have equal precedence. If an expression includes more than one of these operators, it's evaluated from left to right, unless assignment operators are used. (Assignment operators are evaluated from right to left.) For example, `&&` (logical `AND`) and `||` (logical `OR`) have the same precedence, and are evaluated from left to right. Therefore:

- `0 && 0 || 1` is equal to `1`
- `1 || 0 && 0` is equal to `0`.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Operator precedence

2/18/2021 • 2 minutes to read • Edit Online

This topic describes operator precedence.

The order in which a compound expression is evaluated is important. If you do not explicitly tell the compiler the order that you want operations to be performed in, the order is based on operator precedence. You can use parentheses `( )` to explicitly tell the X++ compiler how you want an expression to be evaluated.

Consider the expression `x + y / 100`, which gives a different result depending on whether the addition or the division is performed first. Because the division operator has a higher precedence than the addition operator, the compiler evaluates `y/100` first. So, `x + y / 100` is equivalent to `x + (y / 100)`. If you add parentheses to make the expression `(x + y)/ 100`, then `x + y` is evaluated first.

To make your code easy to read and maintain, be explicit, and indicate with parentheses which operators should be evaluated first.

Order of operator precedence

The operators in the following table are listed in precedence order. The higher in the table an operator appears, the higher precedence it has. Operators with higher precedence are evaluated before operators with a lower precedence. The operator precedence of X++ is not the same as other languages, for example C# and Java.

OPERATORS IN PRECEDENCE ORDER	SYNTAX
unary operators	`- ~ !`
multiplicative, shift, bitwise AND, bitwise exclusive OR	`* / % DIV << >> & ^`
additive, bitwise inclusive OR	`+ -`
relational, equality	`< <= == != > >= like as is`
logical operators (AND, OR)	`&& ||`
conditional	`? :`

Operators on the same line in the table have equal precedence. If there are more than one of these operators in an expression, the expression is evaluated from left to right unless assignment operators are used. Assignment operators are evaluated from right to left. For example, `&&` (logical AND) and `||` (logical OR) have the same precedence and are evaluated from left to right. This means that `0 && 0 || 1 == 1`, and `1 || 0 && 0 == 0`

Classes and methods

2/18/2021 • 18 minutes to read • Edit Online

This topic describes how to create and use classes in X++.

A *class* is a software construct that defines the data and methods of the instances that are later constructed from that class. The *class* is an abstraction of an *object* in the problem domain. The instances that are constructed from the *class* are known as *instances* or *objects*. This topic uses the term *instance*. The data represents the state of the object, whereas the methods represent the behavior of the object.

Variables contain the data for the class. Every instance that is constructed from the class declaration has its own copy of the variables. These variables are known as *instance variables*.

Methods define the behavior of a class. They are the sequences of statements that operate on the data (instance variables). By default, methods are declared to operate on the instance variables of the class. These methods are known as *instance methods* or *object methods*.

You can declare *static methods* and *static fields*, that do not have access to *instance variables*. These are described in X++ static classes.

Declare a class

You must use the **Add new item** dialog in Visual Studio to add a class to your project.

1. In Server Explorer, right-click the project, and then click **Add**.
2. In the **New Item** dialog box, select **Installed > Dynamics 365 Items > Code** in the left navigation. Then select **Class**, and then enter a name for the class.
3. Click **Add**.

All classes are public. If you remove the **public** modifier, the system still treats the class as public. You can specify other modifiers on the class declaration, such as **final** and **extends**.

Variables

Instance variables are **protected** by default. This means that they can only be accessed in the same class or a derived class. You can modify an instance variable declaration by using the **private**, **protected**, or **public** keywords.

The following example shows how to use accessor methods to make the variable data public. The variable **firstName** is protected, so accessor (get and set) methods are implemented to allow access to the protected variable. The variable **lastName** is public, so code can directly get and set the value of the variable.

```
// This is the class definition.
public class HasAFirstName
{
    str firstName;
    public str lastName;
    public str getFirstName()
    {
        return firstName;
    }

    public void setFirstName(str newName)
    {
       firstName = newName;
    }
}

// This code creates an instance of the class and gets the variables.
public static void TestLastName()
{
    HasAFirstName hasFirstName = new HasAFirstName();
    hasFirstName.setFirstName("Dion");
    info(hasFirstName.getFirstName());
    hasFirstName.lastName = ("Townes");
    info(hasFirstName.lastName);
}
// The output is "Dion" and "Townes".
```

Constructors

To create an instance of a class, you must instantiate it by using a *constructor*.

- You can define only one **new** method (constructor) in a class.
- If you do not define a constructor, a default constructor with no parameters is created automatically by the compiler.
- You can simulate a default constructor by assigning default values to the parameters in the **new** method.

The following examples defines a parameterless constructor in the **Point** class.

```
class Point
{

    // Instance variables that are public. In practice, you would probably make this protected or private
    // and create accessor methods.
    public real x = 0.0;
    public real y = 0.0;

    void new() {
    }
}
```

Following is information about how to create a clean inheritance model and minimize problems when code is upgraded:

- Each class must have a single public construction method unless the class is abstract. If no initialization is required, use a static construct method. Otherwise, use a static **new** method (the default constructor for the class should be protected).
- Each class should have at least one static **construct** method method.
- Each class should have at least one static **new** method.
- Each class should have a **new** method (the default constructor). This method should be **protected**.

- Create accessor methods to get and set class variables.
- Create **init** methods to carry out any specialized initialization tasks that should be carried out after instantiation.

Create other objects in a constructor

A class constructor can instantiate other objects in addition to creating an instance of the class. For example, the following code declares a **Rectangle** class that uses two **Point** objects to define its bounds. In this case, the **Point** class has a constructor that has two **real** parameters.

```
class Point
{
    // Instance variables that are public. In practice, you would probably make this protected or private
    // and create accessor methods.
    public real x = 0.0;
    public real y = 0.0;

    // Constructor to initialize to a specific or default value
    void new(real _x = 10, real _y = 10)
    {
        x = _x;
        y = _y;
    }
}

class Rectangle
{
    public Point lowerLeft;
    public Point upperRight;

    void new(real _topLeftX = 0.0, real _topLeftY = 0.0, real _bottomRightX = 1.0, real _bottomRightY = 1.0)
    {
        lowerLeft  = new Point(_topLeftX, _topLeftY);
        upperRight = new Point(_bottomRightX, _bottomRightY);
    }

}

// This code creates two instances of the Rectangle class.
Rectangle defaultRectangle = new Rectangle();
info(any2Str(defaultRectangle.lowerLeft.x));
info(any2Str(defaultRectangle.lowerLeft.y));
// Output is "0.0" and "0.0".

Rectangle customRectangle = new Rectangle(1.0, 1.0, 2.0, 2.0);
info(any2Str(customRectangle.lowerLeft.x));
info(any2Str(customRectangle.lowerLeft.y));
// Output is "1.0" and "1.0".
```

Create an instance of an object

The constructor, **new**, returns a new instance of the class. The following code example creates two instances of the Point class.

```
// Declare a variable to refer to a Point instance.
Point myPoint;

// Create an instance of the Point class.
myPoint = new Point();

// Declare and instantiate a Point instance.
Point ap = new Point();
```

Destructors

You use a *destructor* to explicitly destroy a class instance. Instances are automatically destroyed when there are no references to them. However, you can destroy objects explicitly in the following ways:

- Use the **finalize** method.
- Set the reference variable to **null**.

Use the finalize method

Use the **finalize** method to explicitly destroy an object. There are no implicit calls to the **finalize** method. You must call the method to run the statements in it. In the **finalize** method, you should also put any clean-up code that is required. For example, if your class uses a dynamic-link library (DLL) module, you can use the **finalize** method to release the DLL when you no longer require it. Use the **finalize** method carefully. It will destroy an object even if there are references to it.

The following example shows the basic structure for a call to the **finalize** method.

```
// From any method in a class.
if (condition)
{
    // Removes object from memory.
    this.finalize();
}
```

Set reference variable to null

Set the reference variable to **null** to terminate an object. This approach destroys an object only if no other variables point to that object. You should verify that other code isn't using the variable. The following example creates an reference variable and then sets it to **null**.

```
Point myPoint = new Point();
myPoint = null;
```

Methods

Instance methods

Instance methods are embedded in each instance that is created from the class. You must instantiate the object before you can use the method. The following code shows how to define an instance method and call it from an instance.

```
class Square
{

    int side = 0;

    void new(int _side = 1) {
        side = _side;
    }

    int getArea() {
        return side * side;
    }

}

// This code creates an instance of Square and calls getArea.
Square square = new Square(15);
int area = square.getArea();
info(int2Str(area));
// Output is "225".
```

Static methods

Static methods, which are also known as *class methods*, belong to a class and are created by using the keyword **static**. You don't have to instantiate an object before you use static methods. Static methods are often used to work with data that is stored in tables. Member variables can't be used in a static method.

You use the following syntax to call static methods.

```
ClassName::methodName();
```

If you convert an instance method to a static method, you must restart the client. Otherwise, the compiler doesn't detect the change. After you've converted an instance method to a static method, you can no longer call the method from the instance of the class. Instead, you must call the method from the class itself. For more information about static methods, see X++ static classes.

main methods

A **main** method is a class method that is run directly from a menu option. The method should only create an instance of the object and then call the required member methods. The **_args** parameter lets you transfer data to the method.

```
static void main (Args _args)
{
    // Your code here.
}
```

Declaration of methods

Method declarations consist of a header and a body. The method header declares the method's name and return type), the method modifiers, and parameters. (The return type might be **void**.) The method body consists of variable declarations, method declarations, and statements.

Return type

A return type is required for each method. If a method doesn't return anything, use the **void** keyword as the return type.

The following example shows two methods. One method has a return type, but the other method doesn't have a return type.

```
void methodNameNoReturnValue()
{
    // Your code here.
}

// If a method returns something, you must specify the return type and include a return statement.
int methodNameIntegerReturnValue()
{
    return 1;
}
```

Syntax

Method declaration = *Heading Body* Heading = **[** *Modifiers* **]** *ReturnType MethodName* **(** *ParameterList* **)**

Modifiers = **[client] [server] [edit | display | public | protected | private]** [**static | abstract | final]**

ReturnType = *Datatype* **| void | anytype**

MethodName = *Identifier*

ParameterList = **[** *Parameter* **{ ,** *Parameter* **}]**

Parameter = *Datatype Variableidentifier* **[** = *Expression* **]**

Body = **{ [** *VariableDeclarations* **] [** *EmbeddedFunctionDeclarations* **] [** *Statements* **] }**

EmbeddedFunctionDeclaration = *Heading* **{[** *VariableDeclarations* **] [** *Statements* **]}**

If you use the **anytype** return type, the method can return any data type.

Example of a method that doesn't have a return type

```
void update ()
{
    // Variable declared and initialized
    CustTable this_Orig = this.orig();

    // First statement in body (begin transaction)
    ttsBegin;
    this.setNameAlias();
    // Calls super's implementation of update
    super();
    this.setAccountOnVend(this_Orig);
    if (this_Orig.custGroup != this.custGroup)
        ForecastSales::setCustGroupId(
            this.accountNum,
            this_Orig.custGroup,
            this.custGroup);
    // Commits transaction
    ttsCommit;
}
```

Example of a method that has parameters

In the following example, the **checkAccountBlocked** method returns a Boolean value and acts on the **amountCur** parameter.

```
boolean checkAccountBlocked(AmountCur amountCur)
{
    if (this.blocked == CustVendorBlocked::All
        ||(this.blocked == CustVendorBlocked::Invoice
        && amountCur > 0 ))
    return checkFailed(strFmt("@SYS7987",this.accountNum));
    return true;
}
```

Method modifiers

Several modifiers can be applied to method declarations. Some of the modifiers can be combined (for example, **final static**). Here are the method modifier keywords:

- **abstract**: The method is declared but isn't implemented in a parent class. The method must be overridden in subclasses. If you try to create an object from a subclass where one or more abstract methods that belong to the parent class haven't been overridden, you receive a compiler error.

  Classes can also be abstract. Sometimes, a class should not be instantiated even though it represents an abstract concept. Only subclasses should be instantiated. Base classes of this type can be declared as **abstract**. For example, you want to model the concept of an account. Accounts are abstract, because only derived classes (ledger accounts and so on) exist in the real world. This examples describes a clear case where you should declare the **Account** class as **abstract**.

- **display**: The method's return value should be shown on a page or a report. The value can't be modified on the page or report. Typically, the return value is a calculated value, such as a sum.

- **edit**: The method's return type should be used to provide information for a field that is used on a page. The value in the field can be modified.

- **final**: The method can't be overridden in any class that derives from its class.

- **public**: Methods that are declared as **public** can be accessed anywhere that the class is accessible, and they can be overridden by subclasses. Methods that have no access modifier are implicitly public.

- **protected**: Methods that are declared as **protected** can be called only from methods in the class and in subclasses that extend the class where the method is declared.

- **private**: Methods that are declared as **private** can be called only from methods in the class where the private method is declared.

- **static**: The method is a class method and doesn't act on an instance. Static methods can't refer to instance variables. They aren't invoked on an instance of the class. Instead, they are invoked by using the class name (for example, **MyClass::aStaticProcedure()**).

Methods that have modifiers

The following examples show only the method headers.

```
// A method that cannot be overridden
final int dontAlterMe()

// A static method
static void noChange()

// A display method that returns an integer
display int value()
```

Method access control

You use the accessor keywords **public**, **protected**, and **private** to control whether the methods in other classes can call the methods on your class. The accessor keywords on methods also interact with the rules for class inheritance. Here are the accessor keywords that you use with methods:

- **public**: Methods that are declared as **public** can be called from anywhere that the class is accessible. In addition, a public method can be overridden by a subclass, unless the method is declared as **final**.
- **protected**: Methods that are declared as **protected** can be called only from the following methods:
  - Methods in the class.
  - Methods in a subclass of the class that contains the protected method. Methods that are protected can be overridden in subclasses.
- **private**: Methods that are declared as **private** can be called only from methods in the class where the private method is declared. No private method can be overridden in a subclass. By default, when you create a new method, the **private** accessor keyword appears in the code editor. For maximum security, **private** is the most conservative default accessor keyword.

Static and instance methods

The accessor keywords on methods never restrict calls between two methods that are in the same class, regardless of which method is static or non-static. In a static method, calls to the **new** constructor method are valid even if the **new** constructor method is decorated with the **private** modifier. The syntax for these calls requires that the **new** keyword be used. The code in a static method must construct an instance object of its own class before it can call any instance methods on the class.

Increasing access during overrides

When a method is overridden in a subclass, the overriding method must be at least as accessible as the overridden method. For example, the following compiler rules apply when a protected method is overridden in a subclass:

- A public method in a superclass can be overridden only by a public method in the subclass.
- In a subclass, a public or protected method can override a protected method of the superclass.
- In a subclass, a private method can't override a protected method of the superclass.

Optional parameters

Parameters can be initialized in the method declaration. In this case, the parameter becomes an *optional parameter*. If no value is supplied in the method call, the default value is used. All required parameters must be listed before the first optional parameter. The following examples show how to create and call a method that has optional parameters. The example of the **AddThreeInts** method shows that you can't skip default parameters when you call a method.

Examples of optional parameters

The following code example shows a class with a default parameter.

```
// This is an example of a function being used as the default.
class Person
{
    date birthDate;

    // The constructor that takes a date type as a parameter.
    // That value is assigned to the field member birthDate.
    void new(date _date)
    {
        birthDate = _date;
    }

    // The CalculateAgeAsOfDate method references the birthDate field and has an
    // optional parameter. In this example, the default value is the
    // return value of a function.
    public real CalculateAgeAsOfDate(date _calcToDate =
DateTimeUtil::getToday(DateTimeUtil::getUserPreferredTimeZone()) )
    {
        return (_calcToDate - birthDate) / 365;
    }

    public static void callPerson()
    {

        Person person = new Person(13\5\2010);

        // Optional parameter's default is used.
        Info(strFmt('Age in years today is %1 years',
                real2int(person.CalculateAgeAsOfDate())));

        // January 2, 2044  is the parameter value for _date.
        Info(strFmt('Age in years on %1 is %2 years',
                2\1\2044,
                real2int(person.CalculateAgeAsOfDate(2\1\2044))));
    }

}
```

This is an example of how you cannot skip to a second optional parameter. The **AddThreeInts** method has two optional parameters. The **callAdditions** method calls the **AddThreeInts** method. The commented out code tries to override only the **_i3** default value, but the compiler requires that all prior optional parameters also be overridden in the call.

```
class Additions
{
    public static int AddThreeInts(int _i1, int _i2 = 2,int _i3 = 3)
    {
        return _i1 + _i2 + _i3;
    }

    public static void callAdditions()
    {
        // The next statement does not compile, because it skips the _i2 parameter.
        // info(int2Str(Additions::AddThreeInts(1, , 99)));

        // You must specify both optional parameters.
        info(int2Str(Additions::AddThreeInts(1, 2, 99)));
    }

}
```

Accessor methods

Class variables are protected by default. By hiding details of the internal implementation of a class, you can change the implementation of the class later without breaking any code that uses that class. To access the data from reference variables, you must create accessor methods. The following example defines a **Point** class that uses accessor methods to access the variables **x** and **y**.

```
class Point
{
    // Instance variables
    real x;
    real y;

    // Constructor to initialize to a specific or default value
    void new(real _x = 10, real _y = 10)
    {
        x = _x;
        y = _y;
    }

    // Accessor methods
    void setX(real _x)
    {
        x = _x;
    }

    void setY(real _y)
    {
        y = _y;
    }

    real getX()
    {
        return x;
    }

    real getY()
    {
        return y;
    }
}
```

These method declarations show how the **Point** class provides access to its variables from the outside world. Other objects can manipulate the instance variables of **Point** objects by using the accessor methods.

```
Point myPoint = new Point();
// Set the x variable using the accessor method.
myPoint.setX(4.0);
// Get the x variable using the accessor method.
info(any2Str(myPoint.getX()));
```

Parameters

All methods have their own *scope*. A method can take one or more parameters. Within the scope of the method, these parameters are treated as local variables and are initialized with a value from the parameter in the method call. All parameters are passed by value, which means that you can't change the value of the original variable. You can change only the local variable in the method. This local variable is a copy of the original variable.

Scope of variables in methods

A scope defines the area in which an item can be accessed. Variables that are defined in a class are available to the methods within that class. Variables in methods can be accessed only within the current block.

Local functions

You can declare functions inside a method. These are called local functions. While possible, it is not a best practice. Instead, you should add private methods to the class.

- The declarations of local functions must physically precede any non-declaration statements in the method.
- You can declare more than one local function in your method. However, all local functions must be declared in an uninterrupted series, and the set must be terminated by one semicolon (;).
- Code that is inside the local function can access variables that are declared in the method that contains the local function.
- Code that is outside the local function can't access variables that are declared in the local function.
- A local function can be called only by code in the same method where the local function is declared.
- A local function should never call itself. Such recursion can prevent successful compilation.

The following example shows valid declarations of two local functions, **localFunctionA** and **localFunctionB**. Calls to the local functions occur after the function declarations in the example, as is required.

```
static void StaticFunction()
{
    int number = 654;

    void localFunctionA(int _iNum)  // The local function.
    {
        str innerString = "String in localFunctionA";
        str output = strFmt("localFunctionA: %1 , %2 , %3", _iNum, innerString, number);
        info(output);
    }

    void localFunctionB()
    {
        info("Printing from inside localFunctionB.");
    }

    localFunctionA(55);
    localFunctionB();
    // Next info statement would fail to compile,
    // because innerString is restricted to the
    // scope of the local function in which it is declared.
    // print innerString;
}

// When called, the output is:
// localFunctionA: 55 , String in localFunctionA , 654
// Printing from inside localFunctionB.
```

The this keyword

The **this** keyword is a reference to the instance of the class or table where the **this** keyword is used. The **this** reference is never required, but it can clarify your code and enhances the behavior of IntelliSense in the code editor. All calls to instance methods must be qualified by either the **this** reference or a variable. The **this** reference can be used to qualify the following information:

- The names of other instance (non-static) methods in the same class where the **this** reference is used. Here is an example: `boolColorChanged = this.colorItOrange();`
- The names of methods that are inherited by the **this** object.
- The names of fields on the table that contains the method that the **this** keyword is used in.

The **this** reference can't be used in the following ways:

- It can't qualify the names of member variables that are declared in the **classDeclaration** code.
- It can't be used in a static method.
- It can't qualify the names of static methods of the class or table.

Call stack limitation

The depth of the call stack is limited to 100.

X++ inheritance

2/18/2021 • 3 minutes to read • Edit Online

This topic describes inheritance in X++, including how to create a subclass and override a method.

Creating a subclass

Subclasses are classes that extend or inherit from other classes. A class can extend only one other class. Multiple inheritance isn't supported. If you extend a class, the subclass inherits all the methods and variables in the parent class (the *superclass*). Subclasses let you reuse existing code for a more specific purpose. Therefore, they help save you time during design, development, and testing. To customize the behavior of a superclass, override the methods in a subclass. A superclass is often known as a *base class*, and a subclass is often known as a *derived class*.

Subclass example

The following example first creates a class that is named **Point**. It then extends the **Point** class to create a new class that is named **ThreePoint**.

```
class Point
{
    // Instance fields.
    real x;
    real y;

    // Constructor to initialize fields x and y.
    void new(real _x, real _y)
    {
        x = _x;
        y = _y;
    }
}

class ThreePoint extends Point
{
    // Additional instance fields z. Fields x and y are inherited.
    real z;

    // Constructor is overridden to initialize z.
    void new(real _x, real _y, real _z)
    {
        // Initialize the fields.
        super(_x, _y);
        z = _z;
    }
}
```

Preventing class inheritance

You can prevent classes from being inherited by using the **final** modifier.

```
public final class Attribute
{
    int objectField;
}
```

Overriding a method

The methods in a class are inherited by any class that extends the class. To change the functionality of an inherited method, you create a method in the subclass, and then give that method the same name and parameters as the method in the superclass. This process is known as *overriding* the method. In the following example, **ColorAttribute** is a subclass of **Attribute** and therefore inherits the **methodAttr** method. However, because **ColorAttribute** defines a method that has the same name and the same number of arguments, the method in the superclass is overridden.

When you instantiate the subclass, you can assign the reference to either a variable of the superclass type or the subclass type. Regardless of the type of the variable, the overridden method is called.

In the following code example, the subclass overrides the **write** method. Two variables, both of type **Point** are created. One is assigned a **Point** object, the other is assigned a **ThreePoint** object. When the **write** method is called on the **ThreePoint** object, the **ThreePoint** version of the method is called.

```
class Point
{
    // Instance fields.
    real x;
    real y;

    // Constructor to initialize fields x and y.
    void new(real _x, real _y)
    {
        x = _x;
        y = _y;
    }

    void write()
    {
        info("(" + any2Str(x) + ", " + any2Str(y) + ")");
    }
}

class ThreePoint extends Point
{
    // Additional instance fields z. Fields x and y are inherited.
    real z;

    // Constructor is overridden to initialize z.
    void new(real _x, real _y, real _z)
    {
        // Initialize the fields.
        super(_x, _y);
        z = _z;
    }

    void write()
    {
        info("(" + any2Str(x) + ", " + any2Str(y) + ", " + any2Str(z) + ")");
    }

}

// Code that creates Point objects and calls the write method.
Point point2 = new Point(1.0, 2.0);
Point point3 = new ThreePoint(3.0, 4.0, 5.0);

point2.write();
// Output is "(1.0, 2.0)".

point3.write();
// Output is "(3.0, 4.0, 5.0)".
```

Preventing method overrides

Static methods can't be overridden, because they exist per class. To protect other sensitive methods, or core methods, from being overridden, use the **final** modifier. In the following example, because **methodAtt** is declared as **final**, it can't be overridden in any class that extends **Attribute**. You should not specify **new** or **finalize** methods as **final**.

The following example shows how to use the **final** keyword.

```
public class Attribute
{
    int objectVariable;

    final void methodAtt()
    {
        //Some statements
    }
}
```

Overriding vs. overloading

Overriding occurs when the superclass's implementation of a method is changed by the subclass's implementation of that method, but the signatures of both methods are the same.

By contrast, *overloading* occurs when more than one method has the same name, but the methods have different signatures (return types, parameter lists, or both). X++ supports overriding, but it doesn't support overloading.

X++ static classes

2/18/2021 • 3 minutes to read • Edit Online

This topic describes static class members in X++. In general, static methods are intended for these cases:

- The method has no reason to access the member variables that are declared in the class.
- The method has no reason to call any instance (non-static) methods of the class.

You declare static class members by using the **static** keyword. The **static** keyword instructs the system to create only one instance of the method, regardless of the number of instances of the class there are. This one instance is used throughout your session.

Static methods

This section describes a scenario where a software key type is used to help prevent piracy. Each instance of a software key can have its own unique value. Because all software keys must conform to the rules of software key design, the logic that tests for software key conformance is the same for all software keys. Therefore, the method that contains the conformance validation logic should be static.

Here is an example of a method that is declared by using the **static** keyword.

```
public class SoftwareKey
{
    static public boolean validateSoftwareKey(str _softwareKeyString)
    {
        // Your code here.
        return false;
    }
}
```

In the following example, you don't have to construct an instance of the **SoftwareKey** class before you call a static method on the class. When you want to call the static **validateSoftwareKey** method, the syntax starts with the name of the class that contains the method. A pair of colons (::) is used to connect the class name to the static method name.

```
boolean yourBool = SoftwareKey::validateSoftwareKey(yourSoftwareKeyString);
```

Static fields

Static fields are variables that are declared by using the **static** keyword. Conceptually, they apply to the class, not to instances of the class.

Static constructors

A static constructor is guaranteed to run before any static or instance calls are made to the class. The execution of the static constructor is relative to the user's session. The static constructor has the following syntax.

```
static void TypeNew()
```

You never explicitly call the static constructor. The compiler will generate code to make sure that the constructor is called exactly one time before any other method on the class. A static constructor is used to initialize any static data or perform a particular action that must be performed only one time. No parameters can be provided for

the static constructor, and it must be marked as **static**.

The following code example shows how to create a singleton instance by using a static constructor.

```
public class Singleton
{
    private static Singleton instance;

    private void new()
    {
    }

    static void TypeNew()
    {
        instance = new Singleton();
    }

    public static Singleton Instance()
    {
        return Singleton::instance;
    }
}
```

The singleton guarantees that only one instance of the class will ever be called. The following example shows how to instantiate the singleton.

```
Singleton i = Singleton::Instance();
```

Static methods

Static methods, which are also known as *class methods*, belong to a class and are created by using the keyword **static**. You don't have to instantiate an object before you use static methods. Static methods are often used to work with data that is stored in tables. Member variables can't be used in a static method. You use the following syntax to call static methods.

```
ClassName::methodName();
```

Static and instance methods

The accessor keywords on methods never restrict calls between two methods that are in the same class, regardless of which method is static or non-static. In a static method, calls to the **new** constructor method are valid even if the **new** constructor method is decorated with the **private** modifier. The syntax for these calls requires that the **new** keyword be used. The code in a static method must construct an instance object of its own class before it can call any instance methods on the class.

X++ interfaces

2/18/2021 • 2 minutes to read • Edit Online

An *interface* specifies a set of public instance methods. An interface defines and enforces similarities between unrelated classes without having to derive one class from the other.

All interfaces are public, even if you don't explicitly add the **public** keyword in front of the **interface** keyword in interface declaration. The methods on an interface are also public. Explicit inclusion of the keyword **public** is optional.

To create an interface, follow these steps.

1. In Server Explorer, right-click the project, and then click **Add**.
2. In the **New Item** dialog box, select **Interface**, and then enter a name for the interface.
3. Click **Add**.

When you add the **implements** keyword on a class declaration, the class must declare and define the methods that are specified by the interface. A class declaration can implement multiple interfaces. List the interfaces after the single occurrence of the **implements** keyword, and separate the interface names by using commas.

All interface methods that a class implements must be explicitly declared as **public**. A class that implements an interface must also be declared as **public**. An interface can extend another interface by using the **extends** keyword, however, an interface can't extend more than one interface.

It is customary to preface the name of an interface with `I`.

Interface example

In the following code example, the **Automobile** class implements the **IDrivable** interface. The **is** keyword tests whether a class implements an interface.

```
interface IDrivable
{
    int getSpeed()
    {
    }

    void setSpeed(int newSpeed)
    {
    }
}

class Automobile implements IDrivable
{
    int speed;

    public int getSpeed()
    {
        return speed;
    }

    public void setSpeed(int newSpeed)
    {
        speed = newSpeed;
    }
}

class UseAnAutomobile
{
    void DriveAutomobile()
    {
        IDrivable drivable;
        Automobile myAutomobile = new Automobile();
        str temp;

        myAutomobile = new Automobile();

        if (myAutomobile is IDrivable)
        {
            drivable = myAutomobile;
            drivable.setSpeed(42);
            temp = int2str(drivable.getSpeed());
        }
        else
        {
            temp = "Instance is not an IDrivable.";
        }

        info(temp);
    }
}
```

X++ class library

2/18/2021 • 8 minutes to read • Edit Online

This topic describes the library of classes in X++.

There are two kinds of classes: *application classes* and *system classes*.

- **Application classes** – These classes are implemented in X++. They are available in the **Code > Classes** node in Application Explorer.
- **System classes** – These classes are sometimes known as *kernel classes*. They are listed under the **System Documentation > Classes** node in Application Explorer. The source code for these classes isn't available. For a list of the system classes, see API, class, and table reference.

Typical structure of an application class

The following code block types are standard for application classes:

- **class and variable declarations**: The class declaration contains modifiers, such as **public**, **private**, and **extends**.
- **variable declarations**: These are the field members for objects that are constructed from the class. When you type the keyword **this** on a class instance variable, IntelliSense can show a list of the members.
- **new** method: This method creates an instance of the class. The constructor can be called only by using the **new** keyword. Derived classes can call the **new** method of their constructor by calling the **super** method reference. For more information, see X++ inheritance.
- **finalize** method: This method finalizes an instance of the class. This method is the destructor method. However, it's a destructor by convention only. The system doesn't automatically call the **finalize** method during garbage collection.

Additional methods for a class have the following types:

- Instance methods
- Static methods
- Main methods

Methods can be created on many kinds of items. Here are some examples:

- Classes
- Maps
- Views
- Data Sets
- Forms
- Queries

Substituting application classes for system classes

You should use the *substitute application classes* instead of the system classes that they extend.

In Application Explorer, under **System Documentation > Classes**, several kernel or system classes have names that begin with a lowercase *x*. These classes are known as *x-system classes*. Examples of these system classes are **xApplication** and **xVersionControl**. Some of these classes are extended by application classes. For example, the **Application** class extends the **xApplication** system class.

The classes that derive from x-system classes are known as *substitute application classes*. In Application Explorer, under the **Classes** node, the icon next to the substitute application classes differs from the standard icon.

x-system classes

Some of the substitute application classes are associated with a special global variable that represents an instance of the class. For example, the **appl** variable references a pre-instantiated object from the **Application** class. The advantage of the **appl** variable is that the system maintains the object throughout the scope of your session. Your code would be less efficient if it repeatedly used the **new Application()** syntax to obtain an instance of the **Application** class. You should not use the **xApplication** system class. Instead, use the **Application** substitute application class.

You can reference the static members of the **Application** class by using the following standard syntax: **Application::checkForNewBatchJobs()**. However, to reference the instance members of the **Application** class, you should use that class's **appl** variable, if it exists. This pattern applies to most of the x-system classes. The **Session** substitute application class is one exception, because there is no special global variable for **Session**.

The following table lists the x-system classes that have a corresponding substitute application class. The special global variables are also shown for those classes that have one.

APPLICATION CLASS	X-SYSTEM CLASS	GLOBAL VARIABLE
Args	xArgs	Not applicable
Application	xApplication	**appl**
ClassFactory	xClassFactory	**classFactory**
Company	xCompany	**appl.company**
Global	xGlobal	Not applicable
Info	xInfo	**Infolog**
MenuFunction	xMenuFunction	Not applicable
Session	xSession	Not applicable
VersionControl	xVersionControl	**versionControl**

Example of x-system classes

The following example shows the syntax for using several special variables that reference instances of the substitute application classes.

```
TreeNode treeNode;
Args     args;
FormRun  formRun;

// appl variable
info(appl.buildNo());

// company variable
appl.company().reloadRights();

// infolog variable
treeNode = infolog.findNode("\\forms\\custTable");
info(treeNode.AOTgetProperty("Name"));
// Output is "CustTable".

// classFactory variable
args = new Args(formstr(Batch));
formRun = classFactory.formRunClass(args);
formRun.init();
formRun.run();
formRun.detach();
info("Method is ending. This is a message in the Infolog.");
// Output is "Method is ending. This is a message in the Infolog."
```

Batch processing classes

You implement classes by using the batch processing system, and by extending the **RunBase** and **RunBaseBatch** classes. To remove the **Recurrence** button from the **Batch processing** dialog box, you use the **Args::parmEnum** method. We recommend that you designate a class to run as a server-bound batch method. Server-bound batch methods are more secure than batch methods that aren't server-bound for the following reasons:

- The method is run by using the permissions of the user who submitted the method.
- The method can use only specific **Info** and **Global** class methods to interact with the client that is processing it. This restriction limits interaction with the client.

Enable a class to run as a server-bound batch method

1. Create a class that extends the **RunBaseBatch** class.
2. Override the **RunBaseBatch.runsImpersonated** method to return a value of **true**, as shown in the following example.

   ```
   public boolean runsImpersonated()
   {
       return true;
   }
   ```

3. Confirm that the class calls only the following **Info** and **Global** class methods:

   - add
   - Info.copy
   - Info.cut
   - Info.import
   - Info.export
   - Info.line
   - Info.num
   - Global::error

- Global::info
- Global::warning

The **Info.line** and **Info.num** methods are inherited from the **xInfo** class.

Removing the Recurrence button from the batch processing dialog box

When you implement a class by using the batch processing system, you can remove the **Recurrence** button by calling the **Args.parmEnum** method and passing the **NoYes::Yes** system enumeration value. The **NoYes** system enumeration determines whether the **Recurrence** button is removed from the dialog box. The default value is **NoYes::No**.

In the following example, the **InventTransferMultiShip** class is implemented. The **BatchDialog::main** method creates the **Batch processing** dialog box.

```
static void noRecurrenceButton(Args _args)
{
    Args a;
    InventTransferMultiShip inventTransferMultiShip;
    a = new Args();
    inventTransferMultiShip = InventTransferMultiShip::construct();
    a.caller(inventTransferMultiShip);
    a.parmEnum(NoYes::Yes);
    BatchDialog::main(a);
}
```

Image manipulation classes

Two system classes let you to manipulate graphics and icons: **Image** and **Imagelist**.

- **Image** – This class lets you load, save, and manipulate individual images. For example, you can capture a screen and save it as an image, crop or rotate an image, or manipulate the color depth.
- **Imagelist** – This class lets you work with a set of images that have common properties, such as the size and transparency color. You can view the image lists that are used in the **ImageListAppl** application classes.

Query object model

The query object model contains classes that are used to define and run a query. The query objects are used to define the query data source, the fields that are returned, record ranges, and relations to child data sources. The query classes are more visible when you create a dynamic query in code, but they are also used behind the scenes when you create a static query in Application Explorer.

The following table describes the classes in the query object model.

SYSTEM CLASS	DESCRIPTION
QueryRun	This class runs the query and fetches the data.
Query	This class holds some properties, and has one or more related data sources. It's the top level of the query definition.
QueryBuildDataSource	This class defines access to a single data source in the query. If there is more than one data source at the same level in a query, separate SQL statements are produced and are run sequentially. If one data source is a child of another data source, a join is created between the two data sources.

SYSTEM CLASS	DESCRIPTION
QueryBuildFieldList	This class defines the fields that are returned from the database. By default, the field list is dynamic, and all fields are returned from the data source table, map, or view. Each data source has only one **QueryBuildFieldList** object. This object contains information about all selected fields. You can specify aggregate functions, such as **SUM**, **COUNT**, and **AVG**, on the field list object.
QueryBuildRange	This class defines a subset of records that is returned, based on a single field. A range is translated into a **WHERE** clause in the query SQL statement. If more than one field is used to limit the query (**WHERE** clause), the data source will contain more than one range.
QueryBuildDynalink	This class contains information about a relation (limitation) to an external record. When the query is run, this information is converted to additional entries in the **WHERE** clause of the query SQL statement. This class can exist only on the parent data source of a query. Forms use the function when two data sources are synchronized. The child data source will then contain one or more DLLs to the parent data source. The function is used even if the two data sources are put in two different forms but are still synchronized.
QueryBuildLink	This class specifies the relation between the two data sources in the join. This class can exist only on a child data source.

You can also use the SysDa API to query data.

System classes overview

The source for system classes isn't available. A system class can have the following characteristics:

- Static methods (or class methods)
- Dynamic methods
- Properties – These properties are member functions that are used to set properties. An example is **LeftMargin**.

You can't override system class methods. It isn't our intention that you will use the system classes to design your application objects from scratch. Instead, use them to extend or modify the default functionality in Application Explorer. For example, you can dynamically add extra information to an existing report. Alternatively, you can change the options that are available on a page, based on the user's selection on a previous page.

Collection classes

The *collection classes* let you create lists, sets, structs, maps, and arrays.

Application object classes

These system classes hold functions that are activated whenever you use Application Explorer to create your application. For example, the system uses the **FormDesign** class when you define the layout of your form in the **Designs** node in Application Explorer. These classes also let you to create and modify application objects.

Integration classes

The integration with the environment is typically implemented by classes. Here are some examples of the classes in this category:

- **COM** – The call of methods on COM objects.
- **DLL** – The call of Microsoft Windows DLL functions.
- **IO** – Read and write external files.
- **ODBCConnection** – An Open Database Connectivity (ODBC) interface to a foreign database.

X++ event terminology and keywords

2/18/2021 • 2 minutes to read • Edit Online

This topic describes event terminology and keywords in X++.

You can use the event design pattern to make your code more modular and reusable. The term *event* is a metaphor that explains how delegates are used. When something important occurs during a program run, other modules might have to process the occurrence. These important occurrences are known as *events*. When an event occurs, the program tells its notifier for the event that the notifier must send notifications about the event. A notification must be sent to all the event handlers that are subscribers of the notifier. When the program tells its notifier to send the notifications, we call that process *raising* an event.

The following table shows the terms that are used to describe the event metaphor.

TERM	DESCRIPTION
Event	An important occurrence in a program module where additional modules must process the occurrence.
Notifier	The program element that sends information about the event to all the event handlers that are subscribed to the notifier.
Subscriber	The program functions or methods that are subscribed to an event notifier.
Event handler	The methods that subscribe to an event notifier. Only the appropriate kind of methods can be event handlers.

Keywords that are used for programming that uses delegates

The following table shows the keywords that describe the use of delegates.

KEYWORD OR TERM	CODE	DESCRIPTION
delegate	`delegate myDelegate(str information) {}`	The code shows what the delegate looks like in the code editor. Because the return type is always **void**, it isn't mentioned in the syntax. No code is allowed inside the braces ({}).
eventHandler	`myClassInstance.myDelegate += eventHandler(otherClass.myInstanceMethod);`	Although the syntax of the **eventHandler** keyword might give the impression that **eventHandler** is an X++ function, it isn't a function. The **eventHandler** keyword tells the compiler that a method is being subscribed to a delegate.
Subscribe or add a method to a delegate	`myClassInstance.myDelegate += eventHandler(OtherClass::aStaticMethod);`	In the code, the static method **OtherClass::aStaticMethod** becomes subscribed to the delegate.

KEYWORD OR TERM	CODE	DESCRIPTION
Call a delegate	`myClassInstance.myDelegate("Hello");`	This call to the delegate prompts the delegate to call each method that is subscribed to the delegate. The subscribed methods are called in the same order in which they were added to the delegate. One subscribed method must be completed before the delegate calls the next method.

Example

The two classes in the following code example demonstrate how to define an event, subscribe to an event, and raise an event. The **PointWithEvent** class defines a delegate, **moved**. The **move** method calls the **moved** delegate, thereby notifing any objects that have subscribed to the event. The **PointKeeper** class defines the **writeMove** method and assigns it as the event handler for the **moved** delegate of the **Point** instance created in the **createAndMove** method.

```
class PointWithEvent
{
    // Instance fields.
    real x;
    real y;

    // Constructor to initialize fields x and y.
    void new(real _x, real _y)
    {
        x = _x;
        y = _y;
    }

    void move(real x_offset, real y_offset)
    {
        x += x_offset;
        y += y_offset;
        this.moved(abs(x_offset) + abs(y_offset));
    }

    delegate void moved(real distance)
    {
    }

}

class PointKeeper
{

    public void createAndMove()
    {
        PointWithEvent point = new PointWithEvent(1.0, 2.0);

        point.moved += eventhandler(this.writeMove);

        point.move(4.0, 5.0);
        // Output is "9.0".
    }

    public void writeMove(real distance)
    {
        info(any2Str(distance));
    }

}
```

X++ data selection and manipulation overview

2/18/2021 • 2 minutes to read • Edit Online

You can use SQL statements, either interactively or in source code, to retrieve and modify data that is stored in the database. You can use the **select** statement and API methods for these tasks:

- Select data: Select the data to view or modify.
  - **select statement** – Fetch records.
- Insert data: Add one or more new records to a table.
  - **insert and doInsert methods** – Insert one record at a time.
  - **insert_recordset, RecordInsertList.insertDatabase, and RecordSortedList.insertDatabase methods** – Insert multiple records at the same time.
- Update data: Modify the data in existing table records.
  - **update and doUpdate methods** – Update one record at a time.
  - **update_recordset statement** – Update multiple records at the same time.
- Delete data: Remove existing records from a table.
  - **delete and doDelete methods** – Delete one record at a time.
  - **delete_from statement** – Delete multiple records at the same time.

Here are some other statements that you will use in data access:

- while select statement
- select expression
- next statement

Transactional integrity helps prevent data corruption and improve scalability.

The Conversion of operations from set-based to record-by-record topic provides information about how you can use the record set–based statements and methods more efficiently.

You can also use the SysDa classes to retrieve and modify data. The extensible SysDa API provides almost all the data access possibilities that are available in X++.

The **executeQueryWithParameters** API can help mitigate a SQL injection attack.

Select statement

2/18/2021 • 18 minutes to read • Edit Online

The **select** statement fetches or manipulates data from the database.

- All **select** statements use a table variable to fetch records. This variable must be declared before a **select** statement can be run.
- The **select** statement fetches only one record, or field. To fetch or traverse multiple records, you can use the **next** statement or the **while select** statement.
  - The **next** statement fetches the next record in the table. If no **select** statement precedes the **next** statement, an error occurs. If you use a **next** statement, don't use the **firstOnly** find option.
  - It's more appropriate to use a **while select** statement to traverse multiple records.
- The results of a **select** statement are returned in a table buffer variable.
- If you use a field list in the **select** statement, only those fields are available in the table variable.

Select example

The following example fetches all the columns in the first row of the CustTable table and prints the value in the **AccountNum** column of that row.

```
CustTable custTable;
select * from custTable;
info("AccountNum: " + custTable.AccountNum);
```

For more examples of data selection, see Select data.

Insert example

The following example inserts a new record into the CustTable table. The **AccountNum** column of the new record is set to **2000**, and the **CustGroup** column is set to **1**. Other fields in the record will be blank.

```
ttsBegin;
    CustTable custTable;
    select forUpdate custTable;
    custTable.AccountNum = '2000';
    custTable.CustGroup = '1';
    custTable.insert();
ttsCommit;
```

For more examples of data insertion, see Insert data.

Update example

The following example selects the CustTable table for update. Only records where the value of the **AccountNum** field equals **2000** are updated. Because there is no call to **next**, and this example doesn't use a **select while** statement, only one record is updated. The value of the **CreditMax** field is changed to **5000**.

```
ttsBegin;
    CustTable custTable;
    select forUpdate custTable
        where custTable.AccountNum == '2000';
    custTable.CreditMax = 5000;
    custTable.update();
ttsCommit;
```

For more examples of data updates, see Update data.

Delete example

In the following example, all records in the CustTable table where the **AccountNum** field equals **2000** are deleted from the database. One record is deleted at a time.

```
ttsBegin;
    CustTable custTable;
    while select forUpdate CustTable
        where custTable.AccountNum == '2000'
    {
        custTable.delete();
    }
ttsCommit;
```

For more examples of data deletion, see Delete data.

Syntax of the select statement

The following symbols are used in the syntax:

- **[]** – Brackets enclose an optional element.
- **{}** – Braces enclose an element that can be included zero or more times.
- **+** – A plus sign indicates an element that can be included one or more times.
- **|** – A bar indicates options.

SYMBOL		EXPRESSION
SelectStatement	=	**select** *Parameters*
Parameters	=	{ *FindOption* } [*FieldList* **from**] *TableBufferVariable* [*IndexClause*] [*Options*] [*WhereClause*] [*JoinClause*]
FindOption	=	**crossCompany** [: *ContainerVariable*] | **reverse** | **firstFast** | *FirstOption* | **forUpdate** | **noFetch** | *ForceOption* | **forceSelectOrder** | **forceNestedLoop** | *LockOption* | **repeatableRead** | **validTimeState**
FirstOption	=	**firstOnly** | **firstOnly10** | **firstOnly100** | **firstOnly1000**
LockOption	=	**optimisticLock** | **pessimisticLock**

SYMBOL		EXPRESSION
ForceOption	=	**forcePlaceholders** | **forceLiterals**
FieldList	=	{ *Field* } | *****
Field	=	*Aggregate* **(** *FieldIdentifier* **)** | *FieldIdentifier*
Aggregate	=	**sum** | **avg** | **minof** | **maxof** | **count**
Options	=	*OrderClause* | *IndexClause*
OrderClause	=	[*OrderBy* [*GroupBy*]] | [*GroupBy* [*OrderBy*]]
OrderBy	=	**order** [**by**] *FieldOrder* {, *FieldOrder* }
GroupBy	=	**group** [**by**] *FieldOrder* {, *FieldOrder* }
FieldOrder	=	*FieldIdentifier* [**asc** | **desc**]
IndexClause	=	**index** *IndexName* | **index hint** *IndexName*
WhereClause	=	**where** *Expression InClause*
InClause	=	**in** *List*
JoinClause	=	[**exists** | **notexists** | **outer**] **join** *Parameters*
ContainerVariable	=	A container.
Expression	=	An expression.
TableBufferVariable	=	The variable name for the results.
FieldIdentifier	=	The name of a field in the table.
IndexName	=	The name of an index for a table.
List	=	An array of values.

Aggregate functions

The aggregate functions perform calculations on a single field over a group of records.

- The result is returned in the field that you perform the aggregate function over.
- The fields in the results are the aggregate values and the fields in the **group by** clause.
- You can count, average, or sum only integer and real fields.
- In cases where the **sum** function will return **null**, no rows are returned.

Differences between X++ and SQL

In industry-standard SQL, a database query can contain aggregate functions. Examples include **count(RecID)** and **sum(columnA)**. When an aggregate function is used, but no rows match the **where** clause, a row must be returned to hold the result of the aggregates. The row that is returned shows the value **0** (zero) for the **count** function and **null** for the **sum** function. X++ doesn't support the concept of **null** values for the database. Therefore, in cases where the **sum** function will return **null**, no row is returned to the user. Additionally, every data type has a specific value that is treated as a **null** value in some circumstances.

Grouping and ordering the query results

A query can have multiple **group by** clauses, but the fields can be qualified by a table name in only one **group by** clause. We recommend that you use table name qualifiers. The **order by** clause follows the same syntax patterns as **group by**. Both clauses, if they are provided, must appear after the **join** (or **from**) clause, and both must appear before any **where** clause that exists on the same **join** clause. We recommend that all **group by**, **order by**, and **where** clauses appear immediately after the last **join** clause. The following example shows a **group by** clause where a field is qualified by a table name.

```
CustTable custTable;
CustGroup custGroup;
struct groupSummary = new struct("int CustomerCount; str CustGroup");

while select count(CreditMax) from custTable
    join custGroup
    order by custGroup.Name
    group by custGroup.CustGroup
    where custTable.CustGroup == custGroup.CustGroup
        && custGroup.Name like "*Days*"
{

    groupSummary.value("CustomerCount", custTable.CreditMax);
    groupSummary.value("CustGroup", custGroup.CustGroup);
    info(groupSummary.toString());
}

// Example output:
// (CustomerCount:1; CustGroup:"1")
// (CustomerCount:3; CustGroup:"2")
```

Join tables

The following example shows how an inner join can be performed as part of a **select** statement. The example also shows an **order by** clause where each field is qualified by a table name. Therefore, you can use just one **order by** clause to control how the retrieved records are sorted.

```
CustTable custTable;
CustGroup custGroup;
struct output = new struct("int AccountNum; str CustGroup");

while select AccountNum from custTable
    join Name from custGroup
    order by custGroup.Name, custTable.AccountNum
    where custTable.CustGroup == custGroup.CustGroup
{
    info(custGroup.Name + ': ' + custTable.AccountNum);
}

// Example output:
// Days1: 6000
// Days1: 6001
// Days2: 5000
```

Using where, order by, and index hint together in a query

You use the **order by** keyword in **select** statements to order the data that is returned. Use the **index hint** keyword to specify the index that should be used in the query, and to sort the selected records in the manner that is defined by the index. Indexes optimize the selection of records. To select records in a specific order, combine the **index hint** keyword with an **order by** expression. If you want the output to be sorted in reverse order, use the **reverse** keyword. If a table index has been disabled (that is, if the index's **Enabled** property is set to **No**), the **select** statement that references the index is still valid. However, the database can't use the index as a hint to sort the data, because the index doesn't exist in the database. The following table shows how to use the **index hint** and **order by** keywords in **select** statements.

TASK	SELECT STATEMENT
Select records when the order isn't significant.	select .. where ...
Select records when the order is significant.	select .. order by ... where ...
Select records, and force a specific index to be used.	select .. index hint ... where ...
Select records when the order is significant, and force a specific index to be used.	select .. index hint ... order by ... where ...

The following example shows how to select transactions from the SalesTable table, based on a range of customers and due dates.

```
SalesTable salesTable;
select salesTable
    index hint CustIdx
    order by CustAccount
    where
        salesTable.CustAccount >= '3000'
        && salesTable.CustAccount <= '4000'
        && salesTable.FixedDueDate >= 12\12\2004
        && salesTable.FixedDueDate <= 05\05\2009;
```

asc keyword

The **asc** keyword is an option on the **order by** or **group by** clause. It specifies an ascending sort order. If neither **asc** nor **desc** is specified, the sort is ascending.

```
CustTable custTable;
select * from custTable
    order by Value asc;
```

avg keyword

The **avg** keyword returns the average of the fields.

```
CustTable custTable;
select avg(Value) from custTable;
info(int642Str(custTable.Value));
```

count keyword

The **count** keyword returns the number of records.

```
CustTable custTable;
int64 iCountRows;
select count(RecID) from custTable;
iCountRows = custTable.RecID;
info('Rows: ' + int642Str(iCountRows));
```

crossCompany keyword

The **crossCompany** keyword returns data for all companies that the user is authorized to read from. You can add a container to reduce the number of companies that are involved. The following example returns data for companies that the user is authorized to read from. Results are limited to the **dat** and **dmo** companies.

```
CustTable custTable;
container conCompanies = ['dat','dmo'];
select crossCompany :conCompanies
    * from custTable;
```

desc keyword

The **desc** keyword is an option on the **order by** or **group by** clause. It specifies a descending sort order. If neither **asc** nor **desc** is specified, the sort is ascending.

```
CustTable custTable;
select * from custTable
    order by Value desc;
```

exists keyword

The **exists** keyword is a method that returns a Boolean value and a **join** clause.

```
CtrTable ctrTable;
CustTable custTable;
while select AccountNum, Value from custTable
    order by AccountNum
    exists join * from ctrTable
    where (ctrTable.AccountNum == custTable.AccountNum)
{
}
```

firstFast keyword

The **firstFast** keyword is a priority hint. The first row appears more quickly, but the total return time for this option might be slower. The **firstFast** hint is automatically issued from all pages.

The following code example quickly returns the first row.

```
CustTable custTable;
select firstFast custTable
    order by AccountNum;
```

firstOnly, firstOnly10, firstOnly100, and firstOnly1000 keywords

The **firstOnly** keywords speed up the fetch by returning a limited number of rows. When you include **firstOnly** in your query, the runtime returns a table buffer. When you omit **firstOnly**, the runtime allocates an object that can iterate over records. From a performance perspective, you should use **firstOnly** only when your intent is to fetch the first record.

KEYWORD	DESCRIPTION
firstOnly	Return only the first row.
firstOnly10	Return 10 rows.
firstOnly100	Return 100 rows.
firstOnly1000	Return 1,000 rows.

The following code example returns only the first row of the results.

```
CustTable custTable;
select firstOnly custTable
    index hint AccountIdx
    where custTable.AccountNum == '5000';
```

forceLiterals keyword

The **forceLiterals** keyword instructs the kernel to reveal the actual values that are used in **where** clauses to the Microsoft SQL Server database at the time of optimization. The **forceLiterals** and **forcePlaceholders** keywords are mutually exclusive. For more information, see the forcePlaceholders keyword section.

WARNING

You should not use the **forceLiterals** keyword in **select** statements, because it could expose code to an SQL injection security threat.

forceNestedLoop keyword

The **forceNestedLoop** keyword forces the SQL Server database to use a nested-loop algorithm to process a particular SQL statement that contains a join algorithm. Therefore, a record from the first table is fetched before any records from the second table are fetched. Typically, other join algorithms, such as hash joins and merge joins, are considered. This keyword is often combined with the **forceSelectOrder** keyword.

```
CustGroup custGroup;
CustTable custTable;

while select forceNestedLoop custGroup
    join custTable
    where custGroup.CustGroup == custTable.CustGroup
{
    Info(custTable.CustGroup);
}
```

forcePlaceholders keyword

The **forcePlaceholders** keyword instructs the kernel **not** to reveal the actual values that are used in **where** clauses to the SQL Server database at the time of optimization. By default, this behavior is used in all statements that aren't **join** statements. The advantage of using this keyword is that the kernel can reuse the access plan for similar statements that have other search values. The disadvantage is that, although the access plan is computed, the fact that data distribution might be uneven isn't considered. The access plan is an on-average access plan. The **forcePlaceholders** and **forceLiterals** keywords are mutually exclusive.

The following example iterates through the **CustGroup** table that is joined with the **CustTable** table.

```
CustGroup custGroup;
CustTable custTable;

while select forcePlaceholders custGroup
    join custTable
    where custGroup.CustGroup == custTable.CustGroup
{
    Info(custTable.CustGroup);
}
```

forceSelectOrder keyword

The **forceSelectOrder** keyword forces the SQL Server database to access the tables in a join in the specified order. If two tables are joined, the first table in the statement is always accessed first. This keyword is often combined with the **forceNestedLoop** keyword.

The following example joins the CustGroup and CustTable tables on the **CustGroup** field.

```
CustGroup custGroup;
CustTable custTable;

while select forceSelectOrder custGroup
    join custTable
    where custGroup.CustGroup == custTable.CustGroup
{
    Info(custTable.CustGroup);
}
```

forUpdate keyword

The **forUpdate** keyword selects records for update only. Depending on the underlying database, the records might be locked for other users. The following example selects the **CreditMax** column in the CustTable table for update, for the record where the **AccountNum** value is **2000**.

```
ttsBegin;
    CustTable custTable;
    select forUpdate custTable
        where custTable.AccountNum == '2000';
    custTable.CreditMax = 5000;
    custTable.update();
ttsCommit;
```

group by keyword

The **group by** keyword instructs the database to group selected records by fields.

```
CustTable custTable;
while select sum(CreditMax) from custTable
    group by CustGroup
{
    info(custTable.CustGroup + ' ' + int642Str(custTable.CreditMax));
}
```

in keyword

The **in** keyword filters rows where a value is contained in a list.

If you don't use the **in** keyword, the code that you write will resemble the following example.

```
// This code doesn't use the in keyword.
private CostAmountStdAdjustment calcCostAmountStdAdjustment()
{
    InventSettlement inventSettlement;

    select sum(CostAmountAdjustment) from inventSettlement
        where inventSettlement.TransRecId == this.RecId
            && inventSettlement.Cancelled == NoYes::No
            && (inventSettlement.OperationsPosting == LedgerpostingType::purchStdProfit
                || inventSettlement.OperationsPosting == LedgerpostingType::purchStdLoss
                || inventSettlement.OperationsPosting == LedgerpostingType::InventStdProfit
                || inventSettlement.OperationsPosting == LedgerpostingType::InventStdLoss);

    return inventSettlement.CostAmountAdjustment;
}
```

If you use the **in** keyword, the code that you write will resemble the following example.

```
// This code uses the in keyword.
private CostAmountStdAdjustment calcCostAmountStdAdjustment()
{
    InventSettlement inventSettlement;
    container ledgerPostingTypes = this.ledgerPostingTypesForCostAmountStdAdjustmentCalculation();

    select sum(CostAmountAdjustment) from inventSettlement
        where inventSettlement.TransRecId == this.RecId
            && inventSettlement.Cancelled == NoYes::No
            && inventSettlement.OperationsPosting in ledgerPostingTypes;

    return inventSettlement.CostAmountAdjustment;
}

protected container ledgerPostingTypesForCostAmountStdAdjustmentCalculation()
{
return [
    LedgerPostingType::purchStdProfit,
    LedgerPostingType::PurchStdLoss,
    LedgerPostingType::InventStdProfit,
    LedgerPostingType::InventStdLoss
];
}
```

index keyword

The **index** keyword instructs the database to sort the selected records as specified by the index.

```
CustTable custTable;
while select AccountNum, Value from custTable
    index AccountIdx
{
    Info(custTable.AccountNum +  ": " + int642Str(custTable.Value));
}
```

index hint keyword

The **index hint** keyword gives the database a hint to use a specific index to sort the selected records as specified in the index. The database can ignore the hint. An incorrect index hint can greatly affect performance. Index hints should be applied only to SQL statements that don't have dynamic **where** clauses or **order by** clauses, and where the effect of the hint can be verified.

Before you can use **index hint** in queries, you must call **allowIndexHint(true)** on the table. The default behavior for **index hint** is **false**, and the hint is ignored.

> **WARNING**
> You should use **index hint** sparingly and with caution, and only when you can be sure that it improves performance. The **index hint** keyword and API let you pass the correct hints when they are required. If you're ever in doubt, avoid using **index hint**.

In the following example, the **AccountIdx** index is used to sort the records in the query on the CustTable table.

```
str _accountNum = '111';
CustTable custTable;
custTable.allowIndexHint(true);

while select forUpdate custTable
    index hint AccountIdx
    where custTable.AccountNum == _accountNum
{
}
```

join keyword

The **join** keyword is used to join tables on a column that is shared by both tables. The join criteria are specified in a **where** clause, because there is no **on** keyword, such as is found in SQL. This keyword reduces the number of SQL statements that are required if you want to loop through a table and update transactions in a related table. For example, you process 500 records in a table and want to update related records in another table. If you use a nested **while select**, there will be 501 trips to the database. However, if you use a **join**, there will be just one trip to the database.

```
CustTable custTable;
CustGroup custGroup;
int totalCredit;

while select custGroup
    join custTable
    where custGroup.CustGroup == custTable.CustGroup
{
    totalCredit += custTable.CreditMax;
}
}
```

maxof keyword

The **maxof** keyword returns the maximum of the fields.

```
CustTable custTable;
select maxof(CreditMax) from custTable;
info(int642Str(custTable.Value));
```

minof keyword

The **minof** keyword returns the minimum of the fields.

```
CustTable custTable;
select minof(CreditMax) from custTable;
info(int642Str(custTable.Value));
```

noFetch keyword

The **noFetch** keyword indicates that no records should be fetched now. Typically, this keyword is used when the result of the **select** statement is passed on to another application object, such as a query that performs the actual fetch.

The following example creates a query variable but doesn't fetch the records.

```
CustTable custTable;
select noFetch custTable
    order by AccountNum;
```

notExists keyword

The **notExists** keyword is selected only if there are no posts.

```
CustTable custTable;
CtrTable ctrTable;

while select AccountNum, Value from custTable
    order by AccountNum
    notExists join * from ctrTable
    where (ctrTable.AccountNum == custTable.AccountNum)
{
}
```

optimisticLock keyword

The **optimisticLock** keyword forces a statement to run by using optimistic concurrency control, even if a different value is set on the table.

```
CustTable custTable;
select optimisticLock custTable
    where custTable.AccountNum > '1000';
```

order by keyword

The **order by** keyword instructs the database to sort the selected records by the fields in the **order by** list. The keyword **by** is optional.

```
CustTable custTable;
select * from custTable
    order by accountNum desc
    where custTable.AccountNum > '100';
info("AccountNum: " + custTable.AccountNum);
```

The following example prints the highest **AccountNum** value in the CustTable table.

```
CustTable custTable;
select reverse custTable
    order by accountNum;
info("AccountNum: " + custTable.AccountNum);
```

outer keyword

The **outer** keyword returns all rows from the table that is named first, even rows that have no match in the table that is named second. This join is a left outer join. Default values are substituted for any data that could not be obtained from a matching row in the other joined table.

There is no **left** keyword, and there is no right outer join.

For an inner join, the behavior if you filter on an **on** clause is the same as the behavior if you filter on the **where**

clause.

```
CustTable custTable;
CustGroup custGroupTable;

while select CustGroup from custGroupTable
    order by CustGroup
    outer join * from custGroupTable
    where custTable.CustGroup == custGroupTable.CustGroup
{
    Info(custTable.CustGroup + ', ' + custGroupTable.Name);
}
```

The following example is based on two tables. The field types and example data are included. There is a one-to-many relationship between the SalesOrder parent table and the SalesOrderLine child table. For each row in the SalesOrder table, there are 0 (zero) or more rows in the SalesOrderLine table. There are two rows in the SalesOrder table.

SALESORDERID (INTEGER, PRIMARY KEY)	DATEADDED (DATE)
1	2010-01-01
2	2010-02-02

The SalesOrderLine table contains a foreign key field that is named **SalesOrderID**. This field references the primary key column of the SalesOrder table. A **SalesOrderID** value of **2** doesn't occur in the data for the SalesOrderLine table.

SALESORDERLINEID (STRING, PRIMARY KEY)	QUANTITY (INTEGER)	SALESORDERID (INTEGER, FOREIGN KEY)
AA	32	1
BB	67	1
CC	66	1

The following code has a **select** statement that reads the two tables. The **select** statement includes a left **outer join** clause. Both the join criteria and the data filter are on the **where** clause. The output from the code is also shown. The second record in the output has a **SalesOrderID** value of **2**. However, that value isn't present in the SalesOrderLine table. Therefore, some of the fields in the second record have default values: **0** for an integer and a zero-length string for a string.

```
SalesOrder recSalesOrder;
SalesOrderLine recSalesOrderLine;
struct struct4 = new struct
    ("int SalesOrderID;"
    + "date DateAdded;"
    + "str SalesOrderLineID;"
    + "int Quantity"
    );
while select *
    from
        recSalesOrder
        outer join recSalesOrderLine
    where
        recSalesOrder.SalesOrderID == recSalesOrderLine.SalesOrderID
        && recSalesOrderLine.Quantity == 66
{
    struct4.value("SalesOrderID", recSalesOrder.SalesOrderID);
    struct4.value("DateAdded", recSalesOrder.DateAdded);
    struct4.value("SalesOrderLineID", recSalesOrderLine.SalesOrderLineID);
    struct4.value("Quantity", recSalesOrderLine.Quantity);
    info(struct4.toString());
}

// Example output:
// (SalesOrderID:1; DateAdded:2010/1/1; SalesOrderLineID:"CC"; Quantity:66)
// (SalesOrderID:2; DateAdded:2010/2/2; SalesOrderLineID:""; Quantity:0)
```

pessimisticLock keyword

The **pessimisticLock** keyword forces a statement to run by using pessimistic concurrency control, even if a different value is set on the table.

```
CustTable custTable;
select pessimisticLock custTable
    where custTable.AccountNum > '1000';
```

repeatableRead keyword

This **repeatableRead** keyword specifies that the current transaction must be completed before other transactions can modify data that has been read by logic inside the current transaction. An explicit transaction is completed at either **ttsAbort** or the outermost **ttsCommit**. For a standalone **select** statement, the transaction duration is the duration of the **select** command. However, the database sometimes enforces the equivalent of **repeatableRead** in individual **select** statements, even if this keyword doesn't appear in your code. (The behavior depends on the method that the database uses to determine whether it should scan the tables.) For more information, see the documentation for the underlying relational database product.

reverse keyword

The **reverse** keyword returns records in reverse order.

```
CustTable custTable;
select reverse custTable
    order by AccountNum;
```

sum keyword

The **sum** keyword returns the sum of the fields. It can be used to sum all accounts, order lines, and so on.

```
CustTable custTable;
select sum(CreditMax) from custTable;
info(int642Str(custTable.Value));
```

validTimeState keyword

The **validTimeState** keyword selects rows from a table where the **ValidTimeStateFieldType** property is set to a value other than **None**.

```
CustPackingSlipTransHistory history;
utcDateTime dateFrom, dateTo = DateTimeUtil::utcNow();
anytype recid = -1;
select
    validTimeState(dateFrom, dateTo)
    *
    from history;
recid = history.RecId;
info('RecId:' + int642Str(recid));
```

where keyword

The **where** keyword filters rows from a table where the expression is **true**.

The following example finds a customer that has an **AccountNum** value that is more than 100.

```
CustTable custTable;
select * from custTable
    where custTable.AccountNum > '100';
info("AccountNum: " + custTable.AccountNum);
```

The following examples prints the lowest **AccountNum** value that is more than 100.

```
CustTable custTable;
select * from custTable
    order by accountNum
    where custTable.AccountNum > '100';
info("AccountNum: " + custTable.AccountNum);
```

The following example prints the highest **AccountNum** value that is more than 100.

```
CustTable custTable;
select * from custTable
    order by accountNum desc
    where custTable.accountNum > "100";
info("AccountNum: " + custTable.AccountNum);
```

Select data

2/18/2021 • 2 minutes to read • Edit Online

The **select** statement fetches or manipulates data from the database.

- All **select** statements use a table variable to fetch records. This variable must be declared before a **select** statement can be run.
- The **select** statement fetches only one record, or field. To fetch or traverse multiple records, you can use the **next** statement or the **while select** statement.
  - The **next** statement fetches the next record in the table. If no **select** statement precedes the **next** statement, an error occurs. If you use a **next** statement, don't use the **firstOnly** find option.
  - It's more appropriate to use a **while select** statement to traverse multiple records.
- The results of a **select** statement are returned in a table buffer variable.
- If you use a field list in the **select** statement, only those fields are available in the table variable.

The following example fetches all the columns in the first row of the CustTable table and prints the value in the **AccountNum** column of that row.

```
CustTable custTable;
select * from custTable;
info("AccountNum: " + custTable.AccountNum);
```

The following example prints the value in the **AccountNum** column of each row in the CustTable table.

```
CustTable custTable;
while select * from custTable
{
    info("AccountNum: " + custTable.AccountNum);
}
```

The following example prints the value in the **AccountNum** column of the first two rows that are returned by the **select** statement.

```
CustTable custTable;
select * from custTable;
info("AccountNum: " + custTable.AccountNum);

next custTable;
info("AccountNum: " + custTable.AccountNum);
```

For more examples, see Select statement.

Insert data

2/18/2021 • 7 minutes to read • Edit Online

You can use SQL statements, either interactively or in source code, to insert one or more rows into tables that are stored in the database.

- **insert method** – Insert one row at a time.
- **doInsert method** – Insert one row at a time.
- **insert_recordset statement** – Copy multiple records directly from one or more tables into another table in one database trip.
- **RecordInsertList.insertDatabase** – Insert multiple rows at the same time in one database trip. Use this construct when you don't have to sort the data.
- **RecordSortedList.insertDatabase** – Insert multiple rows at the same time in one database trip. Use this construct when you want a subset of data from a specific table, and you want that data to be sorted in an order that doesn't currently exist as an index.

RecordSortedList, **RecordInsertList**, and **insert_recordset** let you insert multiple records. By using these methods, you reduce communication between the application and the database. Therefore, you help increase performance. In some situations, record set–based operations can fall back to record-by-record operations. For more information, see Conversion of operations from set-based to record-by-record.

insert method

The **insert** method inserts one record at a time. It generates values for the **RecId** field and system fields, and then inserts the contents of the buffer (that is, the column values) into the database.

- Don't use a **select** statement on the table variable before you call the **insert** method.
- The **insert** method doesn't handle all the key field requirements and table dependencies. You must write code to handle them.

Here is how the **insert** method works:

- Only the specified columns of the rows that have been selected by the query are inserted into the named table.
- The columns of the table that is copied from and the columns of the table that is copied to must be type-compatible.
- If the columns of both tables match in type and order, the column list can be omitted from the **insert** clause.

The following example inserts a new record into the CustGroup table. The **CustGroup** column of the new record is set to **41**. Other fields in the record will be blank.

```
CustGroup custGroup;
ttsBegin;
    custGroup.CustGroup = '41';
    custGroup.insert();
ttsCommit;
```

To override the behavior of the **insert** method, use the **doInsert** method.

doInsert method

The **doInsert** method generates values for the **RecId** field and other system fields, and then inserts the contents of the buffer into the database. Use this method when the **insert** method on the table must be bypassed.

> **WARNING**
> A call to **doInsert** skips all logic, including database event handlers (for example **oninserting** and **oninserted**), chain-of-command **onInsert()**, and the **insert()** call itself. It's generally considered bad practice to use **doInsert**, and we don't recommend that you use it.

insert_recordset statement

The **insert_recordset** statement copies data directly from one or more source tables into one destination table in one server trip. It's faster to use **insert_recordset** than an array insert (**RecordInsertList.insertDatabase** or **RecordSortedList.insertDatabase**). However, array inserts are more flexible if you want to handle the data before you insert it. Although **insert_recordset** is a record set–based operator that performs operations on multiple records at a time, it can fall back to record-by-record operations in many situations. For more information, see Conversion of operations from set-based to record-by-record.

In the following syntax for the **insert_recordset** statement, brackets ([]) indicate optional elements of the statement.

insert_recordset *DestinationTable* **(** *ListOfFields* **)**

select *ListOfFields1* **from** *SourceTable* **[where** *WhereClause* **]**

[join *ListOfFields2* **from** *JoinedSourceTable* **[where** *JoinedWhereClause* **]]**

- *ListOfFields* in the destination table must match the list of fields in the source tables. Data is transferred in the order in which it appears in the list of fields. Fields in the destination table that aren't present in the list of fields are assigned **0** (zero) values, as in other areas. System fields, such as **RecId**, are assigned transparently by the kernel in the destination table.
- *WhereClause* and *JoinedWhereClause* are described in the *WhereClause* clause in the **select** statement.

insert_recordset: Inserting data from another table

In this example, the **Value** column in the NameValuePair table is summed for each **Name** value. The results of the aggregation are stored in the ValueSumByName table.

```
ValueSumByName valueSumName;
NameValuePair nameValuePair;

insert_recordset valueSumName (Name, ValueSum)
    select Name, sum(Value)
    from nameValuePair
    group by Name;
```

insert_recordset: Inserting data from variables

The following example shows that the **insert_recordset** statement can insert variable data.

- Include the **firstonly** keyword to insert only one new record. If you omit **firstonly**, a record is inserted for each record in the CustTable table.
- Literals, such as **128** or **"this literal string"**, can't be used in the query as a source of data that is inserted.
- The columns in the source table don't have to correspond to the target table.

In this example, one new record is inserted into the NameValuePair table. This record has an **Id** value of **1**, a

Name value of **Name1**, and a **Value** value of **1**.

```
NameValuePair nameValuePair;
CustTable custTable;

int id_var = 1;
str name_var = 'Name1';
int value_var = 1;

insert_recordset nameValuePair (Id, Name, Value)
select firstonly id_var, name_var, value_var from custTable;
```

insert_recordset: Inserting data by using a join

The following example shows a join of three tables on an **insert_recordset** statement that has a subselect. It also shows a **while select** statement that has a similar join. A variable is used to supply the inserted value for one column. The **str** variable must be declared, and it must have a length that is less than or equal to the maximum length of the corresponding database field.

In this example, there is an **insert_recordset** statement for the tabEmplProj5 table. One of the target fields is named **Description**, and its data comes from the local **sDescriptionVariable** variable. The **insert_recordset** statement succeeds even when the configuration key for the **Description** field is turned off. The system ignores both the **Description** field and the **sDescriptionVariable** variable. Therefore, this code provides an example of *configuration key automation*. Configuration key automation occurs when the system can automatically adjust the behavior of an **insert_recordset** statement that inserts data into fields that the configuration key is turned off for.

```
static void InsertJoin42Job(Args _args)
{
    GmTabDepartment tabDept2;
    GmTabEmployee tabEmpl3;
    GmTabProject tabProj4;
    GmTabEmployeeProject tabEmplProj5;
    str 64 sDescriptionVariable = "From variable.";
    DELETE_FROM tabEmplProj5;
    INSERT_RECORDSET tabEmplProj5
        (
        Description
        , EmployeeRecId
        , ProjectRecId
        )
    Select
        sDescriptionVariable
        , RecId
    from
        tabEmpl3
        join
            tabDept2
            where tabEmpl3 .DepartmentGuid == tabDept2 .DepartmentGuid
        join RecId
            from tabProj4
            where tabDept2 .DepartmentGuid == tabProj4 .DepartmentGuid
    info(int642str(tabEmplProj5 .rowCount())
        + " ==Number of rows inserted.");
    WHILE SELECT *
        from
            tabEmplProj5
            join tabEmpl3
                where tabEmplProj5 .EmployeeRecId == tabEmpl3 .RecId
            join tabProj4
                where tabEmplProj5 .ProjectRecId == tabProj4 .RecId
    {
        info(
            tabEmpl3 .EmployeeName
            + "  --works on--  "
            + tabProj4 .ProjectName
            + " (" + tabEmplProj5 .Description + ")."
            );
    }

/*****************  Actual Infolog output
Message (01:05:41 pm)
4 ==Number of rows inserted.
Alice  --works on--  Project ZZZ (From variable.).
Alice  --works on--  Project YY (From variable.).
Beth  --works on--  Project ZZZ (From variable.).
Beth  --works on--  Project YY (From variable.).
****************/
}
```

Handling DuplicateKeyException exceptions

The following example shows how you can catch a **DuplicateKeyException** exception in the context of an explicit transaction. The exception is thrown when a call to **xRecord.insert** fails because the key value already exists. In the **catch** block, your code can either take corrective action or log the error for later analysis. Your code can then continue without losing all the pending work of the transaction. You can't catch a **DuplicateKeyException** exception that is caused by a set-based operation such as **insert_recordset**.

This example depends on two tables: SourceTable and DestinationTable. Each table has one mandatory integer field. The fields are named **SourceKeyField** and **DestinationKeyField**, respectively. A unique index is defined

on each key field. The SourceTable table must have at least one record in it.

```
static void JobDuplicKeyException44Job(Args _args)
{
    SourceTable sourceTable; // Must have at least one record.
    DestinationTable destinationTable;
    int countTries = 0;
    int numberAdjust = 0;
    int newKey;
    int inote;
    container notes;

    // Empty the destination table.
    delete_from destinationTable;

    // Copy all the records from SourceTable to DestinationTable
    insert_recordset destinationTable (destinationKeyField)
        select SourceKeyField from sourceTable order by SourceKeyField asc;

    // Copy the records from SourceTable to DestinationTable, one at a time.
    // This immediately throws a DuplicateKeyException.
    ttsBegin;
    try
    {
        countTries++;
        notes += strFmt("Inside the try block, try count is %1.", countTries);
        while select * from sourceTable order by SourceKeyField asc
        {
            destinationTable.clear();
            newKey = sourceTable.SourceKeyField + numberAdjust;
            destinationTable.DestinationKeyField = newKey;
            notes += strFmt("%1 is the key to be tried." , newKey);
            destinationTable.insert();
            notes += "Success: .insert()";
        }
        ttsCommit;
    }
    catch (Exception::DuplicateKeyException, destinationTable) // Table is optional.
    {
        notes += "Inside the catch block.";
        notes += 'Error: ' + infolog.text().strReplace('\n', '');
        if (countTries <= 1)
        {
            notes += "Will issue retry.";
            numberAdjust = 1;
            retry; // Erases Infolog.
        }
        else
        {
            notes += "Aborting the transaction.";
            ttsAbort;
        }
    }

    for (inote = 1; inote <= conLen(notes); inote++)
    {
        info(conPeek(notes, inote));
    }
}

/* Output
    ---- Inside the try block, try count is 1.
    ---- 11 is the key to be tried.
    ---- Inside the catch block.
    Cannot create a record in DestinationTable (DestinationTable).
    The record already exists.
    ---- Will issue retry.
    ---- Inside the try block, try count is 2.
```

```
    ---- 12 is the key to be tried.
    ---- .insert() successful.
*/
```

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Update data

2/18/2021 • 3 minutes to read • Edit Online

You can use SQL statements, either interactively or in source code, to update one or more rows in a table that is stored in the database.

- **update method** – Update the current record with the contents of the buffer. Also update the appropriate system fields.
- **doUpdate method** – Update one row at a time.
- **update_recordset statement** – Update multiple records in one database trip. By using the **update_recordset** statement, you reduce communication between the application and the database. Therefore, you help increase performance. In some situations, record set–based operations can fall back to record-by-record operations. For more information, see Conversion of operations from set-based to record-by-record.

update method

The **update** method updates the current record with the contents of the buffer. It also updates the appropriate system fields. The optional **where** clause specifies a condition that the **update** method tests as it processes each row of the table. Only those rows that test **true** against the condition are updated with the new values.

The following example selects the CustTable table for update. Only records where the value of the **AccountNum** field equals **4000** are updated. Because there is no call to **next**, and this example doesn't use a **select while** statement, only one record is updated. The value of the **CreditMax** field is changed to **5000**.

```
CustTable custTable;
ttsBegin;
    select forUpdate custTable
        where custTable.AccountNum == '4000';
    custTable.CreditMax = 5000;
    custTable.update();
ttsCommit;
```

doUpdate method

To override the behavior of the **update** method, use the **doUpdate** method. The **doUpdate** method updates the current record with the contents of the buffer. It also updates the appropriate system fields. You should use the **doUpdate** method when the **update** method on the table must be bypassed. The syntax for a **doUpdate** table method is **void doUpdate()**.

WARNING

A call to **doUpdate** skips all logic, including database event handlers (for example **onUpdating** and **onUpdated**), chain-of-command **onUpdate()**, and the **update()** call itself. It's generally considered bad practice to use **doUpdate**, and we don't recommend that you use it.

```
CustTable custTable;
ttsBegin;
select forUpdate custTable
    where custTable.CreditMax == 3000;
if (custTable)
{
    custTable.CreditMax += 1000;
    custTable.doUpdate();
}
ttsCommit;
```

update_recordset statement

The **update_recordset** operator is a record set–based operator that updates multiple records in one trip to the server. Therefore, the power of Microsoft SQL Server can help improve the performance of some tasks. The **update_recordset** statement resembles **delete_from** in X++ and **update set** in SQL. It doesn't retrieve each record separately by fetching, changing, and updating. Instead, it works on an SQL-style record set on the database server side. If the **update** method is overridden, the implementation falls back to a classic looping construction, where one record at a time is updated. (This behavior resembles the behavior of **delete_from** for deletions.) Therefore, the construction works on temporary tables and whole table–cached tables by using the looping construction.

The following example updates the CustTable table and increments the value in the **CreditMax** column by **1000** for records where the **CreditMax** value is more than **0** (zero).

```
CustTable custTable;
ttsBegin;
update_recordset custTable
    setting CreditMax = custTable.CreditMax + 1000
    where custTable.CreditMax > 0;
ttsCommit;
```

The following example updates multiple columns.

```
CustTable custTable;
ttsBegin;
update_recordset custTable
    setting
        CreditMax = custTable.CreditMax + 1000,
        AccountStatement = CustAccountStatement::Always
    where custTable.CreditMax > 0;
ttsCommit;
```

The following example shows that the **update_recordset** statement supports joins of several tables. Data from the joined tables can be used to assign values to fields in the table that is being updated.

```
TableEmployee tabEmpl;
TableDepartment tabDept;
TableProject tabProj;
update_recordset tabEmpl
    setting
        currentStatusDescription = tabDept.DeptName + ", " + tabProj .ProjName
join tabDept
    where tabDept.DeptId == tabEmpl.DeptId
join tabProj
    where tabProj.ProjId == tabEmpl .ProjId;
```

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Delete data

2/18/2021 • 5 minutes to read • Edit Online

You can use SQL statements, either interactively or in source code, to delete one or more rows from tables that are stored in the database.

- **delete method** – Delete one row at a time.
- **doDelete method** – Delete one row at a time.
- **delete_from statement** – Delete multiple rows at the same time. By using the **delete_from** statement, you reduce communication between the application and the database. Therefore, you help increase performance. In some situations, this set-based operation can fall back to a record-by-record operation. For more information, see Conversion of operations from set-based to record-by-record.

delete method

The **delete** method deletes the current record from the database. To use this method, use a **where** clause to specify the rows to delete. One record at a time is then removed from the specified table.

The **delete** method can be overridden. For example, you might want to add extra validation before records are deleted. If you override the **delete** method, you can run the original (base) version of the **delete** method by calling the **doDelete** method. Therefore, a call to the **doDelete** method is equivalent to a call to **super()** in the **delete** method.

In the following example, all records in the NameValuePair table that satisfy the **where** clause (that is, all records where the value of the **Name** field equals **Name1**) are deleted from the database. One record is deleted at a time.

```
ttsBegin;
    NameValuePair nameValuePair;

    while select forUpdate nameValuePair
        where nameValuePair.Name == 'Name1'
    {
        nameValuePair.delete();
    }
ttsCommit;
```

The following example deletes records from the LedgerJournalTrans table and updates the associated number sequence.

```
int counter = 0;
str _journalNum = '';
str _voucher = '';
LedgerJournalTrans ledgerJournalTrans;
LedgerJournalTable ledgerJournalTable;

ttsBegin;
    while select forUpdate ledgerJournalTrans
        index hint NumVoucherIdx
        where ledgerJournalTrans.journalNum == _journalNum
            && ledgerJournalTrans.voucher == _voucher
    {
        ledgerJournalTrans.doDelete();
        counter++;
    }

    if (counter && ledgerJournalTable.journalType != LedgerJournalType::Periodic)
    {
        NumberSeq::release(ledgerJournalTable.voucherSeries, _voucher);
    }
ttsCommit;
```

doDelete method

Like the **delete** table method, the **doDelete** table method deletes the current record from the database. Use the **doDelete** method if the **delete** table method has been overridden, and you want to run the original (base) version of that method instead of the overridden version. Therefore, a call to the **doDelete** method is equivalent to a call to **super()** in the **delete** method.

> **WARNING**
> A call to **doDelete** skips all logic, including database event handlers (for example, **onDeleting** and **onDeleted**), chain-of-command **onDelete()**, and the **delete()** call itself. It's generally considered bad practice to use **doDelete**, and we don't recommend that you use it.

delete_from statement

The **delete_from** operator is a record set–based operator that removes multiple records at the same time. This approach can be more efficient and faster than an approach that uses the **delete** method in a loop to delete one record at a time. If you've overridden the **delete** method, the system interprets the **delete_from** statement into code that calls the **delete** method one time for each row that is deleted.

The following example deletes all records in the NameValuePair table where the value in the **Name** column is **Name1**.

```
NameValuePair nameValuePair;
delete_from nameValuePair where nameValuePair.Name == 'Name1';
```

In contrast to the previous example, the following example is inefficient, because it issues a separate SQL **delete** call to the database server for each record. The **delete** method never deletes more than one record per call.

```
// Example of inefficient code.
MyWidgetTable tabWidget; // extends xRecord.
ttsBegin;
    while select forUpdate tabWidget
        where tabWidget .quantity <= 100
    {
        tabWidget.delete();
    }
ttsCommit;
```

A delete operation that has an inner join

Inner joins aren't supported on the **delete_from** statement. Therefore, you can't use the unmodified **join** keyword on the **delete_from** statement. However, you can logically perform an inner join by using other techniques.

The following example shows the recommended way to use the **delete_from** method and inner joins. This example is relatively efficient. It issues a separate **delete_from** statement for each loop iteration. However, each **delete_from** statement can delete multiple records (a subset of all the records that the job deletes).

```
MyWidgetTable tabWidget; // extends xRecord.
ttsBegin;
while select from tabGalaxy
    where tabGalaxy .isTrusted == 0
{
    delete_from tabWidget
        where tabWidget .GalaxyRecId == tabGalaxy .RecId;
}
ttsCommit;
```

A delete operation that uses the notexists join keyword

You can use the **notexists join** keyword pair in a **delete_from** statement. The **delete_from** statements in the following example are efficient. The **notexists join** clause enables the **delete_from** statement to delete a specific set of rows. In this example, the **delete_from** statement removes all parent-order header rows that there are no child-order line rows for. You can also use the **exists join** clause on the **delete_from** statement.

```
static void DeleteFromNotexists3bJob(Args _args)
{
    GmTabOrderHeader tabOHeader;
    GmTabOrderLine tabOLine;
    AddressState tabAddressState;
    str 127 sOH_Info;
    str 127 sOL_Data;
    int64 i64OHRecId;
    delete_from tabOLine;
    delete_from tabOHeader;
    // Inserts into parent table.
    sOH_Info = "Albert needs tires.";
    insert_recordset tabOHeader
        (OH_Info)
        select firstOnly sOH_Info from tabAddressState;
    sOH_Info = "Benson wants plastic.";
    insert_recordset tabOHeader
        (OH_Info)
        select firstOnly sOH_Info from tabAddressState;
    // Obtain a OrderHeader RecId,
    // use it to insert one child row.
    sOL_Data = "4 re-treads.";
    while select firstOnly tabOHeader
        order by OH_Info
        where tabOHeader .OH_Info like "A*"
    {
```

```
    {
        i64OHRecId = tabOHeader .RecId;
        insert_recordset tabOLine
            (OL_Data ,OrderHeaderRecId)
            select firstOnly
                sOL_Data ,i64OHRecId
                from tabAddressState;
        break;
    }
    // Before the delete notexists.
    // Display all parent, and then all child rows.
    while select tabOHeader
        order by OH_Info
    {
        info(strFmt("Before: OHeader:  OH_Info==%1 , RecId==%2"
            ,tabOHeader .OH_Info ,tabOHeader .RecId));
    }
    while select tabOLine
        order by OL_Data
    {
        info(strFmt("Before: OLine:  OL_Data==%1 , OrderHeaderRecId==%2"
            ,tabOLine .OL_Data ,tabOLine .OrderHeaderRecId));
    }
    // Delete_From NotExists Join, to remove from the
    // parent table all order headers without children.
    delete_from tabOHeader
        notexists join tabOLine
            where tabOHeader .RecId ==
                tabOLine .OrderHeaderRecId;
    info(strFmt("%1 is the number of childless OHeader records deleted."
        ,tabOHeader.rowCount()));
    // After the delete notexists.
    // Display all parent, and then all child rows.
    info("- - - - - - - - - - - - - - -");
    while select tabOHeader
        order by OH_Info
    {
        info(strFmt("After: OHeader:  OH_Info==%1 , RecId==%2"
            ,tabOHeader .OH_Info ,tabOHeader .RecId));
    }
    while select tabOLine
        order by OL_Data
    {
        info(strFmt("After: OLine:  OL_Data==%1 , OrderHeaderRecId==%2"
            ,tabOLine .OL_Data ,tabOLine .OrderHeaderRecId));
    }

/**************  Actual Infolog output
Message (12:54:14 pm)
Before: OHeader:  OH_Info==Albert needs tires. , RecId==5637144608
Before: OHeader:  OH_Info==Benson wants plastic. , RecId==5637144609
Before: OLine:  OL_Data==4 re-treads. , OrderHeaderRecId==5637144608
1 is the number of childless OHeader records deleted.
- - - - - - - - - - - - - - -
After: OHeader:  OH_Info==Albert needs tires. , RecId==5637144608
After: OLine:  OL_Data==4 re-treads. , OrderHeaderRecId==5637144608
**************/
}
```

While select statement

2/18/2021 • 2 minutes to read • Edit Online

A **while select** statement is used to handle data. It's the most widely used form of the **select** statement. The **while select** statement loops over many records that meet specific criteria, and can run a statement on each record. The syntax of a **while select** statement resembles the syntax of a **select** statement, but the statement is preceded by **while select** instead of **select**.

- Typically, when you use the **while select** statement for data manipulation, you use it in a transaction to ensure data integrity.
- The results of the **while select** statement are returned in a table buffer variable.
- If you use a field list in the **select** statement, only those fields are available in the table variable.
- If you use aggregate functions, such as **sum** or **count**, the results are returned in the fields that you perform the **sum** or **count** over. You can count, average, or sum only integer and real fields.
- The **select** statement itself is run only one time, immediately before the first iteration of the statements in the loop.
- Any Boolean expressions that are added to the **while select** statement (for example, **iCounter < 1**) are tested only one time. This behavior differs from the behavior of the **while** statement in languages such as C++ and C#. For example, the following loop can have more than one iteration.

```
int iCounter = 0;
BankAccountTable xrecBAT;

while select * from xrecBAT
    where iCounter < 1
{
    iCounter++;
    info(strFmt("%1 , %2", iCounter, xrecBAT.AccountID));
}
```

The following example prints the **AccountNum** and **SalesGroup** values of every customer in the CustTable table whose account number is within a specified range.

```
CustTable custTable;

while select custTable
    order by custTable.AccountNum
    where  custTable.AccountNum >= '4010' && custTable.AccountNum <= '4100'
{
    info(strFmt("%1 , %2", custTable.AccountNum, custTable.SalesGroup));
}
```

Write select statements as expressions

2/18/2021 • 2 minutes to read • Edit Online

You can use a **select** statement as an expression. This type of **select** statement is known as an *expression **select** statement*.

- You can't use a table buffer variable in an expression **select** statement.
- The name of the table must be used in the **from** clause.
- The **join** keyword isn't supported.
- The table name can't be used to qualify a field name in the **order by** clause.
- In a **where** clause, the table name must be used as a qualifier of the field.
- You can mention only one table in an expression **select** statement. Therefore, subselects aren't supported as a workaround for the unsupported **join** keyword.
- The only column that can be filled with data is the column that is named before the **from** clause in the **select** clause.
- After the closing parenthesis, the name of a column is used to reference the data value.

The following expression returns the value in the **AccountNum** column of the first row in the CustTable table (if a row exists).

```
str accountNum = (select AccountNum from CustTable order by AccountNum desc).AccountNum;
info('Max AccountNum: ' + accountNum);
```

Here is a simpler way to achieve the same result as the previous example.

```
str accountNum = (select maxof(AccountNum) from CustTable).AccountNum;
info('Max AccountNum: ' + accountNum);
```

The following example returns the maximum **RecId** value of customers that aren't blocked. Here, the **maxof** aggregate function is used, and the **RecId** field is mentioned in the function. The field that is mentioned in the aggregate function must match the field name that is used to reference the data value after the closing parenthesis. Otherwise, empty data is returned.

```
int64 nRecId = (select maxof(RecId) from CustTable
            where CustTable.Blocked == CustVendorBlocked::No).RecId;
info("Max RecId: " + int642Str(nRecId));
```

In the following example, the **RecId** field is used to reference a data value that isn't a **RecId** value. The **count** aggregate function doesn't return a **RecId** value. It's a typical practice to use the **RecId** field with the **count** function.

```
int64 nRecId = (select count(RecId) from CustTable
            where CustTable.Blocked == CustVendorBlocked::No).RecId;
info('Count of unblocked customers: ' + int642Str(nRecId));
```

select statements on fields

You can use a **select** statement in a lookup on a field. After a **select** statement that fetches a record in a table,

you can enter **.fieldName** to reference a field in the table. These **select** statements must be used in expressions. A *normal **select** statement* differs from a *field **select** statement* in the following way:

- The field **select** statement operates directly on a table.
- The normal **select** statement operates on a table buffer variable.

The following examples shows how to access fields from a select statement.

```
print((select CustTable order by AccountStatement).AccountStatement);

if ((select custTable where CustTable.AccountNum == '3000').CreditMax < 5000)
{
    info('This customer has a credit maximum less than $5000.');
}
```

X++ transactional integrity

2/18/2021 • 2 minutes to read • Edit Online

This topic describes transactional integrity in the X++ language.

If you don't take steps to ensure the integrity of transactions, data corruption can occur. At the very least, you might experience poor scalability with respect to concurrent users on the system. Two internal checking features help ensure the integrity of transactions: the **forUpdate** check and the **ttsLevel** check.

- A **forUpdate** check helps ensure that a record can be updated or deleted only if it has first been selected for update. You can select a record for update by using either the **forUpdate** keyword in the **select** statement or the **selectForUpdate** method on the table.
- A **ttsLevel** check helps ensure that a record can be updated or deleted only in the same transaction scope where it was selected for update.

The following statements are used to help ensure integrity:

- **ttsBegin** – This statement marks the beginning of a transaction. It helps ensure data integrity and also helps ensure that all updates that are done until the transaction ends (through **ttsCommit** or **ttsAbort**) are consistent.
- **ttsCommit** – This statement marks the successful end of a transaction. It ends and commits a transaction. The Finance and Operations app ensures that a transaction that has been committed will be performed according to intentions.
- **ttsAbort** – This statement lets you explicitly discard all changes in the current transaction. In this case, the database is rolled back to the original state, where nothing has been changed. Typically, you use this statement if you've detected that the user wants to break the current job. The **ttsAbort** statement helps ensure that the database is consistent.

Usually, it's a better idea to use exception handling instead of **ttsAbort**. The **throw** statement automatically aborts the current transaction. As the following example shows, statements between **ttsBegin** and **ttsCommit** can include one or more transaction blocks. In these cases, nothing is committed until a successful exit from the final **ttsCommit** statement occurs.

```
ttsBegin;
    // Some statements.
    ttsBegin;
        // More statements.
    ttsCommit;
ttsCommit;
```

The following example selects a set of records and updates the **CustGroup** field. This code will throw an exception if the **select** statement doesn't return any records.

```
Custtable custTable;
ttsBegin;
    select forUpdate custTable where custTable.AccountNum == '5000';
    custTable.CustGroup = '1';
    custTable.update();
ttsCommit;
```

Example of code that is rejected by the forUpdate check

In this example, the first failure occurs because the **forUpdate** keyword is missing.

```
ttsBegin;
    select myTable; // Rejected by the forUpdate check.
    mytable.myField = 'xyz';
    myTable.update();
ttsCommit;
```

Example of code that is rejected by the ttsLevel check

In this example, the failure occurs because the transaction scope of the update differs from the transaction scope where the record was selected for update in **ttsCommit**.

```
ttsBegin;
    select forUpdate * from myTable;
    myTable.myField = 'xyz';
ttsCommit;

ttsBegin;
    myTable.update(); // Rejected by the ttsLevel check.
ttsCommit;
```

Conversion of operations from set-based to record-by-record

2/18/2021 • 2 minutes to read • Edit Online

You can use the following statements and methods to help improve performance by reducing communication between the application and the database:

- delete_from
- update_recordset
- insert_recordset
- RecordSortedList.insertDatabase
- RecordInsertList.insertDatabase

In some situations, these record set–based operations can be converted to slower record-by-record operations. The following table identifies these situations.

SITUATION	DELETE_FROM	UPDATE_RECORDSET	INSERT_RECORDSET	RECORDSORTEDLIST, RECORDINSERTLIST	USED TO OVERRIDE
Non-SQL tables	Yes	Yes	Yes	Yes	Not applicable
Delete actions	Yes	No	No	No	**skipDeleteActions**
The database log is enabled.	Yes	Yes	Yes	No	**skipDatabaseLog**
Overridden method	Yes	Yes	Yes	Yes	**skipDataMethods**
Alerts are set up for the table.	Yes	Yes	Yes	No	**skipEvents**
The **ValidTimeStateFieldType** property on a table is set to a value other than **None**.	Yes	Yes	Yes	Yes	Not applicable

You can use the **skip*** settings that are shown in the "Used to override" column to explicitly skip or ignore one or more factors that adversely affect performance. If one of the previously mentioned SQL operations is downgraded to a record-by-record operation, all the **skip*** settings are ignored. In the following example code, the **insert** method on the myTable table is run, even though it's explicitly stated that this method should be skipped if a container or memo field is defined for myTable.

```
public void tutorialRecordInsertList()
{
    MyTable myTable;
    RecordInsertList insertList = new RecordInsertList(
        myTable.TableId,
        True);
    int i;
    for ( i = 1; i <=  100; i++ )
    {
        myTable.value = i;
        insertList.add(myTable);
    }
    insertList.insertDatabase();
}
```

Access data by using the SysDa classes

2/18/2021 • 5 minutes to read • Edit Online

This topic explains how to create extensible queries by using the SysDa application programming interface (API).

The extensible SysDa API provides almost all the data access possibilities that are available in X++. In fact, the APIs are wrappers around the code that the X++ compiler would generate. Therefore, use of the SysDa classes carries no overhead, unlike use of the **QueryRun** object, for example. Additionally, the check that the X++ compiler does on data access statements is your responsibility. For example, you create a **where** clause that compares a globally unique identifier (GUID) to an integer. The X++ compiler would diagnose this clause as an error.

The SysDa APIs include an extensive set of APIs for creating custom queries. However, there is a smaller set of types that drives the primary query activities:

- Select: **SysDaQueryObject**, **SysDaSearchObject**, and **SysDaSearchStatement**
- Update: **SysDaUpdateObject** and **SysDaUpdateStatement**
- Insert: **SysDaInsertObject** and **SysDaInsertStatement**
- Delete: **SysDaQueryObject**, **SysDaDeleteObject**, and **SysDaDeleteStatement**

The following sections provide examples of each type of query and the customizations that it supports. The examples use a table that is named TestTable. This table has two fields: a string field that is named **stringField** and an integer field that is named **intField**.

Select query

To run a **select** query, follow these steps.

1. Create and configure a **SysDaQueryObject** object that specifies the table instance that will contain the designated records.
2. Create a **SysDaSearchObject** object, and pass the **SysDaQueryObject** object to the constructor.
3. Iterate over the results of the query by passing the **SysDaSearchObject** object to the **SysDaSearchStatement.next()** method.

The following example finds all rows in TestTable where **intField** <= **5**.

```
// t is the table buffer that will hold the result.
TestTable t;

// Create the query.
var qe = new SysDaQueryObject(t);

// Add clauses to the query. First the projection.
var s = qe.projection()
    .add(fieldStr(TestTable, intField))
    .add(fieldStr(TestTable, stringField));

// At this point the query is:
// intField, stringField FROM TestTable

// Add a where clause to include rows where intField is <= 5.
qe.WhereClause(new SysDaLessThanOrEqualsExpression(
    new SysDaFieldExpression(t, fieldStr(TestTable, intField)),
    new SysDaValueExpression(5)));

// Now the query is:
// intField, stringField FROM TestTable WHERE (TestTable.intField<= 5)

// Order the results by intField.
qe.OrderByClause().addDescending(fieldStr(TestTable, intField));

// Now the query is:
// intField, stringField FROM TestTable ORDER BY intField DESC WHERE (TestTable.intField<= 5)

var so = new SysDaSearchObject(qe);
var ss = new SysDaSearchStatement();

// Enumerate the designated values by using ss.
while (ss.next(so))
{
    info(t.stringField);
}
```

Update statement

To run an **update** statement, follow these steps.

1. Create and configure a **SysDaUpdateObject** object.
2. Update data by passing the **SysDaUpdateObject** object to the **SysDaUpdateStatement.execute()** object. Because updates modify the data in the database, you must wrap the call to **execute** in **ttsbegin** and **ttscommit** statements.

The following example updates **stringField** to **"fifty"** for all rows where **intField** = **50**.

```
TestTable t;

// Create an update query to find rows where intField = 50.
var uo = new SysDaUpdateObject(t);

// Set stringField to "fifty".
uo.settingClause()
    .add(fieldStr(TestTable, stringField), new SysDaValueExpression("fifty"));

// At this point the update statement is:
// UPDATE_RECORDSET TestTable SETTING stringField=fifty

uo.whereClause(new SysDaEqualsExpression(
    new SysDaFieldExpression(t, fieldStr(TestTable, intField)),
    new SysDaValueExpression(50)));

// Now the update statement is:
// UPDATE_RECORDSET TestTable SETTING stringField=fifty WHERE (TestTable.intField == 50)

// Update the rows.
ttsbegin;
    new SysDaUpdateStatement().execute(uo);
ttscommit;

// Verify the results of the update query.
TestTable t1;
select intField, stringField from t1 where t1.intField == 50;
info("Updated value is: " + t1.stringField);
// Output is: "Updated value is: fifty".
```

Insert statement

To run an **insert** statement, follow these steps.

1. Create and configure a **SysDaInsertObject** object to specify which fields are updated during the insertion.
2. Create and configure a **SysDaQueryObject** object that specifies the source of the rows to insert. The order of the fields in **SysDaQueryObject.projection()** must match the order of the fields in **SysDaInsertObject.fields()**.
3. Assign the **SysDaQueryObject** object to the **SysDaInsertObject** object.
4. Insert the new row by passing the **SysDaInsertObject** object to the **SysDaInsertStatement.executeQuery()** method.

The following example inserts rows where **intField** = **40** and **stringField** = **"en-us"** into TestTable.

```
TestTable t;

// Specify the fields in the new row.
var insertObject = new SysDaInsertObject(t);
insertObject.fields()
    .add(fieldStr(TestTable, stringField))
    .add(fieldStr(TestTable, intField));

// At this point the insert statement is:
// INSERT_RECORDSET TestTable(stringField, intField) SELECT

// Retrieve the data to insert from the LanguageTable by using a query.
LanguageTable source;
var qe = new SysDaQueryObject(source);

var s1 = qe.projection()
    .Add(fieldStr(LanguageTable, LanguageId))
    .AddValue(40);

// The query statement is:
// LanguageId, 40 FROM LanguageTable

qe.WhereClause(new SysDaEqualsExpression(
        new SysDaFieldExpression(source, fieldStr(LanguageTable, LanguageId)),
        new SysDaValueExpression("en-us")));

// Now the query is:
// LanguageId, 40 FROM LanguageTable WHERE (LanguageTable.LanguageId == en-us)

// Assign the query to the insert statement.
insertObject.query(qe);

// The insert statement is now:
// INSERT_RECORDSET TestTable(stringField, intField) SELECT LanguageId, 40 FROM LanguageTable WHERE
(LanguageTable.LanguageId == en-us)

var insertStmt = new SysDaInsertStatement();
ttsbegin;
    insertStmt.executeQuery(insertObject);
ttscommit;

// Verify the results of the insert query.
TestTable t1;
select * from t1 where t1.stringField == "en-us";
info(any2Str(t1.intField) + ":" + t1.stringField);
// The output is "40:en-us".
```

Delete statement

To run a **delete** statement, follow these steps.

1. Create and configure a **SysDaQueryObject** object to specify which rows to delete.
2. Create a **SysDaDeleteObject** object, and pass the **SysDaQueryObject** object to the constructor.
3. Delete the rows by passing the **SysDaDeleteObject** object to the **SysDaDeleteStatement.executeQuery()** method.

The following example deletes rows where **intField** is an even number.

```
TestTable t;

// Build the query that specifies which rows to delete.
var qe = new SysDaQueryObject(t);

var s = qe.projection()
    .add(fieldStr(TestTable, intField));

// At this point the query is:
// intField FROM TestTable

// Delete rows where intField is even.
qe.WhereClause(new SysDaEqualsExpression(
    new SysDaModExpression(
    new SysDaFieldExpression(t, fieldStr(TestTable, intField)),
    new SysDaValueExpression(2)),
    new SysDaValueExpression(0)));

// Now the query is:
// intField FROM TestTable WHERE ((TestTable.intField MOD 2) == 0)

var ds = new SysDaDeleteStatement();
var delobj = new SysDaDeleteObject(qe);

// The deletion statement, from the SysDaDeleteObject, is:
// DELETE_FROM intField FROM TestTable WHERE ((TestTable.intField MOD 2) == 0)

ttsbegin;
    ds.executeQuery(delobj);
ttscommit;

info("Number of rows after deletion: " + any2Str(t.RowCount()));
```

Clauses

SysDa queries support several clauses:

- **whereClause** – The **where** clause is constructed from objects that inherit from **SysDaQueryExpression**. Examples are **SysDaEqualsExpression**, **SysDaNotEqualsExpression**, and **SysDaLessThanExpression**. You can find the full list by filtering in Application Explorer.
- **orderByClause**
- **groupByClause**
- **joinClause** with **joinClauseKind**
- **joinedQuery**
- **settingClause**

Troubleshooting

You can use the **toString()** method on **SysDaQueryObject**, **SysDaUpdateObject**, **SysDaInsertObject**, and **SysDaQueryObject** objects to view the statement that you're building.

Mitigate a SQL injection attack

2/18/2021 • 4 minutes to read • Edit Online

IMPORTANT

Some or all of the functionality noted in this topic is available as part of a preview release. The content and the functionality are subject to change. For more information about preview releases, see Service update availability.

An SQL injection attack occurs when malicious data values are passed to Microsoft SQL Server in a query string. Those values can cause lots of damage in a database. SQL injection can occur if you aren't careful about how you use a query to pass data that comes from an uncontrolled source, such as user input, to SQL Server. SQL injection isn't usually an issue in Finance and Operations apps, because the built-in data access statements in X++ prevent it. However, if you use Direct-SQL, SQL injection can occur when raw SQL code is passed to the server.

A new API will help mitigate these attacks. The API is available starting with platform updates for version 10.0.17 of Finance and Operations apps (April 2021).

The issue

Consider a scenario where a developer writes the following code to look up the first name of customers, based on their last name.

```
public str GetFirstName(str name)
{
    str sqlStatementText = "SELECT TOP(1) firstName FROM Customer WHERE customer.Name = '" + name + "'";

    // Create a connection to the SQL Server
    var connection = new Connection();

    // Create a statement and submit the sql statement to the server:
    Statement statement = connection.createStatement();
    var results= statement.executeQuery(sqlStatementText);

    // Get the first record:
    results.next();
    statement.close();

    // Harvest the results.
    return results.getString(1);
}
```

Additionally, there is either a page where users can enter customer names in a string field, or a service endpoint that enables names to come into the server.

In this scenario, everything works well if users enter valid names such as "Jones." However, a malicious user might enter the following string as a name.

```
'; drop table Customer --
```

In this case, here is the final query that the server runs.

```
SELECT TOP(1) firstName FROM Customer WHERE customer.Name = ''; drop table Customer --'
```

The first quotation mark in the given string just ends the string literal that should contain the name that the user is looking for. Then another SQL statement is run because of the semicolon (;), which is a statement terminator token. This second statement irretrievably deletes the **Customer** table and all the data in it. Finally, the commenting characters (--) ensure that the single quotation mark at the end doesn't cause syntax errors. Therefore, the string is valid Transact SQL (T-SQL).

SQL injection occurs because the connection to SQL Server doesn't impose any restrictions that prevent it from performing operations that create or delete tables, views, and stored procedures at runtime. Therefore, organizations must rely on the assumption that developers are reasonable people who know what they are doing.

The solution

SQL Server mitigates the threat by using *statement parameters*. Statement parameters never use literals that are subject to textual changes to the resulting string. Instead, they provide named parameters, the actual content of which is provided contextually. For this release, Microsoft has added a new API that lets you use parameters instead of building SQL strings in code.

The following example shows what the code from the previous example looks like after these changes are incorporated.

```
public str GetFirstName(str name)
{
    str sqlStatementText = "SELECT TOP(1) firstName FROM Customer WHERE customer.Name = @Name";

    // Create a connection to the SQL Server
    var connection = new Connection();

    // Submit the sql statement to the server:
    Statement statement = connection.createStatement();

    // Create a mapping from parameter names onto values
    Map paramMap = SqlParams::create();
    paramMap.add('Name', name);

    // Execute the query, providing both the query
    // and the parameters.
    var results= statement.executeQueryWithParameters(sqlStatementText, paramMap);

    // Capture the results:
    results.next();
    statement.close();

    return results.getString(1);
}
```

The updated example uses the new **executeQueryWithParameters** API instead of the old API that didn't take parameters. The code builds the map that contains the mapping from parameter names to parameter values. In this case, **Name** will be the value of **@Name** in SQL. The incoming **name** value can be anything.

A related method on the **Statement** type is used to run statements that return integer values instead of rows. Typically, the integer value indicates the number of rows that are affected. The following example uses the X++ data statements with the **executeQueryWithParameters** API.

```
public void InsertWithStrParameter()
{
    var connection = new Connection();
    Statement statement = connection.createStatement();

    connection.ttsbegin();

    str sql = @"
        UPDATE Wages
        SET Wages.Wage = Wages.Wage * @percent
        WHERE Wages.Level = @Level";

    Map paramMap = SqlParams::create();
    paramMap.add('percent', 1.1);        // 10 percent increase
    paramMap.add('Level', 'Manager');    // Management increase

    int cnt = statement.executeUpdateWithParameters(sql, paramMap);
    statement.close();

    connection.ttscommit();
}
```

Conclusion

As Microsoft introduces the new methods, we are also marking the existing methods (that is, the methods without the parameters) as obsolete. The usual deprecation periods apply. Therefore, you can update your code to take advantage of the new protection that the parameters provide.

Although the new **executeQueryWithParameters** API helps you protect your customers from disasters, you aren't required to use it. You can still do string concatenations and provide an empty parameter set. However, in this case, you don't gain the advantages that the parameters provide. We hope that you will take this opportunity to eliminate any dangerous usage that you have in your code.

Macros in X++

2/18/2021 • 25 minutes to read • Edit Online

This topic describes how to create and use macros in X++.

Precompiler directives are processed before the code is compiled. The directives declare and handle macros and their values. The directives are removed by the precompiler so that the X++ compiler never encounters them. The X++ compiler only sees the sequence of characters written into the X++ code by the directives.

#define and #if directives

All precompiler directives and symbols begin with the # character. A macro can be defined at any point in the code. The variable can have a value that is a sequence of characters, but it is not required to have a value. The **#define** directive tells the precompiler to create the macro variable, including an optional value. The **#if** directive tests whether the variable is defined, and optionally, whether it has a specific value. The X++ precompiler directives, the macro names that they define, and the **#if** directive value tests are all case-insensitive. However, it is a best practice to begin macro names with an uppercase letter.

Code example

In the following code sample, a macro named **MyMacro** is defined. It is not given a value in the **#define.MyMacro** definition line. Therefore it behaves as if the value is a zero-length sequence of characters. The **#if.MyMacro** statement tests whether **MyMacro** is defined. Because **MyMacro** is defined, the lines of code before the first **#endif** are included in the X++ code at the test location. Near the end of the example there is an **#ifnot.MyMacro** test. Because **MyMacro** is defined, that test is false and no lines are written into the X++ code. After the precompile phase ends for this method, the **MyMacro** definition goes out of scope and is no longer known to the system.

```
static void SimpleDefineIfJob(Args _args)
{
    str sTest = "Initial value.";

    #define.MyMacro // MyMacro is now defined.
    #if.MyMacro
        sTest = "Yes, MyMacro is defined.";
        info(sTest);
    #endif
    // Notice the non-code sentence line causes no X++ compiler error,
    // because the X++ compiler never sees it.
    #ifnot.MyMacro
        The X++ compiler would reject this sentence.
        sTest = "No, MyMacro is not defined.";
        info(sTest);
    #endif
/********** Actual output
Message (03:46:20 pm)
Yes, MyMacro is defined.
**********/
}
```

Precompile and Compile Error Messages

When you are developing code that contains macros, you must understand whether an error message is generated during the precompile or the compile phase. The two key words to look for are:

- Lexical – This indicates a precompile error.
- Syntax – This indicates a compile error.

The following code example has a lexical error caused by the first closing parenthesis, which marks the end of the directive. Therefore the precompiler is confused by the last two characters, """;)".

```
#define.MyMacro1(info("Hello");)
```

The following code example has a syntax error caused by using the non-existent **++++** operator. The X++ compiler encounters this operator after **#MyMacro2** is replaced by the macro value. The macro definition is correct even though its value is not accepted X++ syntax.

```
#define.MyMacro2(++++iTest;)
#MyMacro2
```

#undef directive

You can use the **#undef** directive to remove a macro definition that exists from a previous **#define**. After a macro name has been created by **#define** and then removed by **#undef,** the macro can be created again by another **#define**. **#undef** has no effect on macros that are created by the **#localmacro** directive. In the following code sample, the macro **MyMacro** is undefined by using the #undef directive. The #undef occurs between the two #if tests for its existence. The output shows only the first #if test was true.

```
static void UndefMacroJob(Args _args)
{
    #define.MyMacro
    #if.MyMacro
        info("Macro is defined (1)");
    #endif
    #undef.MyMacro // Removes the macro.
    #if.MyMacro
        info("Macro is defined (2)");
    #endif
/************* Actual output
Message (10:19:15 am)
Macro is defined (1)
*************/
}
```

Use a Macro Value

You can define a macro name to have a value. A macro value is a sequence of characters. A macro value is not a string (or **str**) in the formal sense of a data type. You assign a value to a macro by appending the value enclosed in parentheses at the end of a #define directive. You can use the macro symbol where you want the value to occur in the X++ code. A macro symbol is the name of the macro with the # character added as a prefix. The following code sample shows a macro symbol **#MyMacro**. The symbol is replaced by the value of the macro.

```
static void MacroWithIntValueJob(Args _args)
{
    int iTest = 8;
    ;
    #define.MyMacro(32)
    // This next #define, which has no value for the macro name,
    // would not disrupt the value 32 set by the previous #define.
    //#define.MyMacro
    // This next #define, which has a different value than 32,
    // would overwrite the value 32 set by the previous #define.
    //#define.MyMacro(444)
    iTest = #MyMacro;
    info(int2str(iTest));
/**********  Actual output
Message (04:33:49 pm)
32
**********/
}
```

Test a Macro Value

You can test a macro to see whether it has a value. You can also test to see whether its value is equal to a specific sequence of characters. These tests enable you to conditionally include lines of code in your X++ program. There is no way you can test whether a defined macro has a value. You can only test whether a specific value matches the value of a macro. As a best practice, any macro name that you define should always have a value, or it should never have a value. When you alternate between these modes, your code becomes difficult to understand. For macros that have a value, you can vary the value when you see fit. In the following code sample, two **#if** tests are run to determine whether the macro **MyIntMacro** exists. The **#if.MyIntMacro()** test is true. This syntax behaves the same as **#if.MyIntMacro**.

```
static void TestMacroValue6Job(Args _args)
{
    #define.MyIntMacro(66)
    // #if tests.
    #if.MyIntMacro()
        info("A: " + int2str(#MyIntMacro));
    #endif
    #if.MyIntMacro(66)
        info("B: " + int2str(#MyIntMacro));
    #endif
    // #ifNOT tests.
    #ifNOT.MyIntMacro(7777)
        info("C: " + int2str(#MyIntMacro));
    #endif
    #ifNOT.No_Such_Macro_Name(66)
        info("D: " + int2str(#MyIntMacro));
    #endif
/****************  Actual output
Message (11:24:47 am)
A: 66
B: 66
C: 66
D: 66
****************/
}
```

The following code sample shows the **#if.DebugMacro(heavy)** directive that tests the value of the **DebugMacro** macro. If the value is the five character sequence **heavy**, then the test is true.

```
static void TestMacroSpecificValue8Job(Args _args)
{
    ;
    // Uncomment either one of these defines, or neither.
    //#define.DebugMacro(light) // This line for: light debugging.
    #define.DebugMacro(heavy) // This line for: heavy debugging.
    #if.DebugMacro
        info("Starting the job.");
    #endif
    #if.DebugMacro(heavy)
        info("UTC == "
            + DateTimeUtil ::toStr
                (DateTimeUtil::utcNow()
                )
            );
    #endif
    // Do something useful here.
/**********  Actual output
Message (01:58:12 pm)
Starting the job.
UTC == 2007-12-05T21:58:12
**********/
}
```

#defInc and #defDec directives

#defInc and **#defDec** are the only directives that interpret the value of a macro and they apply only to macros that have a value that can be converted to the formal **int** type. The value can only contain numerals. The only non-numeric character allowed is a leading negative sign (-). The integer value is treated as an X++ **int**, not as an **int64**. For macro names that are used by the **#defInc** directive, it is important that the **#define** directive that creates the macro not reside in a class declaration. The behavior of **#defInc** in these cases is unpredictable. Instead, such macros should be defined in only a method. We recommend that the **#defInc** and **#defDec** directives only be used for macros that have an integer value. The precompiler follows special rules for **#defInc** when the macro value is not an integer, or when the value is unusual or extreme. The following table lists the values that **#defInc** converts to zero (0) and then increments. When a value is converted to 0 by **#defInc,** the original value cannot be recovered, not even by **#defDec.**

MACRO VALUE	BEHAVIOR
(+55)	The positive sign (+) prefix makes the precompiler treat this as a non-numeric string. The precompiler treats all non-numeric strings as 0 when it handles a **#defInc** (or **#defDec)** directive.
("3")	Integers enclosed in quotation marks are treated as 0. The quotation marks are discarded, and these changes persist.
( )	A string of spaces is treated as 0, and then incremented.
()	A zero-length string is treated as 0, and then incremented, when the value is enclosed in parentheses, as in **#define.MyMac().**
(Random string.)	Any non-numeric string of characters is treated as 0, and then incremented.

MACRO VALUE	BEHAVIOR
(0x12)	Hexadecimal numbers are treated as non-numeric strings. Therefore they are converted to 0, and then incremented.
(-44)	Negative numbers are acceptable, including integers without the negative sign (-).
(2147483647)	The maximum positive **int** value is changed to the minimum negative **int** value by #defInc.
(999888777666555)	Any large number, beyond the capacity of **int** and **int64**. This is treated as the maximum positive **int** value.
(5.8)	Real numbers are truncated by #defDec (and #defInc). Subsequent symbol substitution shows that the truncation persists.
	When no value and no parentheses are provided for the directive #define.MyValuelessMacro, the precompiler rejects use of the directive #defInc.MyValuelessMacro.

Code example

In the following code sample, the initial value of the macro **CounterMacroA** is a string that can be converted into an integer. The sample shows how the **#defInc** and **#defDec** directives can be used for this macro name.

```
static void SimpleDefINCJob(Args _args)
{
    ;
    #define.CounterMacroA(1)
    #defInc.CounterMacroA
    info("mg11: # CounterMacroA == " + int2str(#CounterMacroA));
    #if.CounterMacroA(2)
        info("mg12: # if confirms CounterMacroA == 2");
    #endif
    #defDec.CounterMacroA
    info("mg23: # CounterMacroA == " + int2str(#CounterMacroA));
    #if.CounterMacroA(1)
        info("mg24: # if confirms CounterMacroA == 1");
    #endif
/**************  Actual Infolog output
Message (12:47:57 pm)
mg11: # CounterMacroA == 2
mg12: # if confirms CounterMacroA == 2
mg23: # CounterMacroA == 1
mg24: # if confirms CounterMacroA == 1
**************/
}
```

#globaldefine directive

The **#globaldefine** directive is similar to the **#define** directive. The difference is that **#define** directives generally take precedence over **#globalmacro** directives. This is true regardless of which directive occurs first in the X++ code. A **#globaldefine** never overwrites a **#define** directive that has both a macro name and a value. A **#globaldefine** can overwrite another **#globaldefine. **A **#define** directive that has only a name does not overwrite a #globalmacro that has both a name and a value. It is recommended that you use **#define,** and that you do not use **#globaldefine.** Use of **#globaldefine** can create uncertainty that makes code difficult to maintain. The exact semantics of **#globaldefine** cannot be achieved through **#if** test directives. By using **#if**

tests you can avoid overwriting a **#define** and a **#globaldefine.** But **#if** tests cannot distinguish between **#define** and **#globaldefine** macros. The following code sample is the closest you can come to achieving the **#globaldefine** semantic with other directives such as **#if.**

```
static void IfNotDefineNestGlobalJob (Args _args)
{
    ;
    #undef.MaybeMac
    #ifnot.MaybeMac
    #define.MaybeMac(4444) // Works.
    #endif
    #if.MaybeMac
        info(int2str(#MaybeMac));
    #endif
/**********************  Actual Infolog output
Message (07:43:32 pm)
4444
**********************/
}
```

Code example

The following code sample shows a difference in the behavior of **#define** and **#globaldefine**. Following the code sample is a table explaining the conclusions from the output. The primary test case in the code sample is labeled **12**.

```
static void InteractDefineGlobalJob(Args _args)
{
    ;
    // Pairs of #define - #globaldefine directives (same macro names).
    #define.11_DEFINEvalue_GLOBALnoval("11__DEFINE.")
    #globaldefine.11_DEFINEvalue_GLOBALnoval
    #define.12_DEFINEnoval_GLOBALvalue
    #globaldefine.12_DEFINEnoval_GLOBALvalue("12_GLOBAL.")
    #define.13_DEFINEvalue_GLOBALvalue("13__DEFINE.")
    #globaldefine.13_DEFINEvalue_GLOBALvalue("13_GLOBAL.")
    // Pairs of #globaldefine - #define directives.
    #globaldefine.27_GLOBALvalue_DEFINEnoval("27_GLOBAL.")
    #define.27_GLOBALvalue_DEFINEnoval
    #globaldefine.28_GLOBALnoval_DEFINEvalue
    #define.28_GLOBALnoval_DEFINEvalue("28__DEFINE.")
    #globaldefine.29_GLOBALvalue_DEFINEvalue("29_GLOBAL.")
    #define.29_GLOBALvalue_DEFINEvalue("29__DEFINE.")
    // Pairs of same directive types.
    #define.50_DEFINEvalue_DEFINEnoval("50__DEFINE 1.")
    #define.50_DEFINEvalue_DEFINEnoval
    #globaldefine.64_GLOBALvalue_GLOBALnoval("64_GLOBAL 1.")
    #globaldefine.64_GLOBALvalue_GLOBALnoval
    #globaldefine.65_GLOBALnoval_GLOBALvalue
    #globaldefine.65_GLOBALnoval_GLOBALvalue("65_GLOBAL 2.")
    #globaldefine.66_GLOBALvalue_GLOBALvalue("66_GLOBAL 1.")
    #globaldefine.66_GLOBALvalue_GLOBALvalue("66_GLOBAL 2.")
    // Infolog outputs.
    info(#11_DEFINEvalue_GLOBALnoval);
    info(#12_DEFINEnoval_GLOBALvalue);
    info(#13_DEFINEvalue_GLOBALvalue);
    info(#27_GLOBALvalue_DEFINEnoval);
    info(#28_GLOBALnoval_DEFINEvalue);
    info(#29_GLOBALvalue_DEFINEvalue);
    info(#50_DEFINEvalue_DEFINEnoval);
    info(#64_GLOBALvalue_GLOBALnoval);
    info(#65_GLOBALnoval_GLOBALvalue);
    info(#66_GLOBALvalue_GLOBALvalue);
/*****************************  Actual Infolog output
Message (09:31:26 pm)
11__DEFINE.
12_GLOBAL. Shows that #globaldefine overwrites #define when #globaldefine is giving a value to an existing
macro that has no value. Test cases 13 and 29 are more common and realistic.
13__DEFINE. Shows that #define usually takes precedence over #globaldefine, regardless of which directive
occurs first in the X++ code. Test case 12 shows this is not always true.
27_GLOBAL.
28__DEFINE.
29__DEFINE. Shows that #define usually takes precedence over #globaldefine, regardless of which directive
occurs first in the X++ code. Test case 12 shows this is not always true.
50__DEFINE 1. Shows that a #globaldefine that has a name but no value does not overwrite a #globaldefine
that has both a name and a value. The same is true between a pair of #define directives. This resembles test
case 12.
64_GLOBAL 1. Shows that a #globaldefine that has a name but no value does not overwrite a #globaldefine that
has both a name and a value. The same is true between a pair of #define directives. This resembles test case
12.
65_GLOBAL 2. Shows that a #globaldefine that has both a name and a value overwrites a previous
#globaldefine.
66_GLOBAL 2. Shows that a #globaldefine that has both a name and a value overwrites a previous
#globaldefine.
*****************************/
}
```

Macro parameters

You can define macro values to include parameter symbols. The first parameter symbol is %1, the second is %2, and so on. You pass values for the parameters when you reference the macro symbol name for expansion.

Macro parameter values are character sequences of no formal type, and they are comma delimited. There is no way to pass in a comma as part of a parameter value. The number of parameters passed can be less than, greater than, or equal to the number of parameters that the macro value is designed to receive. The system tolerates mismatches in the number of parameters passed. If fewer parameters are passed than the macro expects, each omitted parameter is treated as a zero-length sequence of characters. In the following code sample, **MyMacro** is defined to have a value that contains parameters. Macro substitution symbols are given with parameter values in parentheses.

```
static void MacroParameterSubstitutionJob(Args _args)
{
    ;
    #define.MyMacro("One == [%1] ,  Two == [%2]")
    // Make parameter substitutions:
    info("AA: " + #MyMacro(Apple));
    info("BB: " + #MyMacro(Apple,Banana));
    info("CC: " + #MyMacro(Apple, Banana, cranberry));
    info("DD: " + #MyMacro(,Apple,Banana));
    info("EE: " + #MyMacro());
    info("FF: " + #MyMacro);
/************  Actual Infolog output
Message (03:22:07 pm)
AA: One == [Apple] ,  Two == []
BB: One == [Apple] ,  Two == [Banana]
CC: One == [Apple] ,  Two == [ Banana]
DD: One == [] ,  Two == [Apple]
EE: One == [] ,  Two == []
FF: One == [] ,  Two == []
************/
}
```

#localmacro and #globalmacro directives

The **#localmacro** directive is a good choice when you want a macro to have a value that is several lines long, or when your macro value contains a closing parenthesis. The **#localmacro** directive is a good choice when you want your macro value to be lines of X++ or SQL code. The **#localmacro** directive can be written as **#macro.** However, **#localmacro** is the recommended term. Both macros have the same behavior. By using the **#if** directive, you can test whether a macro name is declared with the **#define** directive. However, you cannot test whether the macro name is declared with the **#localmacro** directive. Only macros declared by using the **#define** directive are affected by the **#undef** directive. In a **#define** directive, you can specify a name that is already in scope as a **#localmacro.** The effect is to discard the **#localmacro** and create a **#define** macro. This also applies to the opposite sequence, which means that a **#localmacro** can redefine a **#define.** A **#localmacro** (that has both a macro name and a value) always overrides a previous **#localmacro** that has the same name. However, you cannot always be sure whether the override occurs when you use **#globalmacro.** For this reason we recommend that you do not use **#globalmacro. **This same problem occurs with **#globaldefine.** The main difference between a **#define** macro and a **#localmacro** macro is in how their syntax is terminated. The terminators are as follows:

- **#define** – is terminated by– **)**
- **#localmacro** – is terminated by– **#endmacro**

#localmacro is a better choice for macros with multiple line values. Multiple line values are typically lines of X++ or SQL code. X++ and SQL contain lots of parentheses, and these would prematurely terminate a #define. Both **#define** and **#localmacro** can be declared and terminated on either a single line or on subsequent lines. In practice, the **#define** is terminated on the same line that it is declared on. In practice, the **#localmacro** is terminated on a subsequent line. Where both macro names and values are supplied, the **#globalmacro** directive cannot override the #define directive. Also, the **#globaldefine** directive cannot override the **#localmacro** directive.

Code examples

The following code sample shows how to use the **#localmacro** directive. It demonstrates that the **#undef** directive does not affect **#localmacro** macros. It also shows that #if tests cannot determine whether a **#localmacro** macro has been defined.

```
static void LocalMacroJob(Args _args)
{
    ;
    #localmacro.LMacReportLog
        print("%1  --LM, print.");
        info("%1  --LM, Infolog.");
    #endmacro
    #LMacReportLog(g11: Hello World )
    #if.LMacReportLog
        info("The # IF LMacReportLog is true");
    #endif
    #undef.LMacReportLog
    #LMacReportLog(g22: Greetings World)
    #localmacro.LMacReportLog
    #endmacro // No lines for value before this end.
    #LMacReportLog(g33: Bye Bye) // Not present in the output.
/*************  Actual Infolog output
Message (03:10:17 pm)
g11: Hello World   --LM, Infolog.
g22: Greetings World  --LM, Infolog.
*************/
}
```

The following X++ code sample shows that **#localmacro** overrides a **#globalmacro** of the same macro name, but that **#globalmacro** does not override **#localmacro**.

```
static void GlobalMacroNotOverrideJob(Args _args)
{
    ;
    //---------  LGMa ,  L then G  ---------
    #localmacro.LGMa
        info("LGMa: Loc 11");
    #endmacro
    #globalmacro.LGMa
        info("LGMa: Glob 12");
    #endmacro
    #LGMa
    //---------  LGMb ,  G then L  ---------
    #globalmacro.LGMb
        info("LGMb: Glob 24");
    #endmacro
    #localmacro.LGMb
        info("LGMb: Loc 25");
    #endmacro
    #LGMb
/****************  Actual Infolog output
Message (06:39:42 am)
LGMa: Loc 11
LGMb: Loc 25
***************/
}
```

Nesting Macro Symbols

You can nest precompiler definition directives inside an outer definition directive. The main definition directives are **#define** and **#localmacro. **The cases for which this topic provides code samples are as follows:

- Transitive substitution: A **#define** macro can have the symbol for another macro as its value. Transitive substitution of the symbol occurs in X++ code.
- No transitive substitution: An **#if** directive test of a macro value does not perform substitutions.
- Macro within a macro: A #define directive can be given inside a **#localmacro** directive, or a **#localmacro** can be inside a **#define**.

Transitive Substitution

In the following code sample, the value of the first **#define** variable includes a symbol (#D) of the second **#define** variable. This works even though the expansion symbol #D occurs before macro **D** is defined.

```
static void NestMacroJobA1(Args _args)
{
    ;
    #define.Cd("Cd +: # D == " + #D)
    // If not commented out, this next code line would cause the
    // error message "The macro does not exist.", because
    // #D in the value of #Cd cannot be expanded before it is defined.
    //info(#Cd);
    #define.D("D")
    info(#Cd);
/************  Actual Infolog output
Message (10:42:13 am)
Cd +: # D == D
************/
}
```

No transitive substitution

The following code sample tries to determine whether two macro variables have the same value, without specifying what that value might be. The output shows that this determination cannot made..

```
static void NestMacroJobA2(Args _args)
{
    ;
    #define.A1(5)
    #define.A2(5)
    info("Status: A1==" + int2Str(#A1) + " , A2==" + int2Str(#A2));
    #if.A1(#A2)
        info("Yes, symbol substitution does work on # IF test.  Unexpected.");
    #endif
    #ifNOT.A1(#A2)
        info("No, symbol substitution does not work on # IF test.");
    #endif
/************  Actual Infolog output
Message (11:27:38 am)
Status: A1==5 , A2==5
No, symbol substitution does not work on # IF test.
************/
}
```

The following code sample shows that the #defInc directive does not lead to transitive substitution of symbol values. For more information about the #defInc directive, see How to: Use the #defInc and #defDec Directives. After the #defInc.E2 directive, the subsequent output value for #E2 shows the value for **E2** is converted to zero (0) by #defInc.E2 before it is incremented to one (1). Before the conversion, the value of **E2** was the three characters #E2. The output for test case **36** shows the value has been converted to 1.

```
static void NestMacroJobA4(Args _args)
{
    ;
    #define.E1(5)
    #define.E2(#E1)
    info("11:  # E1 == " + int2Str(#E1));
    info("12:  # E2 == " + int2Str(#E2));
    #defInc.E1
    info(" -------");
    info("23: After Inc.E1,  # E1 == " + int2Str(#E1));
    info("24: After Inc.E1,  # E2 == " + int2Str(#E2));
    #defInc.E2
    info(" -------");
    info("35: After Inc.E2,  # E1 == " + int2Str(#E1));
    info("36: After Inc.E2,  # E2 == " + int2Str(#E2));
/************  Actual Infolog output
Message (02:39:41 pm)
11:  # E1 == 5
12:  # E2 == 5
-------
23: After Inc.E1,  # E1 == 6
24: After Inc.E1,  # E2 == 6
-------
35: After Inc.E2,  # E1 == 6
36: After Inc.E2,  # E2 == 1
************/
}
```

Macro within a macro

A **#define** directive can be given inside a **#localmacro** directive, and a **#localmacro** can be inside a #define. This is shown in the following code sample.

```
static void NestMacroJobB5(Args _args)
{
    int iTest = 31;
    ;
    //-------------  J  ---------------
    #localmacro.LocMacOuterJL
        #define.DefinInnerJD(5)
    #endmacro
    #LocMacOuterJL
    info("J: Directive nesting works if 5 appears: # DefinInnerJD == "
            + int2Str(#DefinInnerJD));
    //-------------  K  ---------------
    #define.DefinOuterKD(#localmacro.LocMacInnerKL ++iTest; #endmacro)
    ++iTest; // Result is 32.
    #DefinOuterKD
    #LocMacInnerKL
    info("K: Directive nesting works if 33 appears: iTest == "
            + int2Str(iTest));
/**************  Actual Infolog output
Message (11:21:02 am)
J: Directive nesting works if 5 appears: # DefinInnerJD == 5
K: Directive nesting works if 33 appears: iTest == 33
**************/
}
```

#macrolib directive

In the Application Explorer under the Macros node, there are many library nodes that contain sets of macro directives. Both **#define** and **#localmacro** often appear in the contents of these macro libraries. You can use the **#macrolib.MyAOTMacroLibrary** to include the contents of a macro library in your X++ code. The **#if** and

#undef directives do not apply to **#macrolib** names. However, they do apply to **#define** directives that are the contents of a **#macrolib** macro. The directive **#macrolib.MyAOTMacroLibrary** can also be written as **#MyAOTMacroLibrary.** The **#macrolib** prefix is recommended because it is never ambiguous to a person who later reads the code.

Create a macro library

To create a macro library:

1. In **Solution Explorer,** right-click on the project, select **Add** and then **New item.**
2. In the **Add New Item** dialog, select Installed and then **AX Artifacts** in the left pane.
3. In the middle pane, select **Macro.**
4. Enter a name and click **Add.** Save the macro file and refresh the Application Explorer to find your macro library.

Code examples

There is a macro library that is named **Event.** This macro library contains the directive **#define.DefaultEventPollFrequency(15)**. The following code sample shows that the **#macrolib.Event** directive makes the macro **#DefaultEventPollFrequency** available.

```
static void SystemProvidedMacroLibraryJob(Args _args)
{
    ;
    #macrolib.Event // Contains: #define.DefaultEventPollFrequency(15)
    info("# DefaultEventPollFrequency == "
            + int2str(#DefaultEventPollFrequency));
/***************  Actual Infolog output
Message (06:31:26 pm)
# DefaultEventPollFrequency == 15
***************/
}
```

The following code example shows what happens when you write a **#define** for a name that is already the name of a node in the macro library. For this example, there is a node named **MacLib23,** and its contents are one **#define** as follows:

```
#define.DefinInMacLib23("_This is inside AOT macrolib MacLib23.")
```

After a #macrolib directive is issued for **MacLib23**, **#define** and **#undef** directives have no effect on the **#macrolib** macro (see output _BB). However, a **#define** in the contents of a **#macrolib** macro can be overwritten by a subsequent #define or **#undef** in the code (see output _DD).

```
static void PrecedenceMacrolibDefineJob(Args _args)
{
    ;
    #define.MacLib23("_11:  Plain #define value for MacLib23, same name as the AOT macrolib macro.")
    info("_AA: " + #MacLib23);
    #macrolib.MacLib23
    info("_BB: " + #DefinInMacLib23); // Defined inside the macrolib macro.
    info("_CC: " + #MacLib23); // Output shows plain #define, not the macrolib macro contents.
    #define.DefinInMacLib23("_33:  Plain #define in the job code, overwrite of same macro name defined
inside the macrolib macro.")
    info("_DD: " + #DefinInMacLib23);
/**************************  Actual Infolog output
Message (10:53:13 am)
_AA: 11:  Plain #define value for MacLib23, same name as the AOT macrolib macro.
_BB: This is inside AOT macrolib MacLib23.
_CC: 11:  Plain #define value for MacLib23, same name as the AOT macrolib macro.
_DD: 33:  Plain #define in the job code, overwrite of same macro name defined inside the macrolib macro.
**************************/
}
```

#linenumber Directive

You can use the #linenumber directive during your development and debugging of code. It is replaced by the physical line number in the code file.

Code example

The following X++ code sample shows the behavior of the **#linenumber** directive.

```
static void LinenumberPhysicalJob(Args _args)
{
    ;
    #define.Debug(light)
    #if.Debug
        info("Physical Line 8: # linenumber == "
            + int2Str(#linenumber));
    #endif
/******************  Actual Infolog output
Message (08:55:26 pm)
Physical Line 8: # linenumber == 8
******************/
}
```

Range (scope) of macros

The range in which a macro can be referenced depends on where the macro is defined. In a class, macros that are defined in the parent class can be referenced, but macros defined in a child class cannot be referenced. When the precompiler handles a child class, the precompiler first traces the inheritance chain to the most ascendant class. The precompiler processes all the directives from the class declaration part of the ascendant class. It stores all the macros and their values in its internal tables. The precompiler handles the next class in the inheritance chain the same way. The result of the directives in each class declaration are applied to the internal tables that are already populated from directives that were found earlier in the inheritance chain. When the precompiler reaches the target child class, it again handles the class declaration part. However, it next handles each method in a series of separate operations. The precompiler updates its internal tables in a way that the state of the tables can be restored as they were before processing of the current method began. After the first method is handled, the internal tables are restored before the next method is handled.

The Method is All Contents of the Node

In this context, a method is defined as the contents of a method node in the Application Object Tree (AOT). In the

AOT, you can expand the Classes node, expand a class node, right-click a method node, and then select Edit. Then you can add a line for #define.MyMacro("abc") before the method declaration. The precompiler treats this #define directive as part of the method, even though the #define occurs outside the {} block of the method.

Class Inheritance and Macro Reference Range

The following code example demonstrates the range of macro referencing in class inheritance scenarios. The primary line to notice in the method's output is the line labeled ClassC_h. It shows that a macro defined in a grandparent class can be referenced in a method of the grandchild class. Another important line in the output is labeled ClassA_k. This line shows that a macro defined in a method is not available in other methods. ClassA is the base class and it defines several macros in its class declaration. Its descendant classes reference these macros. The base class also defines a macro inside one of its methods. A second method in this class determines the macro is defined out of range and cannot be referenced in the second method. The **#undef.MacroRange333** in the method **UseOtherMethodMacro** affects the availability of macro **MacroRange333** in the rest of that method. Descendant classes can still reference **MacroRange333**. ClassB extends ClassA and it undefines the macro **MacroRangeA** that is defined in its parent class. This makes the macro unavailable to any class that extends the present class. The present class also redefines the macro **MacroRangeB** that is defined in its parent class. This changes the value of the macro (from positive to negative). ClassC extends ClassB and it uses **#ifnot** to demonstrate that it cannot access the **MacroRangeA** macro that the base class, ClassInheritanceOfMacrosCBase1, defines. The reason is that the mid-level class undefined the macro. This class also demonstrates that it can access the macro **MacroRange333** that ClassInheritanceOfMacrosCBase1 class defines. TestClass contains a method that calls the demonstration methods and displays the results.

```
class ClassA
{
    // Unless disturbed by other directives, these macros can
    // be referenced by method in this class in child classes.
    #define.MacroRangeA
    #define.MacroRangeB(22)
    #define.MacroRange333(333)

    static void UseMacros()
    {
        //  This method shows that a macro that is defined
        // in the class declaration is in range in every
        // method in that class.
        // This method also contains a define for macro
        // MacroDefInMethodD, and tests outside this
        // method determine the macro is not in range.
        ;
        info("ClassA: #MacroRangeB == " + int2str(#MacroRangeB));
        #define.MacroDefInMethodD // Cannot be referenced in other methods.
    }

    static void UseOtherMethodMacro()
    {
        // This method shows that the macro MacroDefInMethodD
        // that was defined in another method in this class
        // cannot be referenced in this method.
        // This method contains an #undef of MacroRange333,
        // yet descendant classes can still reference this macro.
        #ifnot.MacroDefInMethodD
        info("ClassA_k: This means MacroDefInMethodD is in not range here.");
        #endif
        #undef.MacroRange333
    }
}


class ClassB extends ClassA
{
    // This class declaration makes the macro MacroRangeA
```

```
    // This class declaration makes the macro MacroRangeA
    // unavailable to methods in this class and child classes.
    // This class declaration also redefines MacroRangeB for
    // this class and child classes.
    #undef.MacroRangeA // Makes unavailable to child classes.
    #define.MacroRangeB(-22)
    // Redefining with a different value.
    static void UseMacros()
    {
        // This method shows that the value for #MacroRangeB comes
        // from its redefinition in this class, instead of from
        // the definition in the parent class.

        info("ClassB_c: #MacroRangeB == " + int2str(#MacroRangeB)
            + " (Is now negative due to later redefinition.)");
    }
}


class ClassC extends ClassB
{
    static void UseMacros()
    {
        //   This method shows that the #undef in the parent
        // class overwrites the #define in the grandparent class.
        //   This method also shows that a macro defined in the
        // grandparent class is in range in methods on this class.
        ;
        #ifnot.MacroRangeA
        info("ClassC_f: MacroRangeA is no longer defined, due to #undef in ClassB.");
        #endif
        info("ClassC_h: #MacroRange333 == " + int2str(#MacroRange333)
            + " (Defined in ClassA.)");
    }
}


class TestClass
{
    static void JobClassesCC(Args _args)
    {
        ;
        ClassA ::UseMacros();
        ClassA ::UseOtherMethodMacro();
        ClassB ::UseMacros();
        ClassC ::UseMacros();
        /***************  Actual output
        Message (08:10:59 am)
        ClassA_a: #MacroRangeB == 22
        ClassA_k: This means MacroDefInMethodD is not in range here.
        ClassB_c: #MacroRangeB == -22 (Is now negative due to later redefinition.)
        ClassC_f: MacroRangeA is no longer defined, due to #undef in child class ClassB.
        ClassC_h: #MacroRange333 == 333 (Defined in ClassA.)
        ***************/
    }
}
```

X++ attribute classes

2/18/2021 • 4 minutes to read • Edit Online

This topic describes the use of attributes in X++.

An attribute is a non-abstract class that extends (inherits from) the **SysAttribute** class. Attributes represent or store metadata about types and methods. An attribute can be attached to a class, an interface, or a method of a class, interface, or table.

Creating an attribute class

An attribute class can extend the **SysAttribute** class directly, or it can extend any descendant of the **SysAttribute** class. The **SysAttribute** class cannot be used as an attribute because it is declared **abstract**. The following example shows the declaration and design of an ordinary attribute class that you could create.

```
public class PracticeAttribute extends SysAttribute
{
    // Fields in the classDeclaration.
    StartEnd startEndEnum;
    str reason;
    // Constructor.
    public void new(StartEnd _startEndEnum, str _reason)
    {
        startEndEnum  = _startEndEnum;
        reason = _reason;
    }
    // Other methods can go here.
}
```

Decorating a class with an attribute

The following example shows a class and a method that are decorated with the **PracticeAttribute** given in the previous example. If the constructor of the attribute takes no parameters, the parentheses for the parameters are optional. The attribute decoration could be `[AnotherAttribute]` without parentheses.

```
[PracticeAttribute(StartEnd::End, "Use the RegularClass class at the end.")]
public class RegularClass
{
    [PracticeAttribute(Startend::Start, "Use the rehearse method at the start.")]
    public int rehearse()
    {
        // Logic goes here.
    }
    // More fields and methods belong here.
}
```

Attribute constructors

You can enable your attribute class to store tailored metadata each time it is used to decorate a class, by having its constructor take parameters. The parameters for the constructor must be literals of the primitive types, such as **int**, **enum**, or **str**. The compiler does not construct an instance of the attribute class. It stores the name of the attribute class, plus the literal values for its constructor. Therefore, if the logic in an attribute constructor would throw an exception, the exception would not be found by decorating a class with the attribute. The exception would be found later when a process looks at a class to see the attribute it is decorated with. That is when the attribute is constructed.

Naming conventions

All attribute classes have the suffix **Attribute** in their name. The **Attribute** suffix is the name convention that we recommend, but it is not a system requirement. You can determine whether a class **extends** directly from **SysAttribute** by selecting the class in the **Application Explorer** and reviewing the **Extends** property in the **Properties** window.

SysObsoleteAttribute

The system provides several attributes, including the **SysObsoleteAttribute** class. One use of the **SysObsoleteAttribute** class is to notify the compiler that the compile should fail if a particular method is called in the source code. The compiler rejects the compile, and displays the specific message that is stored in this use of the attribute. The **SysObsoleteAttribute** class can also be used to notify the compiler to issue warning messages instead of errors.

SysObsoleteAttribute code example

```
[SysObsoleteAttribute("The Automobile class might have faster performance.", false)]
class Bicycle
{
    // Members of the Bicycle class go here.
}
```

Metadata reflection

You use reflection to find the attribute metadata that is attached to a class. The classes to use for attribute reflection are as follows:

- **DictClass** class – For classes and interfaces.
- **DictMethod** class – For methods on classes, interfaces, or tables.

On the previous reflection classes, the methods for reflecting on attribute metadata are as follows:

- **getAllAttributes** method
- **getAttribute** method
- **getAttributedClasses** method
- **getAttributes** method

> **NOTE**
>
> There is no mechanism for listing all methods or classes that are adorned with a particular attribute from X++ code. However, because the X++ compiler records this information in the cross reference database, the information can be mined from there.

Metadata reflection code example

You use the **DictMethod** class to find the metadata value of an attribute that is decoration on a method. The following code example uses the **SysEntryPointAttribute** class as the attribute. It accepts your parameter values for the method name, and for the name of the class that contains the method. The **parmChecked** method is particular to the **SysEntryPointAttribute** class, and it is not inherited from its base class **SysAttribute**. Each attribute class can have its own method name for its metadata.

```
static public int MetadataOfSysEntryPointAttributeOnMethod
        (
        str _sNameOfClass,
        str _sNameOfMethod
        )
{
    // Return Values:
    // 0 == Has the attribute, its metadata value is false;
    // 1 == Has the attribute, its metadata value is true;
    // 2 == The method lacks the SysEntryPointAttribute.
    int nReturnValue = -1,
        nClassId;
    boolean boolParmChecked;
    DictMethod dm;
    Object attributeAsObject;
    SysEntryPointAttribute sepAttribute;
    Global::info("Starting AttributeReflection"
        + " ::MetadataOfSysEntryPointAttributeOnMethod ....");
    Global::info(strFmt
        ("Parameters are: _sNameOfClass = %1 ,  _sNameOfMethod = %2 .",
        _sNameOfClass, _sNameOfMethod)
        );
    nClassId = Global::className2Id(_sNameOfClass);
    dm = new DictMethod
        (UtilElementType::ClassInstanceMethod,
        nClassId,
        _sNameOfMethod
        );
    attributeAsObject = dm.getAttribute("SysEntryPointAttribute");
    if (attributeAsObject is SysEntryPointAttribute)
    {
        sepAttribute = attributeAsObject as SysEntryPointAttribute;
        boolParmChecked = sepAttribute.parmChecked();
        if (boolParmChecked)
            nReturnValue = 1;
        else
            nReturnValue = 0;
        Global::info(
            strFmt("Return value is %1.",
                nReturnValue)
            );
    }
    else
    {
        nReturnValue = 2;
        Global::error("Object is not a SysEntryPointAttribute??");
    }
    return nReturnValue;
}
/*** Output displayed in the Infolog.
Message (05:03:22 pm)
Starting AttributeReflection ::MetadataOfSysEntryPointAttributeOnMethod ....
Parameters are: _sNameOfClass = CustCustomerService ,  _sNameOfMethod = create .
Return value is 1.
***/
/**************
// Simple AOT > Jobs job to run the method.
static void AttributeReflection33Job(Args _args)
{
    AttributeReflection::MetadataOfSysEntryPointAttributeOnMethod
        ("CustCustomerService", "create");
}
**************/
```

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

X++ and C# comparison

2/18/2021 • 62 minutes to read • Edit Online

This topic compares X++ and C# syntax and programming.

X++, C# Comparison: Hello World

This section compares the simplest X++ program to its counterpart in C#.

X++ to C# Comparisons

The following sections describe some basic similarities and differences between X++ and C#.

Similarities

The following X++ features are the same for C#:

- Single line (`//`) and multi-line (`/* */`) comments.
- `==` (equal) operator for determining whether two values are equal.
- `!=` (not equal to) operator for determining whether two values are not equivalent.
- `+` (plus sign) operator for string concatenation.

Differences

The following table lists X++ features that are different in C#.

FEATURE	X++	C#	COMMENTS
`if` and `else` conditional statements	The `if` statement accepts any type of expression that it can automatically convert to a Boolean. Common examples include an `int` for which 0 means false, or an object for which null means false.	The `if` statement requires a Boolean expression.	The syntax structure regarding curly braces and parentheses is exactly the same between X++ and C#.
Literal string	A literal string can be delimited by either of the following: • A pair of double quotation mark (") characters. • A pair of single quotation mark (') characters.	A literal string must be delimited by a pair of double quotation mark (") characters.	For X++, the double quotation mark characters are usually used to delimit strings. However, it is convenient delimit a string with single quotation mark characters when your string must contain a double quotation mark character.
char `type`	There is no `char` or character type in X++. You can declare a `str` of length one, but it is still a string: `str 1 myString = "a";`	There is a `char` in C#. You cannot pass a `char` as the parameter to a method that inputs a `string` parameter, although you can first explicitly convert the `char` to a `string`.	For more information about X++ data types, see Primitive Data Types.

FEATURE	X++	C#	COMMENTS

Output of messages	X++ delivers messages to the user in the Infolog window. Common methods include the following:
	• The **print** statement:
	• static methods on the `Global` class:
	  ◦ Global::info
	  ◦ Global::warning
	  ◦ Global::error
		For a command line program, messages can be delivered to the console. Common methods include the following:
		• `Console.Out.WriteLine`
		• `Console.Error.WriteLine`	

X++ and C++ Samples

This section contains two simple code samples. One sample is written in X++, and the other is in C#. Both samples achieve the same result. The following X++ features are demonstrated:

- `//` single line comment
- `/\*` `\*/` multi-line comment
- `if` statement
- `==` operator
- `!=` operator
- `+` operator to concatenate strings
- Global::info for message output, with and without the Global:: prefix
- Global::error for message output
- The use of single and double quotation characters (' and ") as string delimiters.

> **NOTE**
> The best practice is to use double quotation marks for any string that might be displayed to the user.

X++ Sample

This X++ code sample is in the form of a job. There is a node titled Jobs in the Application Object Tree (AOT). This sample can be added under the Jobs node, and then the job can be run.

```
static void JobRs001a_HelloWorld(Args _args)
{
    if (1 == 1)
    {
        // These two info() calls are identical to the X++ compiler.
        // The second form is the one typically used in X++.
        Global::info("Hello World, 1.");
        info('Hello World, 2.');
    }
    if (1 != 1)
    {
        error("This message will not appear.");
    }
    else
    {
        // These two methods are also from the Global class.
        // The + operator concatenates two strings.
        warning("This is like info, but is for warnings, 3.");
        error("This is like info, but is for errors, 4.");
    }
}
```

Output

Here is the output from the Infolog window: Message (09:49:48) Hello World, 1. Hello World, 2. This is like info, but is for warnings, 3. This is like info, but is for errors, 4.

C# Sample

The following C# program is a rewrite of the previous X++ program.

```
using System;
class Pgm_CSharp
{
    static void Main( string[] args )
    {
        new Pgm_CSharp().Rs001a_CSharp_HelloWorld();
    }
    void Rs001a_CSharp_HelloWorld()
    {
        if (1 == 1)
        {
            Console .Out .WriteLine("Hello World, Explicit .Out , 1.");
            Console .WriteLine("Hello World, Implicit default to .Out , 2.");
        }
        if (1 != 1)
        {
            Console .Error .WriteLine("This message will not appear.");
        }
        else
        {
            Console .Error .WriteLine(".Error is like .Out, but can be for warnings, 3.");
            Console .Error .WriteLine(".Error is like .Out, but is for errors, 4.");
        }
    }
}
```

Output

Here is the actual output to the C# console:

```
Hello World, Explicit .Out, 1.
Hello World, Implicit default to .Out, 2.
.Error is like .Out, but can be for warnings, 3.
.Error is like .Out, but is for errors, 4.
```

X++, C# Comparison: Loops

This section compares the loop features between X++ and C#.

Similarities

The following features are the same in X++ and C#:

- Declarations for variables of the int primitive data type. Declarations for other primitive types are almost the same, but the types might have different names.
- while statement for loops.
- break statement to exit a loop.
- continue statement to jump up to the top of a loop.
- <= (less than or equal) comparison operator.

Differences

The following table lists X++ features that are different in C#.

FEATURES	X++	C#	COMMENTS
The `for` statement.	The for statement is available for loops.	The C# `for` statement is slightly different from `for` in X++.	In C# you can declare the counter integer in the `for` statement. But in X++ the counter must be declared outside the `for` statement.

++ increment operator.	An ++ increment operator is available in X++. But an **int** variable that is decorated with ++ can only be used as a statement, not as an expression. For example, the following lines of X++ code would not compile:
	`int age=42;`
	`print age++;`
	However, the following lines of X++ code would compile:
	`int age=42;`
	`age++; print age;`
		The C# ++ operator is more flexible than in X++.	The following lines of code are the same in both languages:
			• ++ myInteger;
			• myInteger++;
			But the following lines of code have a different effect from each other, and are valid only in C#:
			• yourInt = ++myInt;
			• yourInt = myInt++;

modulo operator.	In X++ the modulo operator is mod.	In C# the modulo operator is %.	The symbols for the modulo operator are different, but their behavior is the same in both languages.
Temporarily suspend a console program that has already begun.	The `pause` statement.	In C#, a command line program can be paused by the following line of code: `Console.In.Read();`	In X++ you continue by clicking an OK button on a modal dialog box. In C# you continue by pressing any keyboard on the keyboard.

FEATURES	X++	C#	COMMENTS
Display a message.	In X++, the `print` statement displays a message in the Print window.	In C# a message can be displayed on the console by the following line of code: `Console.WriteLine();`	The X++ `print` function is used only when you test. An X++ program that uses `print` almost always uses the `pause` statement somewhere later in the code. For production X++ code, use the Global::info Method instead of `print`. The `strfmt` function is often used together with `info`. There is no reason to use `pause` after `info`.
Make a sound.	The beep function makes a sound that you can hear.	In C# a sound that you can hear is issued by the following line of code: `Console.Beep();`	The statements each produce a short tone.

Print and Global::info

The X++ code samples for loops use the `print` function to display results. In X++ you can use the `print` statement can display any primitive data type without having to call functions that convert it to a string first. This makes `print` useful in quick test situations. Generally the Global::info method is used more often than `print`. The `info` method can only display strings. Therefore the strfmt function is often used together with `info`. A limitation of `print` is that you cannot copy the contents of the Print window to the clipboard (such as with Ctrl+C). Global::info writes to the Infolog window which does support copy to the clipboard.

Example 1: The while Loop

The **while** keyword supports looping in both X++ and C#.

X++ Sample of while

```
static void JobRs002a_LoopsWhile(Args _args)
{
    int nLoops = 1;
    while (nLoops <= 88)
    {
        print nLoops;
        pause;
        // The X++ modulo operator is mod.
        if ((nLoops mod 4) == 0)
        {
            break;
        }
        ++ nLoops;
    }
    beep(); // Function.
    pause; // X++ keyword.
}
```

Output

The output in the X++ Print window is as follows:

```
1
2
3
4
```

C# Sample of while

```
using System;
public class Pgm_CSharp
{
    static void Main( string[] args )
    {
        new Pgm_CSharp().WhileLoops();
    }

    void WhileLoops()
    {
        int nLoops = 1;
        while (nLoops <= 88)
        {
            Console.Out.WriteLine(nLoops.ToString());
            Console.Out.WriteLine("(Press any key to resume.)");
            // Paused until user presses a key.
            Console.In.Read();
            if ((nLoops % 4) == 0) {
                break;
            }
            ++ nLoops;
        }
        Console.Beep();
        Console.In.Read();
    }
}
```

Output

The console output from the C# program is as follows:

```
1
(Press any key to resume.)
2
(Press any key to resume.)
3
(Press any key to resume.)
4
(Press any key to resume.)
```

Example 2: The for Loop

The **for** keyword supports looping in both X++ and C#.

X++ Sample of for

In X++ the counter variable cannot be declared as part of the **for** statement.

```
static void JobRs002a_LoopsWhileFor(Args _args)
{
    int ii; // The counter.
    for (ii=1; ii < 5; ii++)
    {
        print ii;
        pause;
        // You must click the OK button to proceed beyond a pause statement.
        // ii is always less than 99.
        if (ii < 99)
        {
            continue;
        }
        print "This message never appears.";
    }
    pause;
}
```

Output

The output in the X++ Print window is as follows:

```
1
2
3
4
```

C# Sample of for

```
using System;
public class Pgm_CSharp
{
    static void Main( string[] args )
    {
        new Pgm_CSharp().ForLoops();
    }
    void ForLoops()
    {
        int nLoops = 1, ii;
        for (ii = 1; ii < 5; ii++)
        {
            Console.Out.WriteLine(ii.ToString());
            Console.Out.WriteLine("(Press any key to resume.)");
            Console.In.Read();
            if (ii < 99)
            {
                continue;
            }
            Console.Out.WriteLine("This message never appears.");
        }
        Console.Out.WriteLine("(Press any key to resume.)");
        Console.In.Read();
    }
}
```

Output

The console output from the C# program is as follows:

```
1
(Press any key to resume.)
2
(Press any key to resume.)
3
(Press any key to resume.)
4
(Press any key to resume.)
(Press any key to resume.)
```

X++, C# Comparison: Switch

In both X++ and C#, the **switch** statement involves the keywords **case**, **break**, and **default**. The following table lists the differences in the **switch** statement between X++ and C#.

FEATURE	X++	C#	COMMENTS
`break;` at the end of each case block	In X++, when any **case** block matches the expression value on the **switch** clause, all other **case** and **default** blocks are executed until a `break;` statement is reached. No `break;` statement is ever required in an X++ **switch** statement, but `break;` statements are important in almost all practical situations.	In C#, a `break;` statement is always needed after the statements in a **case** or **default** block. If a **case** clause has no statements between itself and the next **case** clause, a `break;` statement is not required between the two **case** clauses.	We recommend against omitting the `break;` statement after any case **block**, because it can confuse the next programmer who edits the code.
`break;` at the end of the **default** block	In X++ there is no effect of adding a `break;` statement at the end of the **default** block.	In C# the compiler requires a `break;` statement at the end of the **default** block.	For more information, see Switch Statements.
Only constant values on a **case** block	In X++ you can specify either a literal value or a variable on a case block. For example, you can write case myInteger:.	In C# you must specify exactly one literal value on each **case** block, and no variables are allowed.	No comments.
Multiple values on one **case** block	In X++ you can specify multiple values on each case block. The values must be separated by a comma. For example, you can write `case 4,5,myInteger:`.	In C# you must specify exactly one value on each **case** block.	In X++ it is better to write multiple values on one **case** block than to omit the `break;` statement at the end of one or more case blocks.

Code Examples for switch

The following sections show comparable switch statements in X++ and C#.

X++ switch Example

The X++ switch example shows the following:

- `case iTemp:` and `case (93-90):` to show that **case** expressions are not limited to constants, as they are in C#.
- `//break;` to show that `break;` statements are not required in X++, although they are almost always

desirable.

- `case 2, (93-90), 5:` to show that multiple expressions can be listed on one **case** clause in X++.

```
static void GXppSwitchJob21(Args _args)  // X++ job in AOT &gt; Jobs.
{
    int iEnum = 3;
    int iTemp = 6;
    switch (iEnum)
    {
        case 1:
        case iTemp:  // 6
            info(strFmt("iEnum is one of these values: 1,6: %1", iEnum));
            break;
        case 2, (93-90), str2Int("5"):  // Equivalent to three 'case' clauses stacked, valid in X++.
            //case 2:
            //case (93-90):  // Value after each 'case' can be a constant, variable, or expression; in X++.
            //case str2Int("5"):
            info(strFmt("iEnum is one of these values: 2,3,5: %1", iEnum));
            //break;  // Not required in X++, but usually wanted.
        case 4:
            info(strFmt("iEnum is one of these values: 4: %1", iEnum));
            break;
        default:
            info(strFmt("iEnum is an unforeseen value: %1", iEnum));
            break;
            // None of these 'break' occurrences in this example are required for X++ compiler.
    }
    return;
}

/*** Copied from the Infolog:
Message (02:32:08 pm)
iEnum is one of these values: 2,3,5: 3
iEnum is one of these values: 4: 3
***
```

C# switch Example

The C# switch example shows the following:

- case 1: has a comment explaining that only constant expressions can be given on a **case** clause.
- `break;` statements occur after the last statement in each **case** block that has statements, as is required by C#.

```
using System;
namespace CSharpSwitch2
{
    class Program
    {
        static void Main(string[] args)  // C#
        {
            int iEnum = 3;
            switch (iEnum)
            {
                case 1:  // Value after each 'case' must be a constant.
                case 6:
                    Console.WriteLine("iEnum is one of these values: 1,6: " + iEnum.ToString());
                    break;
                //case 2,3,5:  // In C# this syntax is invalid, and multiple 'case' clauses are needed.
                case 2:
                case 3:
                case 5:
                    Console.WriteLine("iEnum is one of these values: 2,3,5: " + iEnum.ToString());
                    break;
                case 4:
                    Console.WriteLine("iEnum is one of these values: 4: " + iEnum.ToString());
                    break;
                default:
                    Console.WriteLine("iEnum is an unforeseen value: " + iEnum.ToString());
                    break;
                // All 'break' occurrences in this example are required for C# compiler.
            }
          return;
        }
    }
}
/*** Output copied from the console:
>> CSharpSwitch2.exe
iEnum is one of these values: 2,3,5: 3
>>
***/
```

X++, C# Comparison: String Case and Delimiters

This section compares the treatment of strings with mixed casing in X++ and C#. It also explains the string delimiters that are available in X++.

Similarities

The following X++ features are the same as in C#:

- The backslash (\) is the escape operator for string delimiters.
- The at sign (@) nullifies the escape effect of the backslash when the at sign is written immediately before the open quotation mark of a string.
- The plus sign (+) is the string concatenation operator.

Differences

X++ features that are different in C# are listed in the following table.

FEATURE	X++	C#	COMMENTS
`==` comparison operator	Insensitive: the `==` operator is insensitive to differences in string casing.	In C#, the `==` operator is sensitive to differences in string casing.	In X++ you can use the strCmp Function for case sensitive comparisons between strings.

FEATURE	X++	C#	COMMENTS
String delimiters	In X++ you can use either the single (') or double (`"`) quotation mark as the string delimiter. **Note:** Usually the best practice is to use double quotation marks for strings that might be displayed to the user. However, it is convenient to delimit a string with single quotation marks when a double quotation mark is one of the characters in the string.	In C# you must use the double quotation mark as the string delimiter. This refers to the type `System.String`.	In X++ and C# you have the option of embedding a delimiter in a literal string and escaping it with . In X++ you also have the alternative of embedding single quotation marks in a string that is delimited by double quotation marks (or the reverse), without having to use the escape.
Character delimiters	X++ has a string data type (`str`), but no character type.	In C# you must use the single quotation mark as the character delimiter. This refers to the type `System.Char`.	In the .NET Framework, a `System.String` of length one is a different data type than a `System.Char` character.

Example 1: Case Sensitivity of the == Operator

The `==` and != operators are case insensitive in X++, but are case sensitive in C#, as is illustrated by the following example.

X++	C#	COMMENTS
`"HELLO" == "hello"` True in X++.	`"HELLO" == "hello"` False in C#.	Different case comparisons between X++ and C#.

Example 2: The + String Concatenation Operator

The + and += operators are used to concatenate strings in both X++ and C#, as is shown by the examples in the following table.

X++	C#	COMMENTS

`myString1 = "Hello" + " world";`
Result is equality:
`myString1 == "Hello world"`
	(Same as for X++.)	In both X++ and C#, the behavior of the + operator depends on the data type of its operands. The operator concatenates strings, or adds numbers.

`mystring2 = "Hello";`
`myString2 += " world";`
Result is equality:
`myString2 == "Hello world"`
	(Same as for X++.)	In both X++ and C#, the following statements are equivalent:
		`a = a + b;`
		`a += b;`

Example 3: Embedding and Escaping String Delimiters

Either single or double quotation marks can be used to delimit strings in X++. The escape character (\) can be used to embed delimiters in a string. These are illustrated in the following table.

X++	C#	COMMENTS

`myString1 = "He said \"yes\".";`
Result:
`He said "yes".`
	(Same as for X++.)	The escape character enables you to embed string delimiters inside strings.

`myString2 = 'He said "yes".';`
Result:
`He said "yes".`
	C# syntax does not allow for single quotation marks to delimit strings.	For strings that may be seen by the user, it is considered a best practice to use the escape character instead of the single quotation marks as shown in the example.

`myString3 = "He said 'yes'.";`
Result:
`He said 'yes'.`
	(Same as for X++.)	In X++, the single quotation marks are not treated as delimiters unless the string starts with a single quotation mark delimiter. In C# the single quotation mark has no special meaning for strings, and it cannot be used to delimit strings. In C# the single quotation mark is the required delimiter for literals of type `System.Char`. X++ has no character data type.

`str myString4 = 'C';` Here the single quotation is a string delimiter.	`char myChar4 = 'C';` Here the single quotation mark is a `System.Char` delimiter, not a `System.String` delimiter.	X++ has no data type that corresponds to `System.Char` in the .NET Framework. An X++ string that is limited to a length of one is still a string, not a character data type.

Example 4: Single Escape Character

Examples that illustrate the single escape character in either the input or the output are shown in the following table.

X++	C#	COMMENTS

`myString1 = "Red\ shoe";`
Result:
`Red shoe`
	A literal string in C# cannot contain the two character sequence of escape followed by a space, such as "\ ". A compiler error occurs.	When the X++ compiler encounters the two character sequence of "\ ", it discards the single escape character.

`myString2 = "Red\\ shoe";`
Result:
`Red\ shoe`
	(Same as for X++.)	In a pair of escape characters, the first negates the special meaning of the second.

Comparison: Array Syntax

There are similarities and differences in the features and syntax for arrays in X++ versus C#.

Similarities

Overall there is much similarity in the syntax and treatment of arrays in X++ and C#. However there are many differences.

Differences

The following table lists areas in the [] syntax for arrays that are different for X++ and C#.

CATEGORY	X++	C#	COMMENTS

Declaration	An array is declared with square brackets appended to the variable name.	An array is declared with square brackets appended to the data type.	`int myInts[]; // X++`
			Note: An X++ array cannot be a parameter in a method.
			`int[] myInts; // C#`

Declaration	The array syntax supports only primitive data types, such as `int` and `str`. The syntax does not support classes or tables.	The array syntax supports primitive data types and classes.	In X++ you can use the `Array` Array for an array of objects.
Declaration	X++ is limited to single dimension arrays (myStrings[8]).	C# adds support for multi-dimensional arrays (myStrings[8,3]) and for jagged arrays (myStrings[8][3]).	In X++ you cannot have an array of arrays. However, there is advanced syntax for limiting the amount of active memory that a large array can consume, which looks like the multi-dimensional syntax in C#: int intArray[1024,16];. For more information, see Best Practice Performance Optimizations: Swapping Arrays to Disk.
Declaration	In X++ an array is a special construct but it is not an object.	In C# all arrays are objects regardless of syntax variations.	X++ does have an Array class, but its underlying mechanism differs from arrays created by using the [] syntax. In C# all arrays use the same underlying mechanism, regardless of whether [] syntax of the `System.Array` class is used in your code.
Length	In X++ the length of a static sized array is determined in the declaration syntax.	In C# the size of an array is determined when the array object is constructed.	When you use the [] declaration syntax in X++, no more preparation is needed before you assign values to the array. In C# you must declare and then construct the array before assigning to it.

Length	An X++ array can have a dynamic length that can be increased even after population has begun. This applies only when the array is declared without a number inside the []. Performance might be slowed if the length of the dynamic array is increased many times.	In C# the length of an array cannot be changed after the length is set.	In the following fragment of X++ code, only the `myInts` array is dynamic and can increase in size.
			`int myInts[];`
			`int myBools[5];`
			`myInts[2] = 12;`
			`myInts[3] = 13;`
			`myBools[6] = 26; //Error`

CATEGORY	X++	C#	COMMENTS
Length	You can get the length of some arrays by using the `dimOf` function.	C# arrays are objects that have a `Length` property.	No comments.
Indexing	Array indexing is 1 based.	Array indexing is 0 based.	mtIntArray[0] would cause an error in X++.
Constant	In X++ a constant value is best achieved by using the **#define** precompiler directive.	In C# you can decorate your variable declaration with the keyword **const**, to achieve a constant value.	X++ has no **const** keyword. C# cannot assign values to variables that are created by its #define precompiler directive.

X++ and C# Samples

The following code samples show how arrays of primitive data types are handled. The first sample is in X++, and the second sample is in C#. Both samples achieve the same results.

X++ Sample

```
static void JobRs005a_ArraySimple(Args _args)
{
    #define.macroArrayLength(3)
    // Static length.
    str sSports[#macroArrayLength];
    // Dynamic length, changeable during run time.
    int years[];
    int xx;
    Global::warning("-------- SPORTS --------");
    sSports[#macroArrayLength] = "Baseball";
    for (xx=1; xx <= #macroArrayLength; xx++)
    {
        info(int2str(xx) + " , [" + sSports[xx] + "]");
    }
    warning("-------- YEARS --------");
    years[ 4] = 2008;
    years[10] = 1930;
    for (xx=1; xx <= 10; xx++)
    {
        info(int2str(xx) + " , " + int2str(years[xx]));
    }
}
```

Output

The output to the Infolog is as follows:

```
Message (14:16:08)
-------- SPORTS --------
1 , []
2 , []
3 , [Baseball]
-------- YEARS --------
1 , 0
2 , 0
3 , 0
4 , 2008
5 , 0
6 , 0
7 , 0
8 , 0
9 , 0
10 , 1930
```

C# Sample

```
using System;
public class Pgm_CSharp
{
    static public void Main( string[] args )
    {
        new Pgm_CSharp().ArraySimple();
    }
    private void ArraySimple()
    {
        const int const_iMacroArrayLength = 3;
        // In C# the length is set at construction during run.
        string[] sSports;
        int[] years;
        int xx;
        Console.WriteLine("-------- SPORTS --------");
        sSports = new string[const_iMacroArrayLength];
        sSports[const_iMacroArrayLength - 1] = "Baseball";
        for (xx=0; xx < const_iMacroArrayLength; xx++)
        {
            Console.WriteLine(xx.ToString() + " , [" + sSports[xx] + "]");
        }
        Console.WriteLine("-------- YEARS --------");
        // In C# you must construct the array before assigning to it.
        years = new int[10];
        years[ 4] = 2008;
        years[10 - 1] = 1930;
        for (xx=0; xx < 10; xx++)
        {
            Console.WriteLine(xx.ToString() + " , [" + years[xx].ToString() + "]");
        }
    }
} // EOClass
```

Output

The output from the C# program to the command line console is as follows:

```
-------- SPORTS --------
0 , []
1 , []
2 , [Baseball]
-------- YEARS --------
0 , [0]
1 , [0]
2 , [0]
3 , [0]
4 , [2008]
5 , [0]
6 , [0]
7 , [0]
8 , [0]
9 , [1930]
```

Additional array-like X++ features

The **container** is a special data type that is available in X++. It can be considered as similar to an array, or similar to a `List` collection.

Comparison: Collections

In a Finance and Operations application, you can use the X++ `List` collection class. The .NET Framework that is used in C# has a similar class named `System.Collections.Generic.List`.

Comparing the Use of the List Classes

The following table compares methods on the X++ `List` class to the methods on `System.Collections.Generic.List` from the .NET Framework and C#.

FEATURE	X++	C#	COMMENTS
Declaration of collection	`List myList;`	`List<string> myList;`	The X++ declaration does not include the type of elements to be stored.
Declaration of iterator	`ListIterator iter` `ListEnumerator enumer;`	IEnumerator<string> iter;	In X++ the `ListIterator` object has methods that can `insert` and `delete` items from the `List`. The X++ `ListEnumerator` cannot modify the contents of the `List`. In X++ the `ListEnumerator` object is always created on the same tier as the `List`. This is not always true for `ListIterator`.
Obtaining an iterator	`new ListIterator (myList)` `myList.getEnumerator()`	`myList.GetEnumerator()`	In both X++ and C#, the List object has a getter method for an associated enumerator.

FEATURE	X++	C#	COMMENTS
Constructor	`new List(Types::String)`	`new List<string>()`	Information about the type of objects to be stored inside the `List` classes is given to the constructor in both X++ and C#.
Updating data	Enumerator – the enumerator becomes invalid if any items in the `List` are added or removed. Iterator – the iterator has methods that insert and delete items from the `List`. The iterator remains valid.	Enumerator – the enumerator becomes invalid if any items in the `List` are added or removed.	Enumerators become invalid after items are added or deleted from the `List`, in both X++ and C#.
Updating data	In X++ the `List` class has methods for adding items at the start or end of the list.	In C# the `List` class has methods for adding members at any position in the list. It also has methods for removing items from any position.	In X++ items can be removed from the `List` only by an iterator.

Example 1: Declaration of a List

Following are code examples in X++ and C# that declare `List` collections.

```
// X++
List listStrings ,list2 ,listMerged;
ListIterator literator;
```

```
// C#
using System;
using System.Collections.Generic;
List<string> listStrings ,list2 ,listMerged; IEnumerator<string> literator;
```

Example 2: Construction of a List

In both languages, the type of items that the collection stores must be specified at the time of construction. For class types, X++ can get no more specific than whether the type is a class (Types::Class). Following are code examples in X++ and C#.

```
// X++
listStrings = new List( Types::String );
```

```
// C#
listStrings = new List<string>;
```

Example 3: Add Items to a List

In both X++ and C#, the collection provides a method for appending an item to the end of the collection, and for inserting an item the start. In C# the collection provides a method for inserting at any point in the collection based on an index value. In X++ a collection iterator can insert an item at its current position. Following are code

examples in X++ and C#.

```
// X++
listStrings.addEnd ("StringBB.");
listStrings.addStart ("StringAA.");
// Iterator performs a midpoint insert at current position.
listIterator.insert ("dog");
```

```
// C#
listStrings.Add ("StringBB.");
listStrings.Insert (0 ,"StringAA.");
// Index 7 determines the insertion point.
listStrings.Insert (7 ,"dog");
```

Example 4: Iterate Through a List

Both X++ and C# have iterator classes that you can use to step through the items in a collection as shown in the following examples.

```
// X++
literator = new ListIterator (listStrings);
// Now the iterator points at the first item.

// The more method answers whether
// the iterator currently points
// at an item.
while (literator.more())
{
    info(any2str (literator.value()));
    literator.next();
}
```

```
// C#
literator = listStrings .GetEnumerator();
// Now enumerator points before the first item, not at the first item.

// The MoveNext method both advances the item pointer, and
// answers whether the pointer is pointing at an item.
while (literator.MoveNext())
{
    Console.WriteLine (literator.Current);
}
```

Example 4b: foreach in C#

In C# the `foreach` keyword is often used to simplify the task of iterating through a list. The following code example behaves the same as the previous C# example.

```
foreach (string currentString in listStrings)
{
    Console.WriteLine(currentString);
}
```

Example 5: Delete the Second Item

The following code examples delete the second item from the collection. In X++ this requires an iterator. In C# the collection itself provides the method for removing an item.

```
// X++
literator.begin();
literator.next();
literator.delete();
```

```
// C#
listStrings.RemoveAt(1);
```

Example 6: Combine Two Collections

The following code examples combine the contents of two collections into one.

```
// X++
listStrings = List::merge(listStrings ,listStr3);
// Or use the .appendList method:
listStrings.appendList (listStr3);
```

```
// C#
listStrings.InsertRange(listStrings.Count ,listStr3);
```

Comparison: Collections of keys with values

In a Finance and Operations application, you can use the `Map` collection class. The `Map` collection holds pairs of values, the key value plus a data value. This resembles the .NET Framework class named `System.Collections.Generic.Dictionary`.

Similarities

The following list describes similarities between X++ and C# regarding their collections that store key-value pairs:

- Both prevent duplicate keys.
- Both use an enumerator (or iterator) to loop through the items.
- Both key-value collection objects are constructed with designations of the types that are stored as key and value.
- Both can store class objects, and are not limited to storing primitives like **int**.

Differences

The following table describes differences between X++ and C# regarding their collections classes that store key-value pairs:

FEATURE	X++	C#	COMMENTS
Duplicate keys	In X++ the `Map` class prevents duplicate keys by implicitly treating your call to its `insert` method as an operation to update only the value associated with the key.	In C# the `Dictionary` class throws an exception when you try to add a duplicate key.	Duplicate keys are prevented in both languages, although by different techniques.

FEATURE	X++	C#	COMMENTS
Delete items	In X++ the `delete` method on an iterator object is used to remove an unwanted key-value pair from a `Map`.	In C# the `Dictionary` class has a `remove` method.	In both languages, an enumerator is made invalid if the collection item count is modified during the life of the enumerator.

Example 1: Declaration of a Key-Value Collection

In both languages, the type of items that the key-value collection stores must be specified. In X++ the type is specified at time of construction. In C# the type is specified at both the time of declaration and the time of construction. Following are code examples in X++ and C#.

```
// X++
Map mapKeyValue;
MapEnumerator enumer;
MapIterator mapIter;
```

```
// C#
Dictionary<int,string> dictKeyValue;
IEnumerator<SysCollGen.KeyValuePair<int,string>> enumer;
KeyValuePair<int,string> kvpCurrentKeyValuePair;
```

Example 2: Construction of the Collection

In both languages, the type of items that the key-value collection stores specified during construction. For class types, X++ can get no more specific than whether the type is a class (Types::Class). Following are code examples in X++ and C#.

```
// X++
mapKeyValue = new Map(Types::Integer, Types::String);
```

```
// C#
dictKeyValue = new Dictionary<int,string>();
```

Example 3: Add an Item to the Collection

There is almost no difference in how an item is added to a key-value collection in X++ and C# as shown in the following code examples.

```
// X++
mapKeyValue.insert(xx ,int2str(xx) + “_Value”);
```

```
// C#
dictKeyValue.Add(xx ,xx.ToString() + “_Value”);
```

Example 4: Iterate Through a Key-Value Collection

Enumerators are used to loop through the key-value collections in both X++ and C# as shown in the following code examples.

```
// X++
enumer = mapKeyValue.getEnumerator();
while (enumer.moveNext())
{
    iCurrentKey = enumer.currentKey();
    sCurrentValue = enumer.currentValue();
    // Display key and value here.
}
```

```
// C#
enumer = dictKeyValue.GetEnumerator();
while (enumer.MoveNext())
{
    kvpCurrentKeyValuePair = enumer.Current;
    // Display .Key and .Value properties=
    // of kvpCurrentKeyValuePair here.
}
```

Example 5: Update the Value Associated with a Key

The syntax is very different between the two languages for an update of the value associated to a given key. Following are code examples for the key 102.

```
// X++
mapKeyValue.insert(
    102 ,
    ".insert(), Re-inserted" + " key 102 with a different value.");
```

```
// C#
dictKeyValue[102] =
    "The semi-hidden .item property in C#, Updated the value for key 102.";
```

Example 6: Delete One Item

The syntax is very different between the two languages to delete one key-value pair from a collection, while iterating through the collection members. Code examples for the key 102 are shown below.

```
// X++
mapIter = new MapIterator(mapKeyValue);
//mapIter.begin();
while (mapIter.more())
{
    iCurrentKey = mapIter.key();
    if (104 == iCurrentKey)
    {
        // mapKeyValue.remove would invalidate the iterator.
        mapIter.delete();
        break;
    }
    mapIter.next();
}
```

```
// C#
dictKeyValue.Remove(104);
```

Comparison: Exceptions

There are some similarities but many differences when we compare exception related behavior between X++ and C#. The **try**, **catch**, and **throw** keywords behave the same in X++ and C#. But the types of exceptions thrown and caught are different for the two languages.

Similarities

Similarities between X++ and C# regarding their exception features include the following:

- Both languages have the same **try** keyword.
- Both have the same **catch** keyword.
- Both enable for a **catch** statement that does not specify any particular exception. Such a **catch** statement catches all exceptions that reach it.
- Both have the same **throw** keyword.

Differences

Exception-related differences between X++ and C# are described in the following table.

FEATURE	X++	C#	COMMENTS
retry	Jumps to the first instruction in the associated **try** block. For more information, see Exception Handling with try and catch Keywords.	The functionality of the **retry** keyword can be mimicked in C# code, but there is no corresponding keyword.	Only X++ has a **retry** keyword. C# has no counterpart. For more information, see X++, C# Comparison: Automated Retry After an Exception.
finally	The `finally` keyword is supported to follow the `try` and `catch` keywords.	The **finally** keyword marks a block of code that follows the **try** and **catch** blocks. The finally will be executed regardless of whether any exception is thrown or caught.	The semantics are identical to the semantics in C#.
Specific exceptions	In X++ an exception is an element of the `Exception` enum, such as **Error**, **Deadlock**, or **CodeAccessSecurity**. No exception can contain another.	In C# an exception is an instance of the `System.Exception` base class, or any class that inherits from it. An exception can be contained in the `InnerException` property of the thrown exception.	In X++ each thrown exception is a value of the Exception enum. For more information, see Exception Enumeration.
Exception message	In X++ the message that is created when an exception is raised is available only in the Infolog, and the message is not directly tied to the exception.	In C# the message is the `Message` member of the `System.Exception` object.	In X++ the Global::error method is the mechanism that display exception messages in the Infolog. For more information, see Exception Handling with try and catch Keywords.

FEATURE	X++	C#	COMMENTS
Exception conditions	In X++ an error occurs when you call an instance method on an object variable that has not yet had anything assigned to it. However, no exception is raised along with this error. Therefore no `catch` block can gain control even if the unassigned variable is misused in a `try` block. In the following code example, the error caused by the code `box4.toString();` does not cause control to transfer to any `catch` block: `DialogBox box4;` `try` { `box4.toString();` `info("toString did not error, but expected an error.");` } catch (Exception::Error) // No Exception value catches this. { `info("Invalid use of box4 gave control to catch, unexpected.");` }	In C# a `System.NullReferenceException` is raised when an uninitialized variable is treated as an object reference.	There might be several other differences in the conditions that raise exceptions.
SQL transactions	In X++ when an SQL exception occurs in a **ttsBegin** - **ttsCommit** transaction, no **catch** statement inside the transaction block can process the exception.	In C# a catch block inside an SQL transaction can catch the exception.	

Examples

The following X++ features are demonstrated:

- **try** keyword.
- **catch** keyword.
- The behavior after an Exception::Error exception occurs.

X++ Example

```
// X++
static void JobRs008a_Exceptions(Args _args)
{
    str sStrings[4];
    int iIndex = 77;
    try
    {
        info("On purpose, this uses an invalid index for this array: " + sStrings[iIndex]);
        warning("This message does not appear in the Infolog," + " it is unreached code.");
    }
    // Next is a catch for some of the values of
    // the X++ Exception enumeration.
    catch (Exception::CodeAccessSecurity)
    {
        info("In catch block for -- Exception::CodeAccessSecurity");
    }
    catch (Exception::Error)
    {
        info("In catch block for -- Exception::Error");
    }
    catch (Exception::Warning)
    {
        info("In catch block for -- Exception::Warning");
    }
    catch
    {
        info("This last 'catch' is of an unspecified exception.");
    }
    //finally
    //{
    //    //Global::Warning("'finally' is not an X++ keyword, although it is in C#.");
    //}
    info("End of program.");
}
```

Output

Here is the output from the Infolog window:

```
Message (18:07:24)
Error executing code: Array index 77 is out of bounds.
Stack trace
(C)\Jobs\JobRs008a_Exceptions - line 8
In catch block for -- Exception::Error
End of program.
```

C# Sample

The following C# program is a rewrite of the previous X++ program.

```
// C#
using System;
public class Pgm_CSharp
{
    static void Main( string[] args )
    {
        new Pgm_CSharp().Rs008a_CSharp_Exceptions();
    }
    void Rs008a_CSharp_Exceptions()
    {
        //str sStrings[4];
        string[] sStrings = new string[4];
        try
        {
            Console.WriteLine("On purpose, this uses an invalid index for this array: " + sStrings[77]);
            Console.Error.WriteLine("This message does not appear in the Infolog, it is unreached code.");
        }
        catch (NullReferenceException exc)
        {
            Console.WriteLine("(e1) In catch block for -- " + exc.GetType().ToString() );
        }
        catch (IndexOutOfRangeException exc)
        {
            Console.WriteLine("(e2) In catch block for -- " + exc.GetType().ToString() );
        }
        // In C#, System.Exception is the base of all
        // .NET Framework exception classes.
        // No as yet uncaught exception can get beyond
        // this next catch.
        catch (Exception exc)
        {
            Console.WriteLine("This last 'catch' is of the abstract base type Exception: "
                + exc.GetType().ToString());
        }
        // The preceding catch of System.Exception makes this catch of
        // an unspecified exception redundant and unnecessary.
        //catch
        //{
        //    Console.WriteLine("This last 'catch' is"
        //        + " of an unspecified exception.");
        //}
        finally
        {
            Console.WriteLine("'finally' is not an X++ keyword, although it is in C#.");
        }
        Console.WriteLine("End of program.");
    }
} // EOClass
```

Output

Here is the output to the C# console:

```
(e2) In catch block for -- System.IndexOutOfRangeException
'finally' is not an X++ keyword, although it is in C#.
End of program.
```

Comparison: Automated Retry After an Exception

Sometimes you can write code in a catch block that fixes the cause of an exception that occurs during run time. X++ provides a **retry** keyword that can be used only inside a **catch** block. The **retry** keyword enables a program to jump back to the start of the **try** block after the problem has been corrected by code in the **catch** block. C# does not have a **retry** keyword. However, C# code can be written to provide equivalent behavior.

Code Samples for Retry

The following X++ sample program causes an Exception::Error to be raised. This occurs when it first tries to read an element from the `sStrings` array by using an invalid index value. When the exception is caught, corrective action is taken during run time inside the **catch** block. The retry statement then jumps back to the first statement in the **try** block. This second iteration works without encountering any exception.

```
static void JobRs008b_ExceptionsAndRetry(Args _args)
{
    str sStrings[4];
    str sTemp;
    int iIndex = 0;

    sStrings[1] = "First array element.";
    try
    {
        print("At top of try block: " + int2str(iIndex));
        sTemp = sStrings[iIndex];
        print( "The array element is: " + sTemp );
    }
    catch (Exception::Error)
    {
        print("In catch of -- Exception::Error (will retry)." + " Entering catch.");
        ++iIndex;
        print("In catch of -- Exception::Error (will retry)." + " Leaving catch.");
        // Here is the retry statement.
        retry;
    }
    print("End of X++ retry program.");
    pause;
}
```

Output

Here is the output to the Print window:

```
At top of try block: 0
In catch of -- Exception::Error (will retry). Entering catch.
In catch of -- Exception::Error (will retry). Leaving catch.
At top of try block: 1
The array element is: First array element.
End of X++ retry program.
```

C# Sample

The following C# sample is not a line-by-line translation from the previous X++ sample. Instead the C# program has a different structure so that it mimics the behavior of the **retry** keyword that the X++ program relies on. The **try** and **catch** blocks are in a called method. The variables that are used in the **try** block are stored in the caller method. The caller method passes the variables as parameters that are decorated with the **ref** keyword, so that their values can be corrected inside the **catch** block of the called method. The called method captures all exceptions, and returns a **boolean** to communicate back to the caller whether a second call is required.

```
// C#
using System;
public class Pgm_CSharp
{
    static void Main(string[] args)
    {
        new Pgm_CSharp() .Rs008b_CSharp_ExceptionsAndRetry();
    }
    void Rs008b_CSharp_ExceptionsAndRetry() // Caller
    {
        int iIndex = -1
            , iNumRetriesAllowed = 3;
        bool bReturnCode = true; // Means call the callee method.
        for (int xx=0; xx <= iNumRetriesAllowed; xx++)
        {
            if (bReturnCode)
            {
                bReturnCode = this.Rs008b_CSharp_ExceptionsAndRetry_Callee(ref iIndex);
            }
            else
            {
                break;
            }
        }
        Console.WriteLine("End of C# caller method.");
    }

    private bool Rs008b_CSharp_ExceptionsAndRetry_Callee(ref int iIndex)
    {
        bool bReturnCode = true; // Means call this method again.
        string[] sStrings = new string[4];
        string sTemp;
        sStrings[0] = "First array element.";
        try
        {
            Console.WriteLine("At top of try block: " + iIndex.ToString());
            sTemp = sStrings[iIndex];
            Console.WriteLine( "The array element is: " + sTemp );
            bReturnCode = false; // Means do not call this method again.
        }
        catch (Exception)
        {
            Console.WriteLine("In catch of -- Exception. Entering catch.");
            ++iIndex; // The 'ref' parameter in C#.
            Console.WriteLine("In catch of -- Exception. Leaving catch.");
            //retry;
            // In C# we let the caller method do the work
            // that the retry keyword does in X++.
        }
        Console.WriteLine("End of C# callee method.");
        return bReturnCode;
    }
}
```

Output

Here is the output to the console:

```
At top of try block: -1
In catch of -- Exception. Entering catch.
In catch of -- Exception. Leaving catch.
End of C# callee method.
At top of try block: 0
The array element is: First array element.
End of C# callee method.
End of C# caller method.
```

Comparison: Operators

This section compares the operators between X++ and C#.

Assignment Operators

The following table displays the differences between the assignment operators in X++ and C#.

X++ AND C#	DIFFERENCES
`=`	In X++ this operator causes an implicit conversion whenever a loss of precision might occur, such for an assignment from an **int64** to an **int**. But in C# the assignment causes a compile error.
`+=` and `-=`	The only difference is that in C# these operators are also used in delegate manipulation.

++ and --	These are the increment and decrement operators in both languages. The following line is identical in both languages:
	`++myInteger;`
	But in X++ these two operators are for statements, not for expressions. Therefore the following lines generate compile errors in X++:
	`myStr = int2str(++myInteger);`
	`myIntA = myIntBB++;`

Arithmetic Operators

The following table lists the arithmetic operators.

X++ AND C#	DIFFERENCES
*	As the multiplication operator, there are no differences. **Note:** The asterisk is also used in the SQL statements that are part of the X++ language. In these SQL statements the asterisk can also be one of the following: • A wildcard indicating that all the columns should be returned. • A wildcard for characters in a string that is used on a **like** clause.
`/`	The division operator is the same in X++ and C#.
`MOD`	For modulo operations, the only difference is that the % symbol is used in C#.

X++ AND C#	DIFFERENCES
+	The addition operator is the same in X++ and C#. The plus sign is also used for string concatenation. This operator adds numbers and concatenates strings in both languages.
-	The subtraction operator is the same in X++ and C#.

Bitwise Operators

The following table compares the bitwise operators between X++ and C#.

X++ AND C#	DIFFERENCES
<<	The left shift operator is the same in X++ and C#.
>>	The right shift operator is the same in X++ and C#.
~	The bitwise NOT operator is the same in X++ and C#.
&	The binary AND operator is the same in X++ and C#.
^	The binary XOR operator is the same in X++ and C#.

Relational Operators

The following relational operators are the same in X++ and C#:

- `==`
- `<=`
- `<=`
- `>`
- `<`
- `!=`
- `&&`
- `||`
- `!`
- `? :`

Comparison: Events

There are some differences in how X++ and C# implement the event design pattern. For more information, see Event Terminology and Keywords.

Comparison of Events between X++ and C#

There are differences in the way delegates are used for events in X++ versus C#.

CONCEPT	X++	C#	COMMENTS

CONCEPT	X++	C#	COMMENTS
delegate	In X++, a delegate can be declared only as a member on a class. A delegate cannot be a member on a table. All delegates are instance members of their class, not **static** members. No access modifier can be used on a delegate declaration, because all delegates are **protected** members. Therefore, the event can be raised only by code within the same class where the delegate is a member. However, the one exception to the private nature of a delegate is that code outside their class can operate on the delegates by using the += and -= operators.	In C#, each **delegate** is a type, just as every **class** is a type. A delegate is declared independently of any class. Without the **event** keyword, you can have a delegate as a parameter type on a method, just as you can have a class as a parameter type. You can construct an instance of a delegate to pass in for the parameter value.	In X++, each class is a type, but no delegate is a type. You cannot construct an instance of a delegate. No delegate can be a parameter for a method. But you can create a class that has a delegate member, and you can pass instances of the class as parameter values. For more information, see X++ Keywords.
event	In X++ code, an event is one of the following: • An explicit call to a delegate. • The start or end of a method. There is no **event** keyword in X++.	In C#, the **event** keyword is used to declare a **delegate** type as a member of a class. The effect of the **event** keyword is to make the delegate **protected**, yet still accessible for the += and -= operators. You can subscribe event handler methods to an **event** by using the += operator. A **delegate** can be useful without the **event** keyword, as a technique for passing a function pointer as a parameter into a method.	The automatic events that occur before the start of a method, and after the end of a method, can be subscribed to only by using the AOT.
+= and -= operators	In X++, you use the += operator to subscribe methods to a **delegate**. The -= operator unsubscribes a method from a delegate.	In C#, you use the += operator to subscribe methods to an **event**, or to a **delegate** that is not used with the **event** keyword.	The delegate contains a reference to all the objects that have methods subscribed to the delegate. Those objects are not eligible for garbage collection while delegate holds those references.
`eventHandler`	In X++, the **eventHandler** keyword is required when you use either the += or -= operator to subscribe or unsubscribe a method from a delegate.	`System.EventHandler` is a delegate type in the .NET Framework.	This term is used differently in X++ than it is in C# or the .NET Framework. For more information, see X++ Keywords.

X++ Example

The important things to notice in the X++ example are the following:

- The `XppClass` has a delegate member that is named `myDelegate`.

  > **NOTE**
  > The AOT contains a node for the delegate. The node is located at AOT > Classes > XppClass > myDelegate. Several event handler nodes can be located under the myDelegate node. Event handlers that are represented by nodes in the AOT cannot be removed by the -= operator during run time.

- The {} braces at the end of the delegate declaration are required, but they cannot have any code in them.
- The `XppClass` has two methods whose parameter signatures are compatible with the delegate. One method is static.
- The two compatible methods are added to the delegate with the += operator and the **eventHandler** keyword. These statements do not call the event handler methods, the statements only add the methods to the delegate.
- The event is raised by one call to the delegate.
- The parameter value that passed in to the delegate is received by each event handler method.
- The short X++ job at the top of the example starts the test.

```
// X++
// Simple job to start the delegate event test.
static void DelegateEventTestJob()
{
    XppClass::runTheTest("The information from the X++ job.");
}
// The X++ class that contains the delegate and the event handlers.
class XppClass
{
    delegate void myDelegate(str _information)
    {
    }
    public void myEventSubscriberMethod2(str _information)
    {
        info("X++, hello from instance event handler 2: " + _information);
    }
    static public void myEventSubscriberMethod3(str _information)
    {
        info("X++, hello from static event handler 3: " + _information);
    }
    static public void runTheTest(str _stringFromJob)
    {
        XppClass myXppClass = new XppClass();
        // Subscribe two event handler methods to the delegate.
        myXppClass.myDelegate += eventHandler(myXppClass.myEventSubscriberMethod2);
        myXppClass.myDelegate += eventHandler(XppClass::myEventSubscriberMethod3);
        // Raise the event by calling the delegate one time,
        // which calls all the subscribed event handler methods.
        myXppClass.myDelegate(_stringFromJob);
    }
}
```

The output from the previous X++ job is as follows:

```
X++, hello from static event handler
3: The information from the X++ job. X++, hello from instance event handler
2: The information from the X++ job.
```

C# Sample

This section contains a C# code sample for the event design pattern of the previous X++ sample.

```
// C#
using System;
// Define the delegate type named MyDelegate.
public delegate void MyDelegate(string _information);
public class CsClass
{
    protected event MyDelegate MyEvent;
    static public void Main()
    {
        CsClass myCsClass = new CsClass();
        // Subscribe two event handler methods to the delegate.
        myCsClass.MyEvent += new MyDelegate(myCsClass.MyEventSubscriberMethod2);
        myCsClass.MyEvent += new MyDelegate(CsClass.MyEventSubscriberMethod3);
        // Raise the event by calling the event one time, which
        // then calls all the subscribed event handler methods.
        myCsClass.MyEvent("The information from the C# Main.");
    }
    public void MyEventSubscriberMethod2(string _information)
    {
        Console.WriteLine("C#, hello from instance event handler 2: " + _information);
    }
    static public void MyEventSubscriberMethod3(string _information)
    {
        Console.WriteLine("C#, hello from static event handler 3: " + _information);
    }
}
```

The output from the previous C# sample is as follows:

```
CsClass.exe C#, hello from instance event handler
2: The information from the C\# Main. C\#, hello from static event handler
3: The information from the C\# Main.
```

Events and the AOT

There are other event systems that apply only to items in the AOT. For more information, see Event Handler Nodes in the AOT.

Comparison: Precompiler Directives

X++ and C# share some keywords for their precompiler directive syntax, but the meanings are not always the same.

Similarities

The X++ and C# compilers recognize many of the same keywords. In most cases, the keywords mean the same for both language compilers.

Differences

A fundamental difference between the precompiler directives in X++ versus C# is the #define keyword that both language precompilers recognize. Unlike C#, in X++ the #define directive requires a dot in its syntax. In X++, parentheses can be used to give the defined symbol a value. These differences are shown in the following examples:

- In X++: #define.InitialYear(2003)
- In C#: #define InitialYear

A minor difference is that in C# there can be spaces and tab characters between the # character and the directive

keyword, such as # define Testing.

Identical Keywords

The following table lists precompiler directives that are similar in X++ and C#.

KEYWORD	X++	C#	COMMENTS
`#define`	In X++, a precompiler variable name can be defined, and a value can be given to that variable.	In C#, a precompiler variable name can be defined, but no value can be given to that variable. Also, any #define in C# must occur at the top of the file, and cannot occur after any code such as a using statement or a class declaration.	The C# compiler can input a command line parameter of `/define` to define a precompiler variable name without defining the variable in any C# code file. The X++ compiler has no counterpart to `/define`.
`#if`	In X++, #if can determine whether a precompiler variable exists, and whether the variable has a given value.	In C#, #if can only determine whether a precompiler variable exists. It cannot test for any value because no value can be assigned.	
`#endif`	In X++, #endif marks the end of an #if block. It also ends an #ifnot block.	In C#, #endif marks the end of an #if block, regardless of whether the block includes a #else.	

Different Keywords with the Same Processing Result

The following table lists precompiler directives that are named differently in X++ and C#, but that give the same results when processed.

X++	C#	COMMENTS
#ifnot	#if #else	There is no #else directive in X++, but the #ifnot provides similar functionality. In X++, #ifnot can determine whether a precompiler variable exists, and whether the variable does not have a specific given value. In C#, #if can determine whether a precompiler variable exists when the '!' symbol is prefixed to the variable name.
`//BP Deviation documented`	#pragma warning	These X++ and C# entries are not equivalent, but there is a partial similarity. Both suppress compiler warning messages.
#macrolib	.HPP file in C++	There is a partial similarity between the X++ directive #macrolib versus an .HPP file in C++. Both can contain several #define statements.

Precompiler Directives Exclusive to X++

The following table lists X++ precompiler directives that have no direct counterpart in C#.

X++	COMMENTS
#linenumber	The #linenumber directive is for obtaining the line number, so that it can be output to the Infolog. The C# directive #line is different because its purpose is for setting the line number.
#defdec #definc	
#globaldefine	In X++, there is a small difference between #globaldefine versus #define. The difference is that #globaldefine never overwrites a current nonnull value that was assigned to a precompiler variable by #define. C# has nothing similar to this difference, because in C#, a precompiler variable name cannot be given a value.
#localmacro #macro	In X++, #localmacro enables you to assign a multiline value to a precompiler variable. #macro is a synonym, but #localmacro is recommended. In C#, the #define directive has part of this functionality, but it cannot assign a value to a precompiler variable.
#globalmacro	In X++, #globalmacro is almost the same as the preferred #localmacro.

Comparison: Object Oriented Programming

The object oriented programming (OOP) principles of X++ differ from C#.

Conceptual Comparisons

The following table compares the implementation of OOP principles between X++ and C#.

FEATURE	X++	C#	COMMENTS
Casting	The X++ language has the keywords **is** and **as**, which are used to make downcasts safe and explicit. **Tip**: X++ does not require the use of the **as** keyword when you downcast a base class variable to a derived class variable. However, we recommend that all downcast statements use the **as** keyword.	An object can be cast either up or down the inheritance path. Downcasts require the **as** keyword.	For more information about the X++ keywords **is** and **as**, see Expression Operators: Is and As for Inheritance.
Local functions	A method can contain a declaration and code body for zero or more local functions. Only that method can have calls to the local function.	C# 3.0 supports lambda expressions, which have some similarity to anonymous functions and local functions. Lambda expressions are often used with delegates.	

FEATURE	X++	C#	COMMENTS
Method overloading	Method overloading is not supported. A method name can occur only one time per class.	Method overloading is supported. A method name can occur multiple times in one class, with different parameter signatures in each case.	X++ does support optional parameters on methods. Optional parameters can partially mimic method overloading. For more information, see the row for optional parameters in this table.
Method overriding	Method overriding is supported. A derived class can have a method by the same name as in the base class, as long as the parameter signature is the same in both cases. The only exception is that the overriding method can add a default value to a parameter.	Method overriding is supported. The **virtual** keyword must be applied to a method before the method can be overridden in a derived class.	The concept of overriding a method includes the method name, its parameter signature, and its return type. The concept of method overriding does not apply if the base method and the overriding method differ in any of these aspects.
Optional parameters	A parameter declaration can be followed by a default value assignment. The method caller has the option of passing a value for that parameter, or ignoring the parameter to accept the default value. This feature mimics method overloading because two calls to the same method name can pass different numbers of parameters. Each parameter that has a default value must follow the last parameter that does not have a default value.	Optional parameters are supported by the **params** keyword. Even without the **params** keyword, from the point of view of the caller, method overloading can provide partially similar functionality.	For more information, see Parameters and Scoping and Using Optional Parameters.

FEATURE	X++	C#	COMMENTS
Single inheritance	You can derive your X++ class from another X++ class by using the **extends** keyword in the classDeclaration node of your class, in the AOT. No class implicitly derives directly from another class. If you want your class to directly derive from the `Object` class, you must use the **extends** keyword. You can specify only one class on the **extends** keyword. **Caution**: When you modify an X++ base class that other classes derive from, you must recompile that base class using the Compile forward. This option ensures that the derived classes are also recompiled. To ensure the derived classes are also recompiled, right-click the base class node, and then click Add-Ins > Compile forward. The alternative of clicking Build > Compile (or pressing the F7 key) is sometimes insufficient for a base class change. A class can implement zero to many interfaces. An X++ table implicitly inherits from the `Common` table, and from the `xRecord` class.	C# uses the **extends** keyword to derive from another class. All .NET Framework classes implicitly derive from the `System.Object` class, unless they explicitly derive from another class.	

Keyword Comparisons

The following table lists the OOP-related keywords in X++ and C#.

KEYWORD	X++	C#	COMMENTS
abstract			No difference.
class	The modifiers **public** and **private** are ignored on class declarations. There is no concept of a namespace grouping of classes. There are no dots (.) in any class names.	The modifiers **public** and **private** can be used to modify class declarations. C# also has the keyword **internal**, which relates to how classes are grouped together in assembly files.	There is no concept of a **protected** class, only **protected** members of a class.

KEYWORD	X++	C#	COMMENTS
extends	A class declaration can inherit from another class by using the **extends** keyword.	A colon (:) is used where the keywords **extends** and **implements** are used in X++.	
final	A **final** method cannot be overridden in a derived class. A **final** class cannot be extended.	The keyword **sealed** on a class means the same thing that **final** means on an X++ class.	
implements	A class declaration can implement an **interface** by using the **implements** keyword.		
interface	An **interface** can specify methods that the class must implement.	An **interface** can specify methods that the class must implement.	
new	The **new** keyword is used to allocate a new instance of a class. Then the constructor is automatically called. Each class has exactly one constructor, and the constructor is named `new`. You can decide what parameters the constructor should input.	The **new** keyword is used to create a new instance of a class. Then the constructor is automatically called. Constructor methods themselves are not named `new`, they have the same name as the class. **Note:** The **new** keyword can also be used on a method, to modify the way in which the method overrides the same method in the base class.	Both X++ and C# assume a default constructor for classes that have no constructor explicitly written in their code.
null			No difference.
private and **protected**	The **private** and **protected** keywords can be used to modify the declaration of a class member.	The **private** and **protected** keywords can be used to modify the declaration of a class member.	
public	A method that is not modified with **public**, **protected**, or **private** has the default access level of **public**.	A method that is not modified with **public**, **protected**, or **private** has the default access level of **private**.	
static	A method can be **static**, but a field cannot.	Both methods and fields can be **static**.	

KEYWORD	X++	C#	COMMENTS

super	The **super** keyword is used in a derived class to access the same method on its base class.
	`void method2()`
	`{`
	`// Call method2 method`
	`// on the base class.`
	`super();`
	`}`
		The **base** keyword is used in a derived class to access various methods in its base class.
		`void method2()`
		`{`
		`// Call methods on`
		`// the base class.`
		`base.method2();`
		`base.method3();`
		`}`
			In C#, there is special syntax for using **base** to call the base constructor.

this	For a call from one instance method to another on the same object, a qualifier for the called method is required. The keyword **this** is available as a qualifier for the current object.	For a call from one instance method to another on the same object, a qualifier for the called method is not required. However, the **this** keyword is available as a qualifier for the current object. In practice, the keyword **this** can be helpful by displaying IntelliSense information.	
`finalize`	The `Object` class contains the `finalize` method. The `finalize` method is not **final**, and it can be overridden. The `finalize` method appears to resemble the `System.Object.Finalize` method in C#, but in X++ the `finalize` method has no special meaning of any kind. An object is automatically removed from memory when the last reference to the object stops referencing the object. For example, this can happen when the last reference goes out of scope or is assigned another object to reference.	The methods `Finalize` and `Dispose` are common on some types of classes. The garbage collector calls the `Finalize` and `Dispose` methods when it destroys and object.	In C#, the `System.GC.Collect` method in the .NET Framework can be called to start the garbage collector. There is no similar function in X++ because X++ uses a deterministic garbage collector.
`main`	Classes that are invoked from a menu have their `main` method called by the system.	Classes that are invoked from a command line console have their `Main` method called by the system.	

Comparison: Classes

When you use C# in the .NET Framework, classes are grouped into namespaces. Each namespace focuses on a functional area such as file operations or reflection. However, when you use the classes in X++, there are no visible groupings like a namespace.

Comparison: Classes about Reflection

In X++ the `TreeNode` class provides access to the Application Object Tree (AOT). The `TreeNode` class is the center of reflection functionality in X++. The `TreeNode` class and its methods can be compared to the `System.Reflection` namespace in the .NET Framework that C# uses.

The following table lists several classes that are available to you when you write C# code. These are .NET Framework classes. For this table, all C# classes are in the `System.Reflection` namespace unless otherwise specified. Each row shows the corresponding class, or class member, that is available to you when your write X++ code.

X++	C#	COMMENTS
`TreeNode`	`System .Assembly`	Assembly is the first class to use when a C# program must gather reflection information. Static methods on the X++ class `TreeNode` are the starting point for reflection in X++.
`TreeNode`	`System .Type`	Instance methods on `TreeNode` correspond to instance methods on `System.Type`.
`TreeNode .AOTgetSource`	`MethodInfo`	The `AOTgetSource` method returns several pieces of information together in one string. This includes the X++ source code in the method. In contrast, `MethodInfo` has a separate member for each piece of information.

`TreeNode .AOTfirstChild`
`TreeNode .AOTnextSibling`
`TreeNode .AOTiterator`
`AOTiterator`
	MethodInfo[] (an array)	In C#, the `GetMethods` method on `System.Type` returns an array of MethodInfo objects. You can loop through the array by the common technique of incrementing an indexer. In contrast, the X++ model is to navigate the tree control of the AOT. The `TreeNode` methods of `AOTfirstChild` and `AOTnextSibling` accomplish the navigation. As an equivalent alternative, the X++ `AOTiterator` class is designed to navigate the tree control of the AOT. A class node is the parent over several method nodes. The `AOTiterator` steps through child nodes, returning each as another `TreeNode` instance. Additional resources the `TreeNode` methods that are named `AOTparent` and `AOTprevious`.

`TreeNode .AOTgetProperty`
`TreeNode .AOTgetProperties`
`TreeNode .AOTname`
	`PropertyInfo`	In X++, the `AOTgetProperties` method returns a long string that contains name-value pairs for all the properties of the `TreeNode`. The `AOTname` method returns a string that contains only the value for the name property.

X++	C#	COMMENTS
`TreeNode .AOTsave` `TreeNode .AOTinsert`	`System .Reflection .Emit` (namespace of classes)	The `AOTsave` method applies changes from a `TreeNode` object in your X++ code to the AOT, and the changes are persisted. For a large code sample, see TreeNode.AOTsave Method.

Comparison: Classes about File IO

There are several classes that perform file input and output (IO) operations. In the .NET Framework that is used in C#, the counterparts to these classes reside in the `System.IO` namespace.

The following table lists several .NET Framework classes for C# that are in the `System.IO` namespace. Each row in the table shows the X++ class or method that best corresponds to the .NET Framework class.

X++	C#	COMMENTS
`BinaryIo`	`FileStream` `BinaryReader` `BinaryWriter`	X++ classes such as `BinaryIo` that extend from the abstract class `Io` serve as a stream, and they also serve as a reader and writer for that stream. In C#, the stream is a separate class from the class that has more specific read and write methods.
`TextBuffer`	`MemoryStream`	These classes contain an in-memory buffer, and some of the methods treat the buffer as if it were a file on the hard disk.
WINAPI::createDirectory WINAPI::folderExists WINAPI::removeDirectory	`Directory` `DirectoryInfo` `Path`	X++ can use static methods in the `WINAPI` class for many basic operating system functions that involve directories.
WINAPI::getDriveType	`DriveInfo` `DriveType`	These classes and methods are used to obtain drive related information.
WINAPI::copyFile WINAPI::createFile WINAPI::deleteFile WINAPI::fileExists	`File` `FileAttributes` `FileInfo`	X++ can use static methods in the `WINAPI` class for many basic operating system functions that involve files.
`CommaIo` `Comma7Io`	(No corresponding class.)	These X++ classes can generate files that Microsoft Excel can import. In X++ an EPPlus library reference is available for additional interaction with Excel.
`AsciiIo` `TextIo`	`FileStream` `TextReader` `TextWriter`	These classes use different code pages.
`Io`	`Stream` `StreamReader` `StreamWriter` `FileStream`	These are often used as base classes that other classes extend.

X++	C#	COMMENTS
`CodeAccessPermission` `FileIoPermission`	`System.Security.CodeAccessPermission` The namespace `System.Security.Permissions` includes the following classes: • `CodeAccessSecurityAttribute` • `FileIOPermissionAttribute` • `FileIOPermission` • `FileIOPermissionAccess`	The concepts and methods of `assert`, `demand`, and `revertAssert` apply to both languages. However, the `deny` and `revertDeny` methods that are available in C# are not available in X++.

X++, ANSI SQL Comparison: SQL Select

In X++, the SQL **select** statement syntax differs from the American National Standards Institute (ANSI) specification.

Single Table Select

The following table lists differences between the select statements of X++ SQL and ANSI SQL.

FEATURE	X++ SQL	ANSI SQL	COMMENTS
Table name on the **from** clause.	The **from** clause lists a record buffer instance that is declared from a table, such as from the `CustTable` table.	The **from** clause lists a table name, not the name of a buffer.	The record buffer has all the methods that the `xRecord` class has in X++.
Syntax sequence of the order by versus **where** clauses.	The order by clause must appear before the **where** clause. The order by clause must appear after the **from** or **join** clause. The group by clause must follow the same syntax positioning rules that the order by follows.	The order by clause must appear after the **where** clause. The **where** clause must appear after the **from** or **join** clause.	In both X++ and ANSI SQL, the **from** and **join** clauses must appear before the order by and **where** clauses.
Condition negation.	The exclamation mark ('!') is used for negation.	The **not** keyword is used for negation.	X++ does not support the syntax !like. Instead, you must apply the ! operator to a clause.
Wildcard characters for the **like** operator.	0 to many – Asterisk ('*') Exactly 1 – Question mark ('?')	0 to many – Percent sign ('%') Exactly 1 – Underbar ('_')	
Logical operators in the **where** clause.	And – && Or – ||	And – **and** Or – **or**	

Code Example

The following code example illustrates features in the previous table.

```
static void OByWhere452Job(Args _args)
{
    // Declare the table buffer variable.
    CustTable tCustTable;
    ;
    while
    SELECT * from tCustTable
        order by tCustTable.AccountNum desc
        where (!(tCustTable.Name like '*i*i*') &amp;&amp; tCustTable.Name like 'T?e *')
    {
        info(tCustTable.AccountNum + " , " + tCustTable.Name);
    }
}
/*** InfoLog output
Message (04:02:29 pm)
4010 , The Lamp Shop
4008 , The Warehouse
4001 , The Bulb
***/
```

X++ SQL Keywords

The following X++ SQL keywords are among those that are not part of ANSI SQL:

- crosscompany
- firstonly100
- forceliterals
- forcenestedloop
- forceplaceholders
- forceselectorder
- validtimestate

Join Clause

The following table lists differences about the **join** keyword of X++ SQL and ANSI SQL.

FEATURE	X++ SQL	ANSI SQL	COMMENTS
Columns list.	The columns in the columns list must all come from the table listed in the **from** clause, and not from any table in a **join** clause. Columns in the list cannot be qualified by their table name.	The columns in the columns list can come from any table in the **from** or **join** clauses. It helps others to maintain your code when you qualify the columns in the list with their table name.	For more information, see Select Statements on Fields.
Join clause syntax.	The **join** clause follows the **where** clause.	The **join** clause follows a table in the **from** clause.	In the X++ code example, the **join** criteria is an equality of `SalesPoolId` values.
Inner keyword.	The default **join** mode is inner join. There is no **inner** keyword.	The default **join** mode is inner join. The **inner** keyword is available to make the code explicit.	The **outer** keyword exists in both X++ SQL and ANSI SQL.
Left and **right** keywords.	The **left** and **right** keywords are not available. All joins are left.	The **left** and **right** keywords are available to modify the **join** keyword.	No comments.

FEATURE	X++ SQL	ANSI SQL	COMMENTS
Equality operator.	The double equal sign operator (' `==` ') is used to test for the equality of two values.	The single equal sign operator (' `=` ') is used to test for the equality of two values.	No comments.

Code Example

The following code example illustrates the **join** syntax in X++ SQL.

```
static void OByWhere453Job(Args _args)
{
    // Declare table buffer variables.
    CustTable tCustTable;
    SalesPool tSalesPool;
    ;
    while
    SELECT
            // Not allowed to qualify by table buffer.
            // These fields must be from the table
            // in the from clause.
            AccountNum,
            Name
        from tCustTable
            order by tCustTable.AccountNum desc
            where (tCustTable.Name like 'The *')
        join tSalesPool
            where tCustTable.SalesPoolId == tSalesPool.SalesPoolId
    {
        info(tCustTable.AccountNum + " , " + tCustTable.Name);
    }
}
```

Aggregate Fields

The following table lists some differences in how aggregate fields in the **select** column list are referenced between X++ SQL and ANSI SQL. Aggregate fields are those that are derived by functions such as **sum** or **avg**.

FEATURE	X++ SQL	ANSI SQL	COMMENTS
Aggregate field name alias.	The aggregate value is in the field that was aggregated.	You can use the **as** keyword to tag an aggregate field with a name alias. The alias can be referenced in subsequent code.	For more information, see Aggregate Functions: Differences Between X++ and SQL

Code Example

In the following code example, the call to the info method illustrates the way to reference aggregate fields (see `tPurchLine.QtyOrdered`).

```
static void Null673Job(Args _args)
{
    PurchLine tPurchLine;
    ;
    while
    select
        // This aggregate field cannot be assigned an alias name.
        sum(QtyOrdered)
        from tPurchLine
    {
        info(
            // QtyOrdered is used to reference the sum.
            "QtyOrdered:  " + num2str(tPurchLine.QtyOrdered,
            3,  // Minimum number of output characters.
            2,  // Required number of decimal places in the output.
            1,  // '.'  Separator to mark the start of the decimal places.
            2   // ','  The thousands separator.
            ));
    }
    info("End.");
}
/***
Message (12:23:08 pm)
QtyOrdered:  261,550.00
End.
***/
```

Other Differences

The following table lists other differences of the **select** statement between the X++ SQL and ANSI SQL.

FEATURE	X++ SQL	ANSI SQL	COMMENTS
The **having** keyword.	There is no **having** keyword.	The **having** keyword enables you to specify filter criteria for rows that are generated by the group by clause.	No comments.
Null results.	In a **while** select statement, if the **where** clause filters out all rows, no special count row is returned to report that.	In a **select**, if the **where** clause filters out all rows, a special count row is returned. The count value is 0.	No comments.
Cursors for navigating returned rows.	The while select statement provides cursor functionality. The alternative is to use the **next** keyword.	You can declare a **cursor** for looping through the rows that are returned from a **select** statement.	

From clause.	The **from** keyword is optional when no columns are listed and only one table is referenced. The following two syntax options are equivalent:
	`select \* from tCustTable;`
	`select tCustTable;`
		A **select** statement cannot read from a table unless the **from** clause is used.	In X++ SQL, the simple **select** statement fills the table buffer variable with the first row that was returned. This is illustrated by the following code fragment:
			`select \* from tCustTable;`
			`info(tCustTable.Name);`

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

X++ syntax

2/18/2021 • 30 minutes to read • Edit Online

This topic contains the syntax reference for X++.

X++ Keywords

The X++ keywords shown in the following table are reserved. These keywords cannot be used for any other purpose.

RESERVED WORD	DESCRIPTION	MORE INFORMATION
!	Not.	Relational Operators
!=	Inequality operator (not equal to).	Relational Operators
#	Prefix on macro names.	How to: Use #define and #if to Test a Macro
&	Binary AND.	Arithmetic Operators
&&	Logical AND.	Relational Operators
(	Function call operator, which indicates the beginning of the function call.	
)	Function call operator, which indicates the end of the function call.	
*	Multiply. The asterisk (*) is also used in X++ SQL. One use is to signify all fields from the tables on a `select` statement. Another use is as a wildcard with the `like` operator, to signify 0 to many characters of any kind. The `like` operator also uses the ? character.	Arithmetic Operators
^	Binary XOR.	Arithmetic Operators
|	Binary OR.	Arithmetic Operators
||	Logical OR.	Relational Operators
~	Not.	Arithmetic Operators
+	Plus.	Arithmetic Operators
++	Increment.	Assignment Operators
+=	Additive assignment.	Assignment Operators

RESERVED WORD	DESCRIPTION	MORE INFORMATION
,	Comma operator. Expressions separated by commas are evaluated left-to-right.	
-	Minus.	Arithmetic Operators
--	Decrement operator.	Assignment Operators
-=	Subtractive assignment.	Assignment Operators
.	Class member access operator, for example, `formRun.run` accesses the `run` method of an object of the class type `FormRun`.	
/	Divide.	Arithmetic Operators
	Escape in strings. Escapes extra quotation marks, and certain letters such as \t for tab.	
@	Escape of keywords. For example, str @abstract; would fail to compile without the @ sign. Also affects literal strings, by negating the effect of the \ escape character, and by enabling the string to span more than one line in the source code. The new line is represented by one character of hexadecimal 0x0A, which is commonly called a line feed. No carriage return character of hexadecimal 0x0D is included, as in 0x0D0A.	
:	Field declaration or label specifier. The colon (:) character is also used on the `switch` statement.	
::	Used to call static (class) methods: ClassName::methodName.	
;	Terminates statements. Used in `for` loops or as a separator of statements.	
<	Less than.	Relational Operators
<<	Left shift.	Arithmetic Operators
<=	Less than or equal.	Arithmetic Operators
=	Assignment operator. The argument to the left of "=" is set to the value of the argument to the right.	Assignment Operators

RESERVED WORD	DESCRIPTION	MORE INFORMATION
==	Returns true if both expressions are equal.	Relational Operators
>	Greater than.	Relational Operators
>=	Greater than or equal.	Relational Operators
>>	Right shift.	Arithmetic Operators
?	Ternary operator. The question mark (?) character is also used by the `like` operator to signify exactly one character of any kind. The `like` operator also uses the character.	Ternary Operator (?)
[	Array declarator, open. Must be used with "**]**".	
]	Array declarator, close. Must be used with "**[**".	
{	Indicates the beginning of a number of statements. The last of these statements must be followed by a "**}**".	
}	Indicates the end of a number of statements. A "**{**" must appear before the first of these statements.	
abstract	Class and method modifier. An **abstract** class cannot be constructed with the **new** keyword. An **abstract** method cannot be called. A table can also be modified as abstract by setting its Abstract property to Yes in the AOT, or by using the `DictTable` class. The Abstract property defaults to No, and it cannot be set unless the table is extended by another table. Each row in an abstract table must have a dependent row in a derived table. This means that each row in an abstract table has a value greater than 0 (zero) in its InstanceRelationType property field. There are no other effects from marking a table as abstract. Informally, programmers often use the term concrete to describe a class that is non-**abstract**.	Method Modifiers Table Inheritance Overview
anytype	The method can return any data type.	Anytype

RESERVED WORD	DESCRIPTION	MORE INFORMATION
as	Needed when you assign a base class variable to a derived class variable. For example, given a `Derived` class that **extends** a `Base` class, the statement `myDerived = myBase as Derived;` avoids a compiler error by using the **as** keyword. This keyword also applies when you assign a base table variable to a derived table variable.	Expression Operators: Is and As for Inheritance
asc	An option on the `order by` or `group by` clause in a `select` statement. The sorting is ascending.	Select Statement Syntax
at	Specifies the position of a print window.	Print Statements
avg	Returns the average of the fields from the rows specified by the `group by` clause in a `select` statement.	Select Statement Syntax
break	Immediate exit from code block.	Break Statements
breakpoint	Represents a breakpoint that is set for debugging purposes. To set a breakpoint in your code, write: `breakpoint;`	
by	Part of a reserved term, such as group by and order by.	
byref	Specifies that the parameter being passed to the called method is being passed by reference (address), instead of by value. **Byref** is used in X++ when calling a .NET method that takes a parameter by reference (such as with the C# keywords **out** or **ref**).	How to: Use the byref Keyword for CLR Interop.
case	Selection within a `switch` statement.	Switch Statements
catch	Used in exception handling.	Exception Handling with try and catch Keywords
changeCompany	Changes database settings to another company.	Change Company Design Pattern
class	Declares a class.	Classes in X++
client	Method modifier.	Method Modifiers
container	Specifies a variable of type `container` .	Containers

RESERVED WORD	DESCRIPTION	MORE INFORMATION
continue	Forces the next iteration of a loop.	Continue Statements
count	Returns the number of records from the rows specified by the `group by` clause in a `select` statement.	Select Statement Syntax
crossCompany	Causes a `select` statement to return data for all companies that the user is authorized to read from.	Cross-Company X++ Code Basics
date	Specifies a variable of type `date`.	Dates
default	Default case within `switch` statements.	Switch Statements
delegate	A class member that is able to store multiple references to methods in other classes, and to call all those methods when prompted to do so. A delegate can store references to various kinds of methods including the following: • static methods on X++ classes • instance methods on X++ classes • methods on .NET Framework classes	Event Terminology and Keywords X++, C# Comparison: Event
delete_from	Allows you to delete multiple records from the database at the same time.	delete_from
desc	An option on the `order by` or `group by` clause in a `select` statement. The sorting is descending.	Select Statement Syntax
display	Method modifier.	Method Modifiers
div	Integer division.	Arithmetic Operators
do	Beginning of a `do...while` loop.	Do...while Loops
edit	Method modifier.	Method Modifiers
else	Conditional execution (`if...else`).	if and if ... else Statements
eventHandler	Must be used each time you either add or delete a method reference from a delegate by using the += or -= operator. For example: myDelegate += eventHandler(OtherClass::myStaticMethod);	Event Terminology and Keywords X++, C# Comparison: Event

RESERVED WORD	DESCRIPTION	MORE INFORMATION
exists	Used with `join` clauses in `select` statements.	Select Statement Syntax
extends	A class or interface declaration clause. If your class does not explicitly extend another class, your class is considered to extend the `Object` class (as if you had written "extends Object").	Creating a Subclass
false	Boolean literal.	Booleans
final	Class and method modifier.	Method Modifiers
firstFast	Used in `select` statements to speed up the fetch for the first row.	Select Statement Syntax
firstOnly	Used in `select` statements to fetch only the first record. The `firstOnly` keyword does not guarantee that a maximum of one record is retrieved by an X++ SQL `select` statement. If the AOS can use the `EntireTable` cache to satisfy the data demands of the `select` statement, the `firstOnly` keyword is ignored.	Select Statement Syntax Set-based Caching
firstOnly10	Same as **firstOnly**, except returns 10 rows instead of one.	
firstOnly100	Same as **firstOnly**, except returns 100 rows instead of one.	
firstOnly1000	Same as **firstOnly**, except returns 1000 rows instead of one.	
flush	Clears an entire table cache. Here is the syntax for the `flush` statement: `YourTable ytBuffer;` `flush ytBuffer;`	Set-based Caching
for	For loop iteration.	For Loops
forceLiterals	Used in `select` statements to reveal actual values that are used in `where` clauses to the Microsoft SQL Server database at the time of optimization.	Select Statement Syntax
forceNestedLoop	Forces the SQL Server database to use a nested-loop algorithm to process a particular SQL statement containing a `join`.	Select Statement Syntax

RESERVED WORD	DESCRIPTION	MORE INFORMATION
forcePlaceholders	Used in `select` statements to instruct the kernel not to reveal the actual values used in `where` clauses to the Microsoft SQL Server database at the time of optimization.	Select Statement Syntax
forceSelectOrder	Forces the SQL Server database to access the tables in a join in the specified order.	Select Statement Syntax
forUpdate	Selects records exclusively for update. The operation to be performed on the records that are fetched is an update. Depending on the underlying database, the records may be locked for other users.	Select Statement Syntax
from	Part of a `select` statement. The `from` clause specifies the table in which the columns exists.	Select Statement Syntax
group	Part of the `group by` clause in a `select` statement.	Select Statement Syntax
if	Conditional execution.	if and if ... else Statements
implements	Implements an interface.	Interfaces Overview
insert_recordset	Copies data from one or more tables into one resulting destination table on a single server trip.	insert_recordset
int	Specifies a variable of type `integer` (32-bit).	Integers
int64	Specifies a variable of type `integer` (64-bit).	Integers
interface	Interface declaration.	Interfaces Overview
is	Asks whether the object referenced by a class variable either inherits from the given class or is of the given class. For example, given a `Derived` class that **extends** a `Base` class, the expression `(myDerived is Base)` returns **true**. This keyword applies to class inheritance and table inheritance.	Expression Operators: Is and As for Inheritance
join	Tables are joined on columns common to both tables. You can generate a single result set based on multiple tables through the use of joins.	Select Statement Syntax

RESERVED WORD	DESCRIPTION	MORE INFORMATION

like	Tests for matches by pattern, with wildcard symbols * and ?. The string on the right side of the `like` operator must use four backslash characters to represent one backslash. Examples follow:
	• ("" like "") //Resolves to false.
	• ("" like "*") //Resolves to true.
		Relational Operators

maxof	Returns the maximum of the fields from the rows specified by the `group by` clause.	Select Statement Syntax
minof	Returns the minimum of the fields from the rows specified by the `group by` clause.	Select Statement Syntax
mod	Returns the integer remainder of the left expression1 divided by the right expression2. Informally this is sometimes called the modulo operator. `((12 mod 7) == 5)` is true.	
new	Operator. Creates an instance of an anonymous class that is assignment-compatible with the named class/interface reference variables, or allocates memory for an array.	
next	Fetches the next record in a table.	
noFetch	Indicates that no records are to be fetched at present.	Select Statement Syntax
notExists	Used with `join` clauses in `select` statements.	Select Statement Syntax
null	Symbolic constant.	
optimisticLock	Forces a statement to run with optimistic concurrency control, even if a different value is set on the table.	Select Statement Syntax
order	Part of the `order by` clause in a `select` statement.	Select Statement Syntax
outer	outer join.	Select Statement Syntax
pause	Halts the execution of a job. The user is asked to state whether execution should continue.	Select Statements

RESERVED WORD	DESCRIPTION	MORE INFORMATION
pessimisticLock	Forces a statement to run with pessimistic concurrency control, even if a different value is set on the table.	Select Statement Syntax
print	Allows you to display output on the screen.	Print Statements
private	Method access modifier.	Method Access Control
protected	Method access modifier.	Method Access Control
public	Method access modifier.	Method Access Control
real	Specifies a variable of type `real`.	Reals
repeatableRead	Specifies that no other transactions can modify data that has been read by logic inside the current transaction, until after the current transaction completes. An explicit transaction completes at either **ttsAbort** or at the outermost **ttsCommit**. For a stand-alone **select** statement, the transaction duration is the duration of the **select** command. However, the database sometimes enforces the equivalent of **repeatableRead** in individual **select** statements even without this keyword appearing in your X++ code (depending on how the database decides to scan the tables).	For more information, see the documentation for the underlying relational database product.
retry	Used in exception handling.	Exception Handling with try and catch Keywords
return	Exits from a method.	Declaration of Methods
reverse	Records are returned in reverse order.	Select Statement Syntax
select	The `select` clause designates which columns or views are shown in the result set.	Select Statements
server	Method modifier.	Method Modifiers
setting	Used with the update_recordset command.	update_recordset
static	Static methods may not refer to instance variables (only to static variables); may be invoked by using the class name rather than on an instance of the class ("`MyClass.aStaticProcedure`").	Method Modifiers

RESERVED WORD	DESCRIPTION	MORE INFORMATION
str	Specifies a variable of type `string`.	Strings
sum	Returns the sum of the fields from the rows specified by the `group by` clause in a `select` statement.	Select Statement Syntax
super	Calls the method that was overridden by the current method.	Table Methods
switch	Switch selection statement.	Switch Statements
tableLock	Obsolete; **tableLock** is no longer available.	
this	A reference to the current instance of the class. Used in X++ code inside a method of the class. Used to reference *method* members of the class, but not *field* members of the class. `public str getFullName()` { // Next statement fails to compile without 'this.'. `return this.concatenateFirstAndLastNames();` }	Loosely similar to the system variable that is named `element`. You use `element` in form control methods to reference the containing form. For more information, see Using Variables with Forms.
throw	Used in exception handling.	Exception Handling with try and catch Keywords
true	Boolean literal.	Booleans
try	Used in exception handling.	Exception Handling with try and catch Keywords
ttsAbort	Discards all changes in the current transaction.	Transaction Integrity
ttsBegin	Marks the beginning of a transaction.	Transaction Integrity
ttsCommit	Marks the end of a transaction.	Transaction Integrity
update_recordset	Allows the manipulation of row sets within one operation.	update_recordset
validTimeState	Filters rows that are retrieved from a valid time state table by an X++ SQL `select` statement. For example: select validTimeState(myDateEffective) * from xMyTable; ...or... select validTimeState(myDateFrom, myDateTo) * from xMyTable;	Effects of Valid Time State Tables on Read and Write Operations
void	Identifies a method that does not return a value.	Declaration of Methods

RESERVED WORD	DESCRIPTION	MORE INFORMATION
where	Part of a `select` statement. The `where` clause specifies the conditions to be satisfied; that is, the rows that you want to include in the result.	Select Statement Syntax
while	Iteration statement. Executes a statement or block repeatedly when a test condition is true.	While Loops while select Statements
window	Allows you to alter the size of the output window.	Print Statements

Expressions Syntax

An expression in X++ is used in either a mathematical or logical way. Expressions are built on the data types of the language; that is, an expression returns a value of some type. This value can be used in calculations, assignments, conditional statements, and so on.

EBNF Description of Expressions in X++

TERM		DEFINITION
Expression	=	Simple-expression [RelationalOperator Simple-expression]
RelationalOperator	=	=
Simple-expression	=	Simple-expression [+
Term	=	Compfactor { Mult-operator CompFactor }
Mult-operator	=	*
CompFactor	=	[!] [-
Factor	=	Literal
Enum	=	EnumName :: Literal
Variable	=	Identifier [[Expression]] [. Expression]
FunctionCall	=	[Expression (.
If-expression	=	Expression ? Expression : Expression

Semantic restrictions apply on the preceding syntax. You cannot call any method using the :: operator. Similarly, you cannot use the **this** keyword without an active object; that is, if you are not within a method and so on.

Examples

EXAMPLE OF EXPRESSION	DESCRIPTION
`1`	An integer literal.
NoYes::No	An enum-reference.
`A`	A variable-reference.
Debtor::Find("1")	A static method-call (returns a customer variable).
(A > 3 ? true : false)	An if-expression that returns **true** or **false**.
(select CustTable where CustTable.Account == "100").NameRef	A select-expression. Returns the nameref field in the customer table. This is a string.
A >= B	A logical expression. Returns **true** or **false**.
A + B	An arithmetic expression. Sums A and B.
A + B / C	Calculates B/C, and then adds this to A.
~A + this.Value()	Sums binary not A and the result of the method-call Value on the object in scope (this).
Debtor::Find("1").NameRef	Returns the NameRef field of the found customer record.
Debtor::Find("1").Balance()	A method call to `Balance` in the customer table (Debtor::Find returns a customer). Returns the balance of the customer with account number 1.

EBNF Overview

Extended Backus Naur Form (EBNF) is a metalanguage and is used in this guide to describe the language syntax. An EBNF definition consists of production rules, nonterminals, and terminals. The key terms are shown in the following table.

KEY TERMS	EXAMPLE	DESCRIPTION
Terminals	Work_Team	A terminal is one character or a string of characters that never change.
Nonterminals	`Employee`	A nonterminal is a description of part of a valid sentence in the language that is defined either by a production rule or a textual description. A nonterminal symbol can always be expanded to one or more terminal symbols.
Production rules	Employee = Developer	Tester

Example

Work_Team = Manager Employee {, Employee} Employee = Developer | Tester This example defines a Work_Team as consisting of a `Manager` and one or more `Employees`. An `Employee` is defined as being a

`Developer`, or a `Tester`. The symbols used in the example are described in the following table.

Special Symbols in EBNF

SYMBOL	DESCRIPTION
(*Expression*)	Parentheses hold the symbols (terminals and nonterminals) together. They can be placed anywhere on the right side of a production rule.
Expression1	*Expression2*
[*Expression*]	Optional: The items between [and] are optional. All or none of the items in the brackets are included.
{Expression}	Repeat: The items between { and } are optional, but can be repeated as many times as necessary.

For example, if the accessories you buy for your bicycle consist of a saddle, water-bottle holders, bells, and horns, and you could have either a bell or a horn, and zero, one, or more water bottle holders, and exactly one saddle, this could be expressed as: Bicycle_Accessories = saddle [bell | horn] {water_bottle_holders} This grammar defines the following possibilities: `saddle` `saddle bell` `saddle horn` saddle water_bottle_holder saddle bell water_bottle_holder saddle bell water_bottle_holder water_bottle_holder And so on.

X++ Grammar

This topic shows the formal grammar of the X++ language.

How to Interpret the Formal BNF Grammar

This section describes the grammar of X++ in Backus Naur Form (BNF). A small example of BNF is described here.

```
AA ::= BB  CC_SYM
BB ::= JJ_SYM
   ::= KK_SYM
```

`AA` is the name of a production rule. An `AA` requires a `BB`, followed by a CC_SYM. A `BB` is also a production rule. Therefore, `BB` is not a terminal. `BB` must be either a JJ_SYM or a KK_SYM. Both JJ_SYM and KK_SYM are terminals because they are not the names of any other production rules. CC_SYM is also a terminal.

In the BNF for X++ grammar, most of the terminals have _SYM as the suffix of their name.

The Formal X++ Grammar in BNF

This section contains the BNF that defines the grammar of X++.

```
    CMPL_UNIT ::= RETTYPEID  FUNC_HDR  FUNC_HEAD  BODY
              ::= RETTYPEID  DATA_HDR  CLASS_DECL
              ::= EXPR_HDR  IF_EXPR  SEMIOPT
              ::= RETTYPEID  FUNC_HDR  EVENT_DECL  BODY
    SEMIOPT ::= SEMICOLON_SYM
            ::=
    CLASS_DECL ::= CLASS_HEADER  LEFTBR_SYM  DCL_EVENTMAP  DCL_LIST  RIGHTBR_SYM
    CLASS_HEADER ::= ATTRIBUTE_DEF  CLASS_MODIFIERS  CLASSORINTERFACE  STD_ID  EXTENDS  IMPLEMENTS
    ATTRIBUTE_DEF ::= LEFT_BRKT_SYM  ATTRIBUTE_INIT  ATTRIBUTE_LIST  RETTYPEID  RGHT_BRKT_SYM
                  ::=
    ATTRIBUTE_INIT ::=
```

```
ATTRIBUTE_LIST ::= ATTRIBUTE
               ::= ATTRIBUTE_LIST  LIST_SEP_SYM  ATTRIBUTE
ATTRIBUTE ::= STD_ID
          ::= ATTRIBUTE_WITH_ARGS_BEGINS  ATTRIBUTE_WITH_ARGS_ENDS
ATTRIBUTE_WITH_ARGS_BEGINS ::= STD_ID  LEFT_PAR_SYM
ATTRIBUTE_WITH_ARGS_ENDS ::= ATTRIBUTE_ARGS  RGHT_PAR_SYM
ATTRIBUTE_ARGS ::= ATTRIBUTE_CONSTANT
               ::= ATTRIBUTE_ARGS  LIST_SEP_SYM  ATTRIBUTE_CONSTANT
ATTRIBUTE_CONSTANT ::= INT_SYM
                   ::= DBL_SYM
                   ::= STR_SYM
                   ::= DATE_SYM
                   ::= DATETIME_SYM
                   ::= STD_ID  DBLCOLON_SYM  STD_ID
                   ::= TRUE_SYM
                   ::= FALSE_SYM
                   ::= INT64_SYM
                   ::= ATTRIBUTE_INTRINSIC
ATTRIBUTE_INTRINSIC ::= INTRI_ID  LEFT_PAR_SYM  IARGS  RGHT_PAR_SYM
CLASSORINTERFACE ::= CLASS_SYM
                 ::= INTERFACE_SYM
CLASS_MODIFIERS ::= CLASS_MODS
                ::=
CLASS_MODS ::= CLASS_MODIFIER
           ::= CLASS_MODS  RETTYPEID  CLASS_MODIFIER
CLASS_MODIFIER ::= PUBLIC_SYM
               ::= FINAL_SYM
               ::= STATIC_SYM
               ::= ABSTRACT_SYM
               ::= PRIVATE_SYM
EXTENDS ::= EXTENDS_SYM  STD_ID
        ::=
IMPLEMENTS ::= IMPLEMENTS_SYM  IMPLEMENTLIST
           ::=
IMPLEMENTLIST ::= STD_ID
              ::= IMPLEMENTLIST  LIST_SEP_SYM  STD_ID
DCL_EVENTMAP ::=
EVENT_DECL ::= ATTRIBUTE_DEF  EVENT_HEADER  PARM_DCL_LIST
EVENT_HEADER ::= EVENT_MODIFIER  VOID_TYPE_SYM  STD_ID
EVENT_MODIFIER ::= EVENT_SYM
FUNC_HEAD ::= ATTRIBUTE_DEF  FUNCNAME  PARM_DCL_LIST
FUNCNAME ::= FUNCTYPE  STD_ID
FUNCTYPE ::= FUNC_MODIFIERS  DECL_TYPE
FUNC_MODIFIERS ::= FUNC_MODS
               ::=
FUNC_MODS ::= RETTYPEID  FUNC_MODIFIER
          ::= FUNC_MODS  RETTYPEID  FUNC_MODIFIER
FUNC_MODIFIER ::= PUBLIC_SYM
              ::= PRIVATE_SYM
              ::= PROTECTED_SYM
              ::= FINAL_SYM
              ::= STATIC_SYM
              ::= ABSTRACT_SYM
              ::= DISPLAY_SYM
              ::= EDIT_SYM
              ::= SERVER_SYM
              ::= CLIENT_SYM
BODY ::= LEFTBR_SYM  DCL_FUNC_LIST  SEMIOPT  SECAUTHZCHECK  STMTLIST  SECAUTHZEND  RIGHTBR_SYM
SECAUTHZCHECK ::=
SECAUTHZEND ::=
RETTYPEID ::=
FUNCTION_DEF ::= FUNC_HEADER  PARM_DCL_LIST  LOCAL_BODY
FUNC_HEADER ::= DECL_TYPE  STD_ID
PARM_DCL_LIST ::= RETTYPEID  PARM_START  PARM_LIST_OPT  RGHT_PAR_SYM  RETTYPEID
PARM_START ::= LEFT_PAR_SYM
PARM_LIST_OPT ::= PARM_LIST
              ::=
PARM_LIST ::= DCL_INIT
          ::= PARM_LIST  LIST_SEP_SYM  DCL_INIT
```

```
                ::= PARM_LIST  LIST_SEP_SYM  DCL_INIT
    LOCAL_BODY ::= LEFTBR_SYM  DCL_LIST  SEMIOPT  STMTLIST  RETTYPEID  RIGHTBR_SYM
    DCL_LIST ::= DCL_LIST2
             ::=
    DCL_LIST2 ::= DCL_STMT
              ::= DCL_LIST2  DCL_STMT
    DCL_FUNC_LIST ::= DCL_FUNC_LIST2
                  ::=
    DCL_FUNC_LIST2 ::= DCL_STMT
                   ::= FUNCTION_DEF
                   ::= DCL_FUNC_LIST2  DCL_STMT
                   ::= DCL_FUNC_LIST2  FUNCTION_DEF
    DCL_STMT ::= DCL_INIT_LIST  RETTYPEID  SEMICOLON_SYM
    DCL_INIT_LIST ::= DCL_INIT
                  ::= DCL_CLIST  ASG_CLAUSE
    DCL_CLIST ::= DCL_INIT_LIST  LIST_SEP_SYM  STD_ID  ARR_DCL_IDX
    DCL_INIT ::= DECL  ASG_CLAUSE
    DECL ::= DECL_TYPE  STD_ID  ARR_DCL_IDX
    DECL_TYPE ::= STR_TYPE_SYM  STR_LEN
              ::= INT_TYPE_SYM
              ::= DBL_TYPE_SYM
              ::= DATE_TYPE_SYM
              ::= DATETIME_TYPE_SYM
              ::= TYPE_ID
              ::= QUEUE_TYPE_SYM
              ::= VOID_TYPE_SYM
              ::= ANY_TYPE_SYM
              ::= GUID_TYPE_SYM
              ::= INT64_TYPE_SYM
              ::= CLR_TYPE
    CLR_TYPE ::= CLR_NAMESPACE  TYPE_ID  CLR_ARRAY_TYPE_EXT
             ::= CLR_NAMESPACE  CLR_TYPE
    CLR_NAMESPACE ::= TYPE_ID  PERIOD_SYM
    CLR_ARRAY_TYPE_EXT ::= CLR_ARRAY_SPEC
                       ::=
    CLR_ARRAY_SPEC ::= CLR_ARRAY_PART
                   ::= CLR_ARRAY_SPEC  CLR_ARRAY_PART
    CLR_ARRAY_PART ::= CLR_ARRAY_LEFT_PART  CLR_RECTANGULAR_LIST  RGHT_BRKT_SYM
    CLR_ARRAY_LEFT_PART ::= LEFT_BRKT_SYM
    CLR_RECTANGULAR_LIST ::= CLR_COMMA_LIST
                         ::=
    CLR_COMMA_LIST ::= LIST_SEP_SYM
                   ::= CLR_COMMA_LIST  LIST_SEP_SYM
    STR_LEN ::= INT_SYM
            ::=
    ARR_DCL_IDX ::= LEFT_BRKT_SYM  RANGE  ARRAY_MEM  RGHT_BRKT_SYM
                ::=
    RANGE ::= IF_EXPR
          ::=
    ARRAY_MEM ::= LIST_SEP_SYM  IF_EXPR
              ::=
    ASG_CLAUSE ::= INIT_START  IF_EXPR
               ::=
    INIT_START ::= ASG_SYM
    ASG_STMT ::= LVAL_FLD  ASSIGN  IF_EXPR
             ::= LVAL_LIST  ASG_SYM  IF_EXPR
             ::= LVAL_FLD  ASG_INC_DEC
             ::= ASG_INC_DEC  LVAL_FLD
             ::= LVAL_FLD  ASG_EVENT_HANDLER
    ASSIGN ::= ASG_SYM
           ::= ASGINC_SYM
           ::= ASGDEC_SYM
    ASG_INCDEC ::= ASGINC_SYM
               ::= ASGDEC_SYM
    ASG_EVENT_HANDLER ::= ASG_INCDEC  EVENTHANDLER_SYM  LEFT_PAR_SYM  QUALIFIER  STD_ID  RGHT_PAR_SYM
      ::= ASG_INCDEC  EVENTHANDLER_SYM  LEFT_PAR_SYM  STD_ID  DBLCOLON_SYM  STD_ID  RGHT_PAR_SYM
      ::= ASG_INCDEC  EVENTHANDLER_SYM  LEFT_PAR_SYM  QUALIFIER  EVAL_CLR_TYPE  DBLCOLON_SYM  STD_ID
RGHT_PAR_SYM
    ASG_INC_DEC ::= INC_SYM
                ::= DEC_SYM
```

```
            ::= DEC_SYM
LVAL_FLD ::= FIELD
LVAL_START ::= LEFT_BRKT_SYM
LVAL_LIST ::= LVAL_START  LVALUES  RGHT_BRKT_SYM
LVALUE ::= FIELD
LVALUES ::= LVALUE
        ::= LVALUES  NEXTLVAL  LVALUE
NEXTLVAL ::= LIST_SEP_SYM
IF_EXPR ::= COND_TRUE  IF_EXPR
        ::= BOOL_EXPR
COND_TRUE ::= COND_TEST  IF_EXPR  COLON_SYM
COND_TEST ::= BOOL_EXPR  QUEST_SYM
BOOL_EXPR ::= BOOL_EXPR  LOGOP  EXPR
          ::= EXPR
LOGOP ::= AND_SYM
      ::= OR_SYM
EXPR ::= SMPL_EXPR  RELOP  SMPL_EXPR
     ::= SMPL_EXPR  AS_SYM  STD_ID
     ::= SMPL_EXPR  IS_SYM  STD_ID
     ::= SMPL_EXPR  AS_SYM  EVAL_CLR_TYPE
     ::= SMPL_EXPR  IS_SYM  EVAL_CLR_TYPE
     ::= SMPL_EXPR
RELOP ::= LT_SYM
      ::= LE_SYM
      ::= EQ_SYM
      ::= NE_SYM
      ::= GT_SYM
      ::= GE_SYM
      ::= LIKE_SYM
SMPL_EXPR ::= SMPL_EXPR  ADDOP  TERM
          ::= TERM
ADDOP ::= PLUS_SYM
      ::= MINUS_SYM
      ::= PHYSOR_SYM
TERM ::= TERM  MULOP  CMPL_FACT
     ::= CMPL_FACT
MULOP ::= MULT_SYM
      ::= DIV_SYM
      ::= MOD_SYM
      ::= INTDIV_SYM
      ::= SHIFTL_SYM
      ::= SHIFTR_SYM
      ::= PHYSAND_SYM
      ::= PHYSXOR_SYM
CMPL_FACT ::= NOT_SYM  SGND_FACT
          ::= SGND_FACT
SGND_FACT ::= SIGNOP  FACTOR
          ::= FACTOR
SIGNOP ::= UMINUS_SYM
       ::= PHYSNOT_SYM
FACTOR ::= LEFT_PAR_SYM  IF_EXPR  RGHT_PAR_SYM
       ::= CONSTANT
       ::= FIELD
       ::= DIRSEARCH
       ::= FUNCTION
       ::= INTRINSICS
       ::= EVAL
       ::= CONLITTERAL
       ::= NEW_CLR_ARRAY
NEW_CLR_ARRAY ::= NEW_SYM  EVAL_CLR_TYPE  NEW_CLR_ARRAY_PART  LEFT_PAR_SYM  RGHT_PAR_SYM
NEW_CLR_ARRAY_PART ::= CLR_SIZED_ARRAY  CLR_NOSIZED_ARRAY_SPEC
CLR_SIZED_ARRAY ::= LEFT_BRKT_SYM  CLR_SMPL_EXPR_COMMA_LIST  RGHT_BRKT_SYM
CLR_SMPL_EXPR_COMMA_LIST ::= SMPL_EXPR
  ::= CLR_SMPL_EXPR_COMMA_LIST  LIST_SEP_SYM  SMPL_EXPR
CLR_NOSIZED_ARRAY_SPEC ::= CLR_NOSIZED_ARRAY_LIST
                       ::=
CLR_NOSIZED_ARRAY_LIST ::= CLR_NOSIZED_ARRAY
                       ::= CLR_NOSIZED_ARRAY_LIST  CLR_NOSIZED_ARRAY
CLR_NOSIZED_ARRAY ::= LEFT_BRKT_SYM  CLR_EMPTY_COMMA_LIST  RGHT_BRKT_SYM
```

```
CLR_EMPTY_COMMA_LIST ::= CLR_EMPTY_RECT_COMMA_LIST
                     ::=
CLR_EMPTY_RECT_COMMA_LIST ::= LIST_SEP_SYM
                          ::= CLR_EMPTY_RECT_COMMA_LIST  LIST_SEP_SYM
CONLITTERAL ::= LEFT_BRKT_SYM  IF_EXPR  EXPR_LIST  RGHT_BRKT_SYM
CONSTANT ::= INT_SYM
         ::= DBL_SYM
         ::= STR_SYM
         ::= DATE_SYM
         ::= DATETIME_SYM
         ::= STD_ID  DBLCOLON_SYM  STD_ID
         ::= TRUE_SYM
         ::= FALSE_SYM
         ::= NULL_SYM
         ::= INT64_SYM
         ::= QUALIFIER  EVAL_CLR_TYPE  DBLCOLON_SYM  STD_ID
         ::= QUALIFIER  STD_ID  DBLCOLON_SYM  STD_ID
DIRSEARCH ::= DIRS_HEADER  PERIOD_SYM  STD_ID  ARR_IDX
          ::= DIRS_HEADER  PERIOD_SYM  FLD_NUM  ARR_IDX
DIRS_HEADER ::= LEFT_PAR_SYM  SET_DIRS  FIND_JOIN  RGHT_PAR_SYM
SET_DIRS ::=
FIELD ::= QUALIFIER  STD_ID  ARR_IDX
      ::= QUALIFIER  FLD_NUM  ARR_IDX
      ::= STD_ID  ARR_IDX
QUALIFIER ::= EVAL  PERIOD_SYM
          ::= STD_ID  PERIOD_SYM
FLD_NUM ::= LEFT_PAR_SYM  IF_EXPR  RGHT_PAR_SYM
ARR_IDX ::= LEFT_BRKT_SYM  SMPL_EXPR  RGHT_BRKT_SYM
        ::=
EXPR_LIST ::= EXPR_LIST2
          ::=
EXPR_LIST2 ::= LIST_SEP_SYM  IF_EXPR
           ::= EXPR_LIST2  LIST_SEP_SYM  IF_EXPR
FUNCTION ::= FUNC_ID  LEFT_PAR_SYM  EVAL_FUNCTION_NAME  PAR_LIST  RGHT_PAR_SYM
EVAL_FUNCTION_NAME ::=
EVAL_NAME ::= EVAL_ID  LEFT_PAR_SYM
          ::= STD_ID  LEFT_PAR_SYM
          ::= STD_ID  DBLCOLON_SYM  STD_ID  LEFT_PAR_SYM
          ::= SUPER_SYM  LEFT_PAR_SYM
          ::= NEW_SYM  STD_ID  LEFT_PAR_SYM
          ::= NEW_SYM  EVAL_CLR_TYPE  LEFT_PAR_SYM
          ::= QUALIFIER  EVAL_CLR_TYPE  DBLCOLON_SYM  STD_ID  LEFT_PAR_SYM
          ::= QUALIFIER  STD_ID  LEFT_PAR_SYM
          ::= QUALIFIER  STD_ID  DBLCOLON_SYM  STD_ID  LEFT_PAR_SYM
EVAL_CLR_TYPE ::= NAMESPACE  STD_ID
              ::= NAMESPACE  EVAL_CLR_TYPE
NAMESPACE ::= STD_ID  PERIOD_SYM
EVAL ::= EVAL_NAME  PAR_LIST  RGHT_PAR_SYM
PAR_LIST ::= PRM_LIST
         ::=
PRM_LIST ::= PAR_ELEM
         ::= PRM_LIST  LIST_SEP_SYM  PAR_ELEM
PAR_ELEM ::= IF_EXPR
         ::= BYREF_SYM  FIELD
INTRINSICS ::= INTRI_ID  LEFT_PAR_SYM  IARGS  RGHT_PAR_SYM
IARGS ::= STD_ID
      ::= STR_SYM
      ::= STD_ID  LIST_SEP_SYM  STD_ID
      ::=
STMTLIST ::= STATEMENTS
         ::=
STATEMENTS ::= STATEMENT
           ::= STATEMENTS  STATEMENT
STATEMENT ::= COMPOUND_STMT
          ::= WHILE_STMT
          ::= FOR_STMT
          ::= DO_STMT
          ::= SEARCH_STMT
          ::= FIND_STMT
```

```
            ::= PRINT_STMT
            ::= WINDOW_STMT
            ::= IF_STMT
            ::= SWITCH_STMT
            ::= EXPR_STMT
            ::= PAUSE_STMT
            ::= BP_CLAUSE
            ::= BREAK_STMT
            ::= CONTINUE_STMT
            ::= RETURN_CLAUSE
            ::= MOVE_REC_STMT
            ::= THROW_STMT
            ::= TRY_STMT
            ::= RETRY_STMT
            ::= TTS_STMT
            ::= FLUSH_STMT
            ::= TBLLOCK_STMT
            ::= CHANGE_STMT
            ::= UPDATE_STMT
            ::= INSERT_STMT
            ::= UNCHECKED_STMT
COMPOUND_STMT ::= LEFTBR_SYM  STMTLIST  RIGHTBR_SYM
THROW_STMT ::= THROW_SYM  IF_EXPR  SEMICOLON_SYM
TRY_STMT ::= TRY_BLOCK  CATCH_LIST
TRY_BLOCK ::= TRY_START  STATEMENT
TRY_START ::= TRY_SYM
CATCH_LIST ::= CATCH_STMT
            ::= CATCH_LIST  CATCH_STMT
CATCH_STMT ::= CATCH_EXPR  PRE_CATCH  STATEMENT  POST_CATCH
CATCH_EXPR ::= CATCH_SYM  LEFT_PAR_SYM  IF_EXPR  RGHT_PAR_SYM
  ::= CATCH_SYM  LEFT_PAR_SYM  IF_EXPR  LIST_SEP_SYM  TABLEINSTANCE  RGHT_PAR_SYM
  ::= CATCH_SYM
PRE_CATCH ::=
POST_CATCH ::=
TABLEINSTANCE ::= INSTANCENAME
INSTANCENAME ::= QUALIFIER  STD_ID  ARR_IDX
              ::= STD_ID  ARR_IDX
RETRY_STMT ::= RETRY_SYM  SEMICOLON_SYM
WHILE_STMT ::= WHILE_TEST  STATEMENT
WHILE_TEST ::= WHILE  LEFT_PAR_SYM  IF_EXPR  RGHT_PAR_SYM
WHILE ::= WHILE_SYM
DO_STMT ::= DO_BODY  DO_TEST  SEMICOLON_SYM
DO_BODY ::= DO_HEADER  STATEMENT
DO_HEADER ::= DO_SYM
DO_TEST ::= WHILE_SYM  LEFT_PAR_SYM  IF_EXPR  RGHT_PAR_SYM
FOR_STMT ::= FOR_HEADER  STATEMENT
FOR_HEADER ::= FOR_TEST  SEMICOLON_SYM  FOR_ASG  RGHT_PAR_SYM
FOR_TEST ::= FOR_INIT  SEMICOLON_SYM  IF_EXPR
FOR_INIT ::= FOR_SYM  LEFT_PAR_SYM  FOR_ASG
FOR_ASG ::= LVAL_FLD  ASSIGN  IF_EXPR
         ::= LVAL_FLD  ASG_INC_DEC
         ::= ASG_INC_DEC  LVAL_FLD
JOIN_LIST ::= JOIN_SPECS
           ::=
JOIN_SPECS ::= JOIN_SPEC
            ::= JOIN_SPECS  JOIN_SPEC
JOIN_SPEC ::= JOIN_ORDER  WHERE  IF_EXPR
           ::= JOIN_ORDER
JOIN_ORDER ::= JOIN_USING
            ::= JOIN_USING  ORDER_GROUP
JOIN_USING ::= JOIN_CLAUSE  USING_INDEX  STD_ID
            ::= JOIN_CLAUSE  USING_INDEX  HINT_SYM  STD_ID
            ::= JOIN_CLAUSE
JOIN_CLAUSE ::= OUTER  JOIN_SYM  SELECTOPT  TABLE
OUTER ::= OUTER_SYM
       ::= EXISTS_SYM
       ::= NOTEXISTS_SYM
       ::=
SEARCH_STMT ::= SEARCH_JOIN  STATEMENT
```

```
SEARCH_JOIN ::= SEARCH_WHERE  JOIN_LIST
SEARCH_WHERE ::= SEARCH_ORDER  WHERE  IF_EXPR
             ::= SEARCH_ORDER
WHERE ::= WHERE_SYM
SUM_ELEM ::= SUM_FUNC  LEFT_PAR_SYM  STD_ID  RGHT_PAR_SYM
SUM_FUNC ::= SUM_SYM
         ::= AVG_SYM
         ::= CNT_SYM
         ::= MINOF_SYM
         ::= MAXOF_SYM
SEARCH_ORDER ::= SEARCH_USING
             ::= SEARCH_USING  ORDER_GROUP
ORDER_GROUP ::= ORDERBY_CLAUSE  OPT_GROUPBY
            ::= GROUPBY_CLAUSE  OPT_ORDERBY
OPT_GROUPBY ::= GROUPBY_CLAUSE
            ::=
OPT_ORDERBY ::= ORDERBY_CLAUSE
            ::=
ORDERBY_CLAUSE ::= ORDER_SYM  OPT_BY  ORDER_ELEM
               ::= ORDERBY_CLAUSE  LIST_SEP_SYM  ORDER_ELEM
GROUPBY_CLAUSE ::= GROUP_SYM  OPT_BY  ORDER_ELEM
               ::= GROUPBY_CLAUSE  LIST_SEP_SYM  ORDER_ELEM
ORDER_ELEM ::= STD_ID  INDEX  DIRECTION
           ::= ORDER_QUALIFIER  STD_ID  INDEX  DIRECTION
ORDER_QUALIFIER ::= STD_ID  PERIOD_SYM
INDEX ::= LEFT_BRKT_SYM  INT_SYM  RGHT_BRKT_SYM
      ::=
DIRECTION ::= ASCEND_SYM
          ::= DESCEND_SYM
          ::=
OPT_BY ::= BY_SYM
       ::=
SEARCH_USING ::= SEARCH_CLAUSE  USING_INDEX  STD_ID
             ::= SEARCH_CLAUSE  USING_INDEX  HINT_SYM  STD_ID
             ::= SEARCH_CLAUSE
USING_INDEX ::= INDEX_SYM
SEARCH_CLAUSE ::= WHILE_SYM  SELECT_SYM  SELECTOPT  CROSSCOMPANY_CLAUSE  VALIDTIMESTATE_CLAUSE  TABLE
CROSSCOMPANY_CLAUSE ::= CROSSCOMPANY_SYM
                    ::= CROSSCOMPANY_SYM  COLON_SYM  STD_ID
                    ::=
VALIDTIMESTATE_CLAUSE ::= VALIDTIMESTATE_SYM  LEFT_PAR_SYM  STD_ID  LIST_SEP_SYM  STD_ID  RGHT_PAR_SYM
  ::= VALIDTIMESTATE_SYM  LEFT_PAR_SYM  STD_ID  RGHT_PAR_SYM
  ::=
SELECTOPT ::=
          ::= SELECTOPT  REVERSE_SYM
          ::= SELECTOPT  FIRSTFAST_SYM
          ::= SELECTOPT  FIRSTONLY_SYM
          ::= SELECTOPT  FIRSTONLY_SYM1
          ::= SELECTOPT  FIRSTONLY_SYM10
          ::= SELECTOPT  FIRSTONLY_SYM100
          ::= SELECTOPT  FIRSTONLY_SYM1000
          ::= SELECTOPT  FORUPDATE_SYM
          ::= SELECTOPT  NOFETCH_SYM
          ::= SELECTOPT  FORCE_SELECT_ORDER_SYM
          ::= SELECTOPT  FORCE_NESTED_LOOP_SYM
          ::= SELECTOPT  FORCE_LITERALS_SYM
          ::= SELECTOPT  FORCE_PLACEHOLDERS_SYM
          ::= SELECTOPT  REPEATABLEREAD_SYM
          ::= SELECTOPT  OPTIMISTICLOCK_SYM
          ::= SELECTOPT  PESSIMISTICLOCK_SYM
          ::= SELECTOPT  GENERATEONLY_SYM
FIND_STMT ::= FIND_JOIN  SEMICOLON_SYM
FIND_JOIN ::= FIND_WHERE  JOIN_LIST
FIND_WHERE ::= FIND_ORDER  WHERE  IF_EXPR
           ::= FIND_ORDER
FIND_ORDER ::= FIND_USING
           ::= FIND_USING  ORDER_GROUP
FIND_USING ::= FIND_TABLE  USING_INDEX  STD_ID
           ::= FIND_TABLE  USING_INDEX  HINT_SYM  STD_ID
```

```
               ::= FIND_TABLE
    FIND_TABLE ::= SELECT_SYM  SELECTOPT  CROSSCOMPANY_CLAUSE  VALIDTIMESTATE_CLAUSE  TABLE
      ::= DELETE_SYM  SELECTOPT  CROSSCOMPANY_CLAUSE  VALIDTIMESTATE_CLAUSE  TABLE
    TABLE ::= FLD_LIST  OPT_FROM
    FLD_LIST ::= MULT_SYM
             ::= FIELD_LIST
    FIELD_LIST ::= FIELD_SPEC
               ::= FIELD_LIST  LIST_SEP_SYM  FIELD_SPEC
    FIELD_SPEC ::= STD_ID  INDEX
               ::= SUM_ELEM
    OPT_FROM ::= FROM_SYM  STD_ID
             ::=
    SETFIELDSMODE ::=
    UPDATE_STMT ::= UPDATETABLE  SET_SYM  SETFIELDSMODE  FIELDASSIGNMENTS  OPT_WHERE  JOIN_LIST
SEMICOLON_SYM
    UPDATETABLE ::= UPDATE_SYM  SELECTOPT  CROSSCOMPANY_CLAUSE  STD_ID
    OPT_WHERE ::= WHERE  IF_EXPR
              ::=
    FIELDASSIGNMENTS ::= FIELDASSIGNMENTS  LIST_SEP_SYM  FIELDASSIGNMENT
                     ::= FIELDASSIGNMENT
    FIELDASSIGNMENT ::= STD_ID  INDEX  ASG_SYM  IF_EXPR
    INSERT_PART ::= INSERT_SYM  CROSSCOMPANY_CLAUSE  INSERT_NAME  LEFT_PAR_SYM  INSERTFIELDLIST
RGHT_PAR_SYM
    INSERT_NAME ::= STD_ID
    INSERT_STMT ::= INSERT_PART  FIND_JOIN  SEMICOLON_SYM
    INSERTFIELDLIST ::= INSERTFIELD
                    ::= INSERTFIELDLIST  LIST_SEP_SYM  INSERTFIELD
    INSERTFIELD ::= STD_ID  INDEX
    PRINT_STMT ::= PRINT_CLAUSE  AT_CLAUSE  SEMICOLON_SYM
    PRINT_CLAUSE ::= PRINT  IF_EXPR  EXPR_LIST
    PRINT ::= PRINT_SYM
    AT_CLAUSE ::= AT_SYM  IF_EXPR  LIST_SEP_SYM  IF_EXPR
              ::=
    WINDOW_STMT ::= WINDOW_SYM  IF_EXPR  LIST_SEP_SYM  IF_EXPR  AT_CLAUSE  SEMICOLON_SYM
    IF_STMT ::= ELSE_STMT
            ::= IF_CONDS
    IF_CONDS ::= IF_COND  STATEMENT
    IF_COND ::= IF_SYM  LEFT_PAR_SYM  IF_EXPR  RGHT_PAR_SYM
    ELSE_STMT ::= ELSE  STATEMENT
    ELSE ::= IF_CONDS  ELSE_SYM
    SWITCH_STMT ::= CASE_LIST  RIGHTBR_SYM
    CASE_LIST ::= SWITCH_SYM  LEFT_PAR_SYM  IF_EXPR  RGHT_PAR_SYM  LEFTBR_SYM
              ::= CASE_TESTS  STMTLIST
    CASE_TESTS ::= CASE_HEADER  COLON_SYM
               ::= CASE_LIST  DEFAULT_SYM  COLON_SYM
    CASE_HEADER ::= CASE  IF_EXPR
                ::= CASEALT  IF_EXPR
    CASE ::= CASE_LIST  CASE_SYM
    CASEALT ::= CASE_HEADER  LIST_SEP_SYM
    EXPR_STMT ::= ASG_STMT  SEMICOLON_SYM
              ::= FUNCTION  SEMICOLON_SYM
              ::= INTRINSICS  SEMICOLON_SYM
              ::= EVAL  SEMICOLON_SYM
    PAUSE_STMT ::= PAUSE_SYM  SEMICOLON_SYM
    BP_CLAUSE ::= BP_SYM  SEMICOLON_SYM
    BREAK_STMT ::= BREAK_SYM  SEMICOLON_SYM
    CONTINUE_STMT ::= CONTINUE_SYM  SEMICOLON_SYM
    RETURN_CLAUSE ::= RETURN_SYM  SEMICOLON_SYM
                  ::= RETURN_SYM  IF_EXPR  SEMICOLON_SYM
    TTS_STMT ::= TTSABORT_SYM  SEMICOLON_SYM
             ::= TTSBEGIN_SYM  SEMICOLON_SYM
             ::= TTSEND_SYM  SEMICOLON_SYM
    FLUSH_STMT ::= FLUSH  ID_LIST  SEMICOLON_SYM
    FLUSH ::= FLUSH_SYM
    TBLLOCK_STMT ::= TABLELOCK  ID_LIST  SEMICOLON_SYM
    TABLELOCK ::= TABLELOCK_SYM
    ID_LIST ::= STD_ID
            ::= ID_LIST  LIST_SEP_SYM  STD_ID
    MOVE_REC_STMT ::= NEXT_SYM  TABLE  SEMICOLON_SYM
```

```
MOVE_REC_STMT ::= NEXT_SYM  TABLE  SEMICOLON_SYM
CHANGE_STMT ::= CHANGE_HEADER  STATEMENT
CHANGE_HEADER ::= CHANGE  LEFT_PAR_SYM  IF_EXPR  RGHT_PAR_SYM
CHANGE ::= CHANGECOMP_SYM
       ::= CHANGESITE_SYM
UNCHECKED_STMT ::= UNCHECKED_HEADER  STATEMENT
UNCHECKED_HEADER ::= UNCHECKED_SYM  LEFT_PAR_SYM  IF_EXPR  RGHT_PAR_SYM
```

X++ Language Syntax is Stricter in Microsoft Dynamics AX 2012

Starting in Microsoft Dynamics AX 2012, the syntax rules for X++ are stricter than in previous versions of the product. This topic describes the syntax changes.

Table of X++ Syntax Changes

The following table displays a list of syntax changes that start in Microsoft Dynamics AX 2012.

AREA	SYNTAX RULE	BEFORE MICROSOFT DYNAMICS AX 2012	STARTING WITH MICROSOFT DYNAMICS AX 2012
Escape	The backslash character is rejected by the compiler for unrecognized escapes	The compiler used to accept "31\12\2002", but during run time the literal string was interpreted as a different value.	Now the following X++ statement is rejected by the compiler: str myDateString = "31\12\2002"; The proper syntax is "31\12\2002".
Exceptions	Retry is no longer allowed outside of a catch block	It was possible to write the **retry** keyword outside of a **catch** block. This caused the program to end when the **retry** was reached during runtime.	Now **retry** can occur only inside a **catch** block. For more information, see Exception Handling with try and catch Keywords.
Exceptions	Now you can throw and catch only `int` values	It was possible to throw scalar expressions like strings and dates, such as `throw "hello world";`, and get no compile error. At runtime this was catch-able by a `catch` block that was not decorated with any specific value, such as catch {print("Catch worked.");}.	Now the only expression you can put on the **throw** keyword is an `int`. Often the best thing to throw is Global::error("Explanation");. Often the best thing to catch is an element of the `Exception` enum. For more information, see Exception Handling with try and catch Keywords.

AREA	SYNTAX RULE	BEFORE MICROSOFT DYNAMICS AX 2012	STARTING WITH MICROSOFT DYNAMICS AX 2012

Inheritance	Downcasting can now be explicit.
	NOTE
	It is good programming practice to avoid implicit downcasts.
		It was possible to assign a base object to a derived object with the simple assignment operator, which is the equals sign (`=`). The compiler accepted these assignments, but during run time any misuse of an improper downcast assignment caused an error.	Now all downcasts can be explicit. This is accomplished with the new **as** expression operator. Explicit downcasting with the **as** keyword is illustrated by the following code example, in which `ThingClass` extends `Object` :
			`ThingClass myThing = new ThingClass();`
			`Object myObject = myThing;`
			`myThing = myObject as ThingClass; // Explicit downcast, good.`
			For more information, see Expression Operators: Is and As for Inheritance.

Inheritance	Override of a base method cannot be less accessible than the base method	It was possible to have a base method be decorated with **protected** and yet have an override of that method be **private**.	Now when a base method is **protected**, the override method must be either **protected** or **public**, and the override method cannot be **private**. For more information, see Method Access Control.
Inheritance	Override of a base method must have the exact same return type and parameter signature as the base method	Suppose a base class had a method that inputs a parameter of the `Common` table, which is the base of all tables. In a derived class it was possible to override the method to instead input `MyTable` .	Now the parameter signatures of the base method and its override method must match exactly. Also, the return types must match exactly. For more information, see Overriding a Method.
Interfaces	Implementation of an interface method must match the parameter signature exactly	Suppose an interface had a method that input a parameter of an `int` . In a class that implements the interface, it was possible to write the method with a parameter of a `str` .	Now the parameter signatures of the method must exactly match between the interface and the implementation of the method on a class. Also, the return types must match exactly. For more information, see Interfaces Overview.

AREA	SYNTAX RULE	BEFORE MICROSOFT DYNAMICS AX 2012	STARTING WITH MICROSOFT DYNAMICS AX 2012
Interfaces	A non-abstract base class that implements an interface cannot rely on a derived class for that implementation	When a base class implements an interface, it was possible for the class to not implement the methods of the interface if a derived class implemented the methods. The only limitation was that the `new` constructor method could not be called on the class.	Now the compiler requires that every class that implements an interface must have or inherit a complete implementation of every method of the interface. For more information, see X++, C# Comparison: Object Oriented Programming.
Modifiers	The **static** modifier should not be applied to an interface	It was possible to write static interface IMyInterface {}, but the **static** modifier had no effect because it makes no sense in this context.	Sometime after Dynamics AX 2009 the X++ compiler might stop allowing the **static** modifier on interface declarations. For more information, see Interfaces Overview.
Modifiers	The **static** modifier must not be applied to the `new` constructor	It was possible to apply the **static** modifier to the declaration of the `new` constructor method. This caused `new MyClass();` to behave as a null operation. Instead, the statement MyClass::new(); would call the static `new` method, but that would not construct an object.	Now the compiler issues an error when the **static** modifier is applied to the `new` method. For more information, see Constructors.
Modifiers	Use an explicit access modifier on each method	In the past the menu item of AOT > Classes > *MyClass* > New Method created the method without any access modifier. This meant that the method was implicitly **public**, although some X++ developers might not have been fully aware of the default. This created extra work later when a developer needed to modify the code in the method, because the developer had to research everywhere that the method might be called from.	Now the New Method menu item explicitly includes the **private** keyword in its automatic declaration of the new method. The developer can type in a different modifier if appropriate. For more information, see Method Modifiers.

AREA	SYNTAX RULE	BEFORE MICROSOFT DYNAMICS AX 2012	STARTING WITH MICROSOFT DYNAMICS AX 2012
Parameters	Parameters given in a call to a `new` constructor method must match the parameters on the `new` constructor method	It was possible to pass in multiple parameters on call to a `new` constructor method even when the `new` method was declared to input no parameters.	Now the call to the `new` method must exactly match the declared parameter signature of the `new` method. For more information, see Creating a Subclass.
Parameters	Parameters with default values must come after all parameters that do not have default values	It was possible to declare a method that takes in two parameters, and have only the first parameter offer a default value. There was no purpose to this. There was no way to accept the default of the first parameter because the call must specify a value for the second parameter and cannot omit the first parameter.	Now in the declaration of a method, any parameter that offers a default value must come after all the parameters that do not. For more information, see the following topics: • Using Optional Parameters • Best Practices for Parameters
Parameters	Override of a method must have the same default parameters as the overridden method	It was possible to declare a method as public void myMethod(int i=22){} and the override as public void myMethod(){}. But if the override method was called as `derivedObject(333);` an error occurred.	Now the override method must list the same parameter types in the same sequence that they are declared in the overridden method. For more information, see Overriding a Method.
Preprocessor	A **TODO** in a comment must be the first non-whitespace in the first line of the comment	The X++ preprocessor used to detect the **TODO** keyword in a multi-line /* ... */ task comment even when the **TODO** appeared after other text after the first comment line.	Now the X++ preprocessor detects the **TODO** keyword only if **TODO** appears on the first line of the comment, and as the first non-whitespace in the comment. For more information, see TODO Comments for X++ Developer Tasks.

Additional resources

X++ Language Reference

API, class, and table resources

2/18/2021 • 2 minutes to read • Edit Online

This topic describes where to find API documentation in Visual Studio and on the Microsoft docs site.

Application classes and tables

Application class and table documentation is in Visual Studio

You can find documentation for the Application classes in Microsoft Visual Studio. Search for the class name in Application Explorer and then display the code. You can find additional metadata about the class in the **Properties** window. You can download a list of all the tables in the Technical Reference Reports. For more information, see Find information about standard data entities.

Programming with application tables and classes

Application tables are similar to application classes, but with the following differences from classes:

- Tables are persistent.
- Table fields are always public.
- A table almost always corresponds to a real object.
- The definition of a table must sometimes be erased if you later want another table to extend it.

Design pattern of private new in application classes

All application classes are under Application Explorer > Classes. Every application class has the constructor method named `new`, even if the class has no **new** node in the Application Explorer. If the class has no explicit **new** node, the implicit **new** method is public. A design pattern that is sometimes used in the application classes is to declare the explicit **new** constructor method as **private**. Then a **public static** method is added to call the **new** method. The static method can restrict or control the call the **new** method based on various conditions, if necessary.

System classes and tables

System API, class, and table documentation is on the Microsoft docs site

Documentation for the classes and functions that are listed under **System Documentation** in Application Explorer is available on the Microsoft docs site.

X++ compile-time functions

X++ compile-time functions

X++ run-time functions

X++ run-time functions:

- X++ container run-time functions
- X++ business run-time functions
- X++ conversion run-time functions
- X++ date run-time functions
- X++ math run-time functions
- X++ reflection run-time functions

- X++ session run-time functions
- X++ string run-time functions

System tables

System tables

System classes

The reference documentation for the system classes is in the .NET API browser.

Microsoft.Dynamics.Ax.Xpp namespace

Dynamics.AX.Application namespace

X++ compile-time functions

2/18/2021 • 32 minutes to read • Edit Online

This topic lists the compile-time functions and describes their syntax, parameters, and return values.

Overview

Compile-time functions are executed early during compilation of X++ code. They should be used wherever possible in X++ code to make the code resilient to changes to the metadata stored in the Application Explorer. Compile-time functions have their input value verified by the compiler. If the input value is not found to match any existing object in the Application Explorer, the compiler issues an error. The inputs to these functions must be literals, because the compiler cannot determine the value that a variable contains at run time. A compile-time function is a metadata assertion function. It takes arguments that represents an entity in the Application Explorer and validates the arguments at compile time. It has no effect at run time. Attributes are classes that inherit from the **SysAttribute** class. To support the validation of form, report, query, and menu metadata, use the **AutoDeclaration** property on controls. Most of these functions retrieve metadata about items that are in the Application Explorer. Some common compile time functions are as follows:

- `classNum` – Retrieves the ID of a class.
- `classStr` – During compile time, verifies that a class of that name exists. This approach is better than discovering the error later during run time.
- `evalBuf` – Evaluates the input string of X++ code, and then returns the results as a string.
- `literalStr` – retrieves a label ID when given the string representation of a label, such as the string `"@SYS12345"`. For example, `myLabel.exists(literalStr("@SYS12345"));`.

> **NOTE**
> X++ compile time functions cannot be called from a .NET program.

Functions

attributeStr

Validates that the specified attribute class exists in the Application Explorer; if not, a compiler error occurs.

Syntax

```
str classStr(class class)
```

Parameters

PARAMETER	DESCRIPTION
class	The name of the attribute to validate.

Return Value

The name of the attribute.

Remarks

This is a compile-time function. For more information, see Overview.

Example

```
static void attributeStrExample(Args _args)
{
    str s;
    ;
    s = attributeStr(AifDocumentOperationAttribute);
    print s;
    pause;
}
```

classNum

Retrieves the ID of the specified class.

Syntax

```
int classNum(class class)
```

Parameters

PARAMETER	DESCRIPTION
class	The class for which to retrieve the ID.

Return Value

The ID of the specified class.

Remarks

This is a compile-time function. For more information, see Overview.

Example

```
static void classNumExample(Args _args)
{
    int i;
    ;
    i = classNum(Global);
    print i;
    pause;
}
```

classStr

Retrieves the name of a class as a string.

Syntax

```
str classStr(class class)
```

Parameters

PARAMETER	DESCRIPTION
class	The name of the class to return.

Return Value

The name of the class.

Remarks

Use this function instead of literal text to retrieve a string that contains the class name. If the class does not exist, the function generates a syntax error at compile time. This is a compile-time function. For more information, see Overview.

Example

```
static void clStrExample(Args _args)
{
    str s;
    ;
    s = classStr(Global);
    print s;
    pause;
}
```

configurationKeyNum

Retrieves the ID of the specified configuration key.

Syntax

```
int configurationKeyNum(str keyname)
```

Parameters

PARAMETER	DESCRIPTION
keyname	The configuration key for which to return the ID.

Return Value

The ID of the specified configuration key.

Remarks

This is a compile-time function. For more information, see Overview.

Example

```
static void configurationKeyNum(Args _args)
{
    int i;
    ;
    i = configurationKeyNum(AIF);
    print i;
    pause;
}
```

configurationKeyStr

Retrieves the name of a configuration key as a string.

Syntax

```
str configurationKeyStr(str keyname)
```

Parameters

PARAMETER	DESCRIPTION
keyname	The name of the configuration key.

Return Value

The name of the configuration key.

Remarks

Use this function instead of literal text to retrieve a string that contains the configuration key name. If the key does not exist, the function generates a syntax error at compile time. This is a compile-time function. For more information, see Overview.

Example

```
static void configurationKeyStrExample(Args _args)
{
    str s;
    ;
    s = configurationKeyStr(AIF);
    print s;
    pause;
}
```

dataEntityDataSourceStr

Retrieves the name of a data source of a data entity.

Syntax

```
str dataEntityDataSourceStr(str dataEntity, str dataSource)
```

Parameters

PARAMETER	DESCRIPTION
dataEntity	The name of the data entity.
dataSource	The name of the data source.

Return Value

The name of the data source.

Remarks

This is a compile-time function. For more information, see Overview.

Example

No example.

delegateStr

Returns the name of the delegate.

Syntax

```
str delegateStr(str class, str instanceDelegate)
```

Parameters

PARAMETER	DESCRIPTION
class	The name of the class, table, or form.
instanceDelegate	The name of the instance delegate.

Return Value

The name of the delegate.

Remarks

This is a compile-time function. For more information, see Overview.

Example

No example.

dimensionHierarchyLevelStr

Returns the name of the dimension hierarchy level.

Syntax

```
str dimensionHierarchyLevelStr(str dimensionHierarchyLevel)
```

Parameters

PARAMETER	DESCRIPTION
dimensionHierarchyLevel	The name of the dimension hierarchy level.

Return Value

The name of the dimension hierarchy level.

Remarks

This is a compile-time function. For more information, see Overview.

Example

No example.

dimensionHierarchyStr

Returns the name of the dimension hierarchy.

Syntax

```
str dimensionHierarchyStr(str dimensionHierarchy)
```

Parameters

PARAMETER	DESCRIPTION
dimensionHierarchy	The name of the dimension hierarchy.

Return Value

The name of the dimension hierarchy.

Remarks

This is a compile-time function. For more information, see Overview.

Example

No example.

dimensionReferenceStr

Returns the name of the dimension reference.

Syntax

```
str dimensionReferenceStr(str dimensionReference)
```

Parameters

PARAMETER	DESCRIPTION
dimensionReference	The name of the dimension reference.

Return Value

The name of the dimension reference.

Remarks

This is a compile-time function. For more information, see Overview.

Example

No example.

dutyStr

Retrieves a string that represents the name of the specified security duty.

Syntax

```
str dutyStr(str securityDuty)
```

Parameters

PARAMETER	DESCRIPTION
securityDuty	The name of the security duty.

Return Value

The name of the security duty in a string.

Remarks

This is a compile-time function. For more information, see Overview.

Example

No example.

enumCnt

Retrieves the number of elements in the specified enumeration type.

Syntax

```
int enumCnt(enum enumtype)
```

Parameters

PARAMETER	DESCRIPTION
enumtype	The enumeration type.

Return Value

The number of elements in the specified enumeration type.

Remarks

This is a compile-time function. For more information, see Overview.

Example

```
enumCnt(NoYes); //Returns 2, as the two elements are Yes and No.
```

enumLiteralStr

Indicates whether the specified string is an element of the specified enumeration type.

Syntax

```
\enumLiteralStr(enum enum, string str)
```

Parameters

PARAMETER	DESCRIPTION
enum	The enumeration type from which to retrieve the specified value.

Return Value

The value of the *str* parameter if the specified string was found; otherwise, a compilation error.

Remarks

This is a compile-time function. For more information, see Overview.

Example

```
static void getEnumValueAsString()
{
    str i;
    i = enumLiteralStr(ABCEnum, "valueInABCEnum");
    print i;
    pause;
}
```

enumNum

Retrieves the ID of the specified enumeration type.

Syntax

```
int enumNum(enum enum)
```

Parameters

PARAMETER	DESCRIPTION
enum	The enumeration for which to return the ID.

Return Value

The ID of the specified enumeration type.

Remarks

This is a compile-time function. For more information, see Overview.

Example

```
static void enumNum(Args _args)
{
    int i;
    ;
    i = enumNum(ABC);
    print i;
    pause;
}
```

enumStr

Retrieves the name of an enumeration as a string.

Syntax

```
str enumStr(enum enum)
```

Parameters

PARAMETER	DESCRIPTION
enum	The name of the enumeration.

Return Value

The name of the enumeration.

Remarks

This is a compile-time function. For more information, see Overview.

Example

```
static void enumStrExample(Args _args)
{
    str s;
    ;
    s = enumStr(ABC);
    print s;
    pause;
}
```

extendedTypeNum

Retrieves the ID of the specified extended data type.

Syntax

```
int extendedTypeNum(int str)
```

Parameters

PARAMETER	DESCRIPTION
str	The extended data type for which to return the ID.

Return Value

The ID of the specified extended data type.

Remarks

This is a compile-time function. For more information, see Overview.

Example

```
static void EDTNum(Args _args)
{
    int i;
    str s;
    ;

    i = extendedTypeNum(AccountName);
    s = extendedTypeStr(AccountName);
    print  int2Str(i);
    print  s;
    pause;
}
```

extendedTypeStr

Retrieves the name of an extended data type as a string.

Syntax

```
str extendedTypeStr(int str)
```

Parameters

PARAMETER	DESCRIPTION
str	The name of the extended data type.

Return Value

The name of the extended data type.

Remarks

Use this function instead of literal text to return a string that contains the extended data type name. If the data type does not exist, the **extendedTypeStr** function generates a syntax error at compile time. This is a compile-time function. For more information, see Overview.

Example

```
static void EDTStr(Args _args)
{
    int i;
    str s;
    ;

    i = extendedTypeNum(AccountName);
    s = extendedTypeStr(AccountName);
    print  int2Str(i);
    print  s;
    pause;
}
```

fieldNum

Returns the ID number of the specified field.

Syntax

```
int fieldNum(str tableName, str fieldName)
```

Parameters

PARAMETER	DESCRIPTION
tableName	The name of the table.
fieldName	The name of the field.

Return Value

The ID of the specified field.

Remarks

This is a compile-time function. For more information, see Overview.

Example

The following example prints the number of the **CashDisc** field in the **CustTable** table.

```
static void fieldNumExample(Args _args)
{
    int myInt;
    ;

    myInt = fieldNum(CustTable, CashDisc);
    Global::info(strfmt("CashDisc has a field ID of %1 in the CustTable table.", myInt));
}
/****Infolog Display
Message (10:40:00 am)
CashDisc has a field ID of 10 in the CustTable table.
****/
```

fieldPName

Retrieves the label of the specified field.

Syntax

```
str fieldPName(str tableid, str fieldid)
```

Parameters

PARAMETER	DESCRIPTION
tableid	The table that contains the specified field.
fieldid	The field to convert.

Return Value

The label of the field.

Remarks

This is a compile-time function. For more information, see Overview.

Example

The following example prints the label of the **CashDisc** field.

```
static void fieldPNameExample(Args _arg)
{
    str myText;
    ;

    myText = fieldPName(CustTable, CashDisc);
    Global::info(strfmt("%1 is the label of the CashDisc field.", myText));
}
/****Infolog Display
Message (02:00:57 pm)
Cash discount is the label of the CashDisc field.
****/
```

fieldStr

Retrieves the field name of the specified field.

Syntax

```
str fieldStr(str tableid, str fieldid)
```

Parameters

PARAMETER	DESCRIPTION
tableid	The table that contains the field.
fieldid	The field to convert.

Return Value

The field name of the specified field.

Remarks

This is a compile-time function. For more information, see Overview.

Example

The following example assigns the name of the **CashDisc** field to the *myText* variable.

```
static void fieldStrExample(Args _arg)
{
    str myText;
    ;

    myText = fieldStr(CustTable, CashDisc);
    Global::info(strfmt("%1 is the specified field.", myText));
}
/****Infolog Display
Message (09:11:52 am)
CashDisc is the specified field.
****/
```

formControlStr

Causes the X++ compiler to check whether the control exists on the form, and to replace the function call with a string of the valid control name.

Syntax

```
str formControlStr(formName, controlName)
```

Parameters

PARAMETER	DESCRIPTION
formName	The name of the form, not in quotation marks.
controlName	The name of the control that is on the form, not in quotation marks.

Return Value

A string that contains the name of the control as it appears in the Application Explorer.

Remarks

A compile error is issued if the compiler determines that the control does not exist on the form. If your X++ code uses a string that contains quotation marks to supply the control name, the error cannot be discovered until run time. Use of this function enables the compiler to discover the error earlier at compile time. X++ functions such as **formControlStr** that are executed by the compiler are called compile-time functions or compile-time functions. That is why the input parameters are not standard strings in quotation marks. Compile-time functions are not represented in the p-code or other executable that is output by the compiler. This is a compile-time function. For more information, see Overview.

Example

No example.

formDataFieldStr

Returns the name of a data field in a form.

Syntax

```
str formDataFieldStr(str formName, str dataSource, str dataField)
```

Parameters

PARAMETER	DESCRIPTION
formName	The name of the form.
dataSource	The data source of the form.
dataField	The data field of the data source.

Return Value

The name of a data field in a form.

Remarks

This is a compile-time function. For more information, see Overview.

Example

```
str a = formDataFieldStr(FMVehicle, FMModelRate, RatePerDay);
```

formDataSourceStr

Returns the name of a data source in a form.

Syntax

```
str formDataSourceStr(str formName, str dataSource)
```

Parameters

PARAMETER	DESCRIPTION
formName	The name of the form.

PARAMETER	DESCRIPTION
dataSource	The data source of the form.

Return Value

The name of a data source in a form.

Remarks

This is a compile-time function. For more information, see Overview.

Example

```
str b = formDataSourceStr(FMVehicle, FMModelRate);
```

formMethodStr

Returns the name of a method of a form.

Syntax

```
str formMethodStr(str formName, str methodName)
```

Parameters

PARAMETER	DESCRIPTION
formName	The name of the form.
methodName	The method of the form.

Return Value

The name of a method in a form.

Remarks

This is a compile-time function. For more information, see Overview.

Example

The following example prints the name of the **showDialog** method.

```
str c = formMethodStr(Batch,showDialog);
```

formStr

Retrieves the name of a form.

Syntax

```
str formStr(str form)
```

Parameters

PARAMETER	DESCRIPTION
form	The name of a form.

Return Value

A string that represents the name of the form.

Remarks

This is a compile-time function. For more information, see Overview.

Example

The following example prints the name of the InventDim form.

```
static void formStrExample(Args _arg)
{
    ;

    Global::info(formStr(InventDim));
}
/****Infolog Display
Message (11:04:39 am)
InventDim
****/
```

identifierStr

Converts the specified identifier to a string.

Syntax

```
str identifierStr(str ident)
```

Parameters

PARAMETER	DESCRIPTION
ident	The identifier to convert.

Return Value

A string that represents the specified identifier.

Remarks

You will receive a best practice warning if you use the **identifierStr** function. This occurs because existence checking is performed for **identifierStr**. Try to use a more specific compile-time function if one is available. This is a compile-time function. For more information, Overview.

Example

The following code example assigns the *myvar* variable name to the *thevar* variable.

```
static void indentifierStrExample(Args _args)
{
    str myvar;
    str thevar
    ;

    thevar = "[" + identifierStr(myvar) + "]";
    Global::info(strfmt(thevar));
}
/****Infolog Display
Message (09:19:49 am)
[myvar]
****/
```

indexNum

Converts the specified index to a number.

Syntax

```
int indexNum(str tableid, str indexid)
```

Parameters

PARAMETER	DESCRIPTION
tableid	The table that contains the index.
indexid	The index to convert.

Return Value

The index number that represents the specified index.

Remarks

This is a compile-time function. For more information, see Overview.

Example

The following example returns the index value of the Party index.

```
static void indexNumExample(Args _arg)
{
    ;

    Global::info(strfmt("%1 is the index number of Party.", indexNum(CustTable, Party)));
}
/****Infolog Display
Message (11:28:03 am)
3 is the index number of Party.
****/
```

indexStr

Converts the specified index to a string.

Syntax

```
str indexStr(str tableid, str indexid)
```

Parameters

PARAMETER	DESCRIPTION
tableid	The table that contains the index.
indexid	The index to convert.

Return Value

A string that represents the specified index.

Remarks

This is a compile-time function. For more information, see Overview.

Example

The following example assigns the **CashDisc** index value to the *myText* variable.

```
static void fieldStrExample(Args _arg)
{
    str myText;
    ;

    myText = fieldStr(CustTable, CashDisc);
    Global::info(strfmt("%1 is the specified index.", myText));
}
/****Infolog Display
Message (09:11:52 am)
CashDisc is the specified index.
****/
```

licenseCodeNum

Validates that the specified license code exists in the Application Explorer; if not, a compiler error occurs.

Syntax

```
int licenseCodeNum(str codename)
```

Parameters

PARAMETER	DESCRIPTION
codename	The name of the license code to validate.

Return Value

The number of the specified license code.

Remarks

This is a compile-time function. For more information, see Overview.

Example

```
static void licenseCodeNumExample(Args args)
{
    int i;
    ;

    i = licenseCodeNum(SysMorphX);
    Global::info(strfmt("%1 is the license code number for SysMorphX.", i));
}
/****Infolog Display
Message (01:52:35 pm)
24 is the license code number for SysMorphX.
****/
```

licenseCodeStr

Validates that the specified license code exists in the Application Explorer; if not, a compiler error occurs.

Syntax

```
str licenseCodeStr(str codename)
```

Parameters

PARAMETER	DESCRIPTION
codename	The name of the license code to validate.

Return Value

The name of the specified license code.

Remarks

This is a compile-time function. For more information, see Overview.

Example

```
static void licenseCodeStrExample(Args _arg)
{
    str s;
    ;

    s = licenseCodeStr(SysMorphX);
    Global::info(strfmt("%1 is the license code string for SysMorphX.", s));
}
/****Infolog Display
Message (02:33:56 pm)
SysMorphX is the license code string for SysMorphX.
****/
```

literalStr

Validates that the specified string can be a literal string; if not, a compiler error occurs.

Syntax

```
str literalStr(int str)
```

Parameters

PARAMETER	DESCRIPTION
codename	The string to validate.

Return Value

The literal string if valid.

Remarks

This is a compile-time function. For more information, see Overview.

Example

```
{
    str s;
    ;

    s = literalStr("This is a literal str");
    print s;
    pause;
}
```

maxDate

Retrieves the maximum value allowed for a variable of type date.

Syntax

```
date maxDate()
```

Return Value

The maximum value allowed for a variable of type **date**, which is **2154-12-31**.

Remarks

This is a compile-time function. For more information, see Overview.

Example

```
static void maxDateExample(Args _arg)
{
    date maximumDate;
    ;
    maximumDate = maxDate();
    print maximumDate;
    pause;
}
```

maxInt

Retrieves the maximum signed value that can be stored in an **int** type.

Syntax

```
int maxInt()
```

Return Value

The maximum value allowed value of an integer.

Remarks

Any other integer will be less than or equal to the returned value. This is a compile-time function. For more information, see Overview.

Example

```
static void maxIntExample(Args _arg)
{
    int i;
    ;
    print "The maximum value for type int is " + int2Str(maxInt());
    pause;
}
```

measurementStr

Returns the name of a measurement.

Syntax

```
str measurementStr(str measurement)
```

Parameters

PARAMETER	DESCRIPTION
measurement	The name of the measurement.

Return Value

The name of the measurement.

Remarks

This is a compile-time function. For more information, see Overview.

Example

No example.

measureStr

Returns the name of a measure.

Syntax

```
str measureStr(str measure)
```

Parameters

PARAMETER	DESCRIPTION
measure	The name of the measure.

Return Value

The name of the measure.

Remarks

This is a compile-time function. For more information, see Overview.

Example

No example.

menuItemActionStr

Validates that the specified menu item action exists in the Application Object Tree (Application Explorer); if it does not, a compiler error occurs.

Syntax

```
str menuItemActionStr(class menuitem)
```

Parameters

PARAMETER	DESCRIPTION
codename	The name of the menu item action to validate.

Return Value

The name of the menu item action, if it is valid.

Remarks

This is a compile-time function. For more information, see Overview.

Example

```
{
    str s1, s2, s3, s4;
    ;

    s1 = menuItemActionStr(AssetCopy);
    s2 = menuItemDisplayStr(Address);
    s3 = menuItemOutputStr(AssetBarcode);
    s4 = menuStr(Administration);

    print "menuItemActionStr for AssetCopy is " + s1;
    print "menuItemDisplayStr for Address is " + s2;
    print "menuItemOutputStr for AssetBarcode is " + s3;
    print "menuStr for Administration is " + s4;

    pause;
}
```

menuItemDisplayStr

Validates that the specified menu item display exists in the Application Explorer; if it does not, a compiler error occurs.

Syntax

```
str menuitemdisplaystr(class menuItem)
```

Parameters

PARAMETER	DESCRIPTION
codename	The name of the menu item display to validate.

Return Value

The name of the specified menu item display, if it is valid.

Remarks

This is a compile-time function. For more information, see Overview.

Example

```
{
    str s1, s2, s3, s4;
    ;

    s1 = menuItemActionStr(AssetCopy);
    s2 = menuItemDisplayStr(Address);
    s3 = menuItemOutputStr(AssetBarcode);
    s4 = menuStr(Administration);

    print "menuItemActionStr for AssetCopy is " + s1;
    print "menuItemDisplayStr for Address is " + s2;
    print "menuItemOutputStr for AssetBarcode is " + s3;
    print "menuStr for Administration is " + s4;

    pause;
}
```

menuItemOutputStr

Validates that the specified menu item output exists in the Application Explorer; if not, a compiler error occurs.

Syntax

```
str menuItemOutputStr(class menuitem)
```

Parameters

PARAMETER	DESCRIPTION
codename	The name of the menu item output to validate.

Return Value

The specified menu item output if valid.

Remarks

This is a compile-time function. For more information, see Overview.

Example

```
{
    str s1, s2, s3, s4;
    ;

    s1 = menuItemActionStr(AssetCopy);
    s2 = menuItemDisplayStr(Address);
    s3 = menuItemOutputStr(AssetBarcode);
    s4 = menuStr(Administration);

    print "menuItemActionStr for AssetCopy is " + s1;
    print "menuItemDisplayStr for Address is " + s2;
    print "menuItemOutputStr for AssetBarcode is " + s3;
    print "menuStr for Administration is " + s4;

    pause;
}
```

menuStr

Validates that the specified menu exists in the Application Explorer; if not, a compiler error occurs.

Syntax

```
str menuStr(class menu)
```

Parameters

PARAMETER	DESCRIPTION
menu	The name of the menu to validate.

Return Value

The name of the specified menu item if valid.

Remarks

This is a compile-time function. For more information, see Overview.

Example

```
{
    str s1, s2, s3, s4;
    ;

    s1 = menuItemActionStr(AssetCopy);
    s2 = menuItemDisplayStr(Address);
    s3 = menuItemOutputStr(AssetBarcode);
    s4 = menuStr(Administration);

    print "menuItemActionStr for AssetCopy is " + s1;
    print "menuItemDisplayStr for Address is " + s2;
    print "menuItemOutputStr for AssetBarcode is " + s3;
    print "menuStr for Administration is " + s4;

    pause;
}
```

methodStr

Validates that the specified method exists in the specified class; if it does not, a compiler error occurs.

Syntax

```
str methodStr(class class, int method)
```

Parameters

PARAMETER	DESCRIPTION
class	The name of the class.
method	The name of the method to validate.

Return Value

The name of the specified method, if it is valid.

Remarks

This is a compile-time function. For more information, see Overview.

Example

```
{
    #define.timeout(50)
    str s;
    SysHelpInitTimeOut SysHelpInitTimeOut;
    ;

    s = methodStr(SysHelpInitTimeOut, timeout);
    print s;
    pause;
}
```

minInt

Retrieves the minimum signed value that can be stored in an **int** type.

Syntax

```
int minInt()
```

Return Value

The minimum value of an **int** type.

Remarks

Any other integer value will be greater than or equal to the returned value. This is a compile-time function. For more information, see Overview.

Example

```
static void minIntExample(Args _arg)
{
    int i;
    ;
    i = minInt();
    print "minInt() is " + int2Str(i);
    pause;
}
```

privilegeStr

Returns the name of the privilege.

Syntax

```
str privilegeStr(str privilege)
```

Parameters

PARAMETER	DESCRIPTION
privilege	The privilege for which to return the name.

Return Value

The name of the privilege.

Remarks

This is a compile-time function. For more information, see Overview.

Example

No example.

queryDatasourceStr

Causes the X++ compiler to check whether the data source exists on the query, and to replace the function call with a string of the valid data source name.

Syntax

```
str queryDataSourceStr(queryName, dataSourceName)
```

Parameters

PARAMETER	DESCRIPTION
queryName	The name of the query, not in quotation marks.
dataSourceName	The name of the data source that is on the query, not in quotation marks.

Return Value

A string that contains the name of the data source as it appears in the Application Explorer.

Remarks

A compile error is issued if the compiler determines that the data source does not exist on the query. If your X++ code uses a string that contains quotation marks to supply the data source name, the error cannot be discovered until run time. Use of this function enables the compiler to discover the error earlier at compile time. X++ functions such as **queryDataSourceStr** that are executed by the compiler are referred to as compile-time functions or compile-time functions. That is why the input parameters are not standard strings in quotation marks. Compile-time functions are not represented in the p-code or other executable that is output by the compiler. This is a compile-time function. For more information, see Overview.

Example

No example.

queryMethodStr

Returns the name of a method of a query.

Syntax

```
str queryMethodStr(str queryName, str methodName)
```

Parameters

PARAMETER	DESCRIPTION
queryName	The name of the query.
methodName	The method of the form.

Return Value

The name of a method in a query.

Remarks

This is a compile-time function. For more information, see Overview.

Example

No example.

queryStr

Retrieves a string that represents an existing query.

Syntax

```
str queryStr(str query)
```

Parameters

PARAMETER	DESCRIPTION
query	The query to retrieve.

Return Value

The name of the query.

Remarks

This is a compile-time function. For more information, see Overview.

Example

```
static void queryStrExample(Args _arg)
{
    str myText;
    ;

    myText = queryStr(AssetTable);
    Global::info(strfmt("%1 is the name of the query.",myText));
}
/****Infolog Display
Message (09:45:16 am)
AssetTable is the name of the query.
****/
```

reportStr

Retrieves a string that represents the name of the specified report.

Syntax

```
str reportStr(str report)
```

Parameters

PARAMETER	DESCRIPTION
report	The report for which to return the name.

Return Value

The name of the report.

Remarks

This is a compile-time function. For more information, see Overview.

Example

The following example assigns the name of the **AssetAddition** report to the *MyTxt* variable.

```
static void reportStrExample(Args _args)
{
    str MyTxt;
    ;

    MyTxt = reportStr(AssetAddition);
    Global::info(strfmt("%1 is the name of the report.", MyTxt));
}
/****Infolog Display.
Message (10:46:36 am)
AssetAddition is the name of the report.
****/
```

resourceStr

Validates that the specified resource exists in the Application Explorer; if it does not, a compiler error occurs.

Syntax

```
str resourceStr(str resourcename)
```

Parameters

PARAMETER	DESCRIPTION
resourcename	The name of the resource to validate.

Return Value

The name of the specified resource, if it is valid.

Remarks

This is a compile-time function. For more information, see Overview.

Example

```
{
    print "Str for resource StyleSheet_Help_Axapta is "
        + resourceStr(StyleSheet_Help_Axapta);
    pause;
}
```

roleStr

Retrieves a string that represents the name of the specified security role.

Syntax

```
str roleStr(str securityRole)
```

Parameters

PARAMETER	DESCRIPTION
securityRole	The name of the security role.

Return Value

The name of the security role in a string.

Remarks

This is a compile-time function. For more information, see Overview.

Example

No example.

ssrsReportStr

Retrieves a string that represents the name of the specified report. Use this function when you want to specify the report that should be run by a report controller class.

Syntax

```
str ssrsReportStr(str report, str design)
```

Parameters

PARAMETER	DESCRIPTION
report	The report to return the name for.
design	The name of the design that is associated with the report.

Return Value

The name of the report.

Remarks

The **ssrsReportStr** function parses the two values that are passed to it, to validate whether they belong to a valid report. The report name must be set when a menu item points to a controller(), so that the controller can determine which report-design combination must be invoked. Use of the **ssrsReportStr** function provides the benefit of compile-time validation for the report and design name. This is a compile-time function. For more information, see Overview.

Example

```
public static void main(Args _args)
{
    // Initializing the object for a controller class, in this case, the class named AssetListingController.
    SrsReportRunController controller = new AssetListingController();

    // Getting the properties of the called object (in this case AssetListing MenuItem)
    controller.parmArgs(_args);
    // Setting the Report name for the controller.
    controller.parmReportName(ssrsReportStr(AssetListing, Report));

    // Initiate the report execution.
    controller.startOperation();
}
```

staticDelegateStr

Returns the name of a static delegate.

Syntax

```
str staticDelegateStr(str class, str delegate)
```

Parameters

PARAMETER	DESCRIPTION
class	The name of a class, table, or form.
delegate	The name of the delegate.

Return Value

The name of the delegate.

Remarks

This is a compile-time function. For more information, see Overview.

Example

No example.

staticMethodStr

Validates that the specified static method exists in the specified class; if it does not, a compiler error occurs.

Syntax

```
str staticMethodStr(class class, int method)
```

Parameters

PARAMETER	DESCRIPTION
class	The name of the class.
method	The name of the static method to validate.

Return Value

The name of the static method, if it is valid.

Remarks

This is a compile-time function. For more information, see Overview.

Example

No example.

tableCollectionStr

Validates that the specified table collection exists in the Application Explorer; if it does not, a compiler error occurs.

Syntax

```
str tableCollectionStr(class tablecollection)
```

Parameters

PARAMETER	DESCRIPTION
tablecollection	The name of the table collection to validate.

Return Value

The name of the specified table collection, if it is valid.

Remarks

This is a compile-time function. For more information, see Overview.

Example

No example.

tableFieldGroupStr

Retrieves the name of a field group as a string.

Syntax

```
str tableFieldGroupStr(str tableName, str fieldGroupName)
```

Parameters

PARAMETER	DESCRIPTION
tableName	The table that contains the field group.
fieldGroupName	The field group in the table.

Return Value

The name of the field group as a string.

Remarks

This is a compile-time function. For more information, see Overview.

Example

The following example retrieves the name of the **Editing** field group as a string.

```
static void tableFieldGroupStrExample(Args _arg)
{
    ;

    Global::info(tableFieldGroupStr(AccountingDistribution, Editing));
}
/****Infolog Display
Message (03:14:54 pm)
Editing
****/
```

tableMethodStr

Validates that the specified method exists in the specified table; if it does not, a compiler error occurs.

Syntax

```
str tableMethodStr(int table, int method)
```

Parameters

PARAMETER	DESCRIPTION
table	The name of the table.
method	The name of the method to validate.

Return Value

The name of the method, if it is valid.

Remarks

This is a compile-time function. For more information, see Overview.

Example

No example.

tableNum

Retrieves the table ID of the specified table.

Syntax

```
int tableNum(str table)
```

Parameters

PARAMETER	DESCRIPTION
table	The table to retrieve the table ID for.

Return Value

The table ID of the specified table.

Remarks

This is a compile-time function. For more information, see Overview.

Example

The following example sets the **tableID** variable to 77, which is the **ID** of the **CustTable** table.

```
static void tableNumExample(Args _args)
{
    int tableID;
    ;

    tableID = tableNum(CustTable);
    Global::info(strfmt("%1 is the table ID for the CustTable table.", tableID));

}
/****Infolog Display
Message (11:15:54 am)
77 is the table ID for the CustTable table.
****/
```

tablePName

Retrieves a string that contains the printable name of the specified table.

Syntax

```
str tablePName(str table)
```

Parameters

PARAMETER	DESCRIPTION
table	The table to retrieve the printable name for.

Return Value

The name of the specified table.

Remarks

This is a compile-time function. For more information, see Overview.

Example

The following example assigns the label of the **CustTable** table to the *MyText* variable.

```
static void tablePNameExample(Args _args)
{
    str MyText;
    ;

    MyText = tablePname(CustTable);
    Global::info(strfmt("%1 is the label of the CustTable table.", MyText));
}
/**** Infolog Display.
Message (12:13:53 pm)
Customers is the label of the CustTable table.
****/
```

tableStaticMethodStr

Validates that the specified static method exists in the specified table; if it does not, a compiler error occurs.

Syntax

```
str tableStaticMethodStr(int table, int method)
```

Parameters

PARAMETER	DESCRIPTION
table	The name of the table.
method	The name of the static method to validate.

Return Value

The name of the specified static method.

Remarks

This is a compile-time function. For more information, see Overview.

Example

No example.

tableStr

Retrieves a string that contains the name the specified table.

Syntax

```
str tableStr(str table)
```

Parameters

PARAMETER	DESCRIPTION
table	The table to retrieve a string for.

Return Value

A string value that contains the name of the specified table.

Remarks

This is a compile-time function. For more information, see Overview.

Example

The following example assigns the name of the **CustTable** table to the *MyTxt* variable.

```
static void tableStrExample(Args _args)
{
    str MyTxt;
    ;

    MyTxt = tableStr(CustTable);
    Global::info(strfmt("%1 is the str output of the input of CustTable.", MyTxt));
}
/**** Infolog Display.
Message (01:21:49 pm)
CustTable is the str output of the input of CustTable.
****/
```

tileStr

Retrieves a string that represents the name of the specified tile.

Syntax

```
str tileStr(str tile)
```

Parameters

PARAMETER	DESCRIPTION
tile	The name of the tile.

Return Value

The name of the tile in a string.

Remarks

This is a compile-time function. For more information, see Overview.

Example

No example.

varStr

Retrieves a string that contains the name of the specified variable.

Syntax

```
str varStr(str var)
```

Parameters

PARAMETER	DESCRIPTION
var	The name of the variable.

Return Value

A string that contains the name of the specified variable.

Remarks

This is a compile-time function. For more information, see Overview.

Example

```
static void varStrExample(Args _arg)
{
    str myString;
    anytype myVariable;
    ;

    myString = varStr(myVariable);
    Global::info(strfmt("%1 is the variable name.", myString));
}
/****Infolog Display.
Message (02:26:56 pm)
myVariable is the variable name.
****/
```

webActionItemStr

Validates that the specified web action item exists in the Application Explorer; if it does not, a compiler error occurs.

Syntax

```
str webActionItemStr(class webactionitem)
```

Parameters

PARAMETER	DESCRIPTION
webactionitem	The name of the web action item to validate.

Return Value

The name of the specified web action item, if it is valid.

Remarks

This is a compile-time function. For more information, see Overview.

Example

```
{
    str s;
    ;
    s = webActionItemStr(EPFlushData);
    print "webactionitem str is " + s;
    pause;
}
```

webDisplayContentItemStr

Validates that the specified web display content item exists in the Application Explorer; if it does not, a compiler error occurs.

Syntax

```
str webDisplayContentItemStr(class webdisplaycontentitem)
```

Parameters

PARAMETER	DESCRIPTION
webdisplaycontentitem	The name of the web display content item to validate.

Return Value

The name of the specified web display content item, if it is valid.

Remarks

This is a compile-time function. For more information, see Overview.

Example

```
{
    str s;
    ;

    s = webDisplayContentItemStr(EPAdmin);
    print "string for webcontent display item EPAdmin is " + s;
    pause;
}
```

webFormStr

Validates that the specified web form exists in the Application Explorer; if it does not, a compiler error occurs.

Syntax

```
str webFormStr(str name)
```

Parameters

PARAMETER	DESCRIPTION
name	The name of the web form to validate.

Return Value

The name of the specified web form, if it is valid.

Remarks

This is a compile-time function. For more information, see Overview.

Example

```
{
    str s;
    ;
    s = webFormStr(EPAdmin);
    print "String for web form EPAdmin is " + s;
    pause;
}
```

webletItemStr

Validates that the specified weblet item exists in the Application Explorer; if it does not, a compiler error occurs.

Syntax

```
str webletItemStr(class webletitem)
```

Parameters

PARAMETER	DESCRIPTION
webletitem	The name of the weblet item to validate.

Return Value

The name of the specified weblet item, if it is valid.

Remarks

This is a compile-time function. For more information, see Overview.

Example

```
{
    str s;
    ;
    s = webletItemStr(WebFormWeblet);
    print "String for WebFormWeblet is " + s;
    pause;
}
```

webMenuStr

Validates that the specified web menu exists in the Application Explorer; if it does not, a compiler error occurs.

Syntax

```
str webMenuStr(str name)
```

Parameters

PARAMETER	DESCRIPTION
name	The name of the web menu to validate.

Return Value

The name of the specified web menu, if it is valid.

Remarks

This is a compile-time function. For more information, see Overview.

Example

```
{
    str s;
    ;
    s = webMenuStr(ECPAdmin);
    print "String for web menu ECPAdmin is " + s;
    pause;
}
```

webOutputContentItemStr

Validates that the specified web output content item exists in the Application Explorer; if it does not, a compiler error occurs.

Syntax

```
str webOutputContentItemStr(class weboutputcontentitem)
```

Parameters

PARAMETER	DESCRIPTION
weboutputcontentitem	The name of the web output content item to validate.

Return Value

The name of the specified web output content item, if it is valid.

Remarks

This is a compile-time function. For more information, see Overview.

Example

```
{
    str s;
    ;
    s = webOutputContentItemStr(EPPriceList);
    print "string for weboutput content item EPPriceList is " + s;
    pause;
}
```

webpageDefStr

Validates that the specified Web page definition exists in the Application Explorer; if it does not, a compiler error occurs.

Syntax

```
str webpageDefStr(str pagename)
```

Parameters

PARAMETER	DESCRIPTION
pagename	The name of the Web page definition to validate.

Return Value

The name of the specified web-page definition, if it is valid.

Remarks

This is a compile-time function. For more information, see Overview.

Example

No example.

webReportStr

Validates that the specified web report exists in the Application Explorer; if it does not, a compiler error occurs.

Syntax

```
str webReportStr(str name)
```

Parameters

PARAMETER	DESCRIPTION
name	The name of the web report to validate.

Return Value

The name of the specified web report, if it is valid.

Remarks

This is a compile-time function. For more information, see Overview.

Example

```
{
    str s;
    ;
    s = webReportStr(EPCSSSalesConfirm);
    print "String for web report EPCSSalesConfirm is " + s;
    pause;
}
```

websiteDefStr

Validates that the specified web-site definition exists in the Application Explorer; if it does not, a compiler error occurs.

Syntax

```
str websiteDefStr(str resourcename)
```

Parameters

PARAMETER	DESCRIPTION
resourcename	The name of the Web site definition to validate.

Return Value

The name of the specified web-site definition, if it is valid.

Remarks

This is a compile-time function. For more information, see Overview.

Example

```
{
    str s;
    ;

    s = websiteDefStr(AxSiteDef_1033_xip);
    print "string for web site definition AxSiteDef_1033_xip is " + s;
    pause;
}
```

webSiteTempStr

Validates that the specified web-site template exists in the Application Explorer; if it does not, a compiler error occurs.

Syntax

```
str websiteTempStr(str resourcename)
```

Parameters

PARAMETER	DESCRIPTION
resourcename	The name of the Web site template to validate.

Return Value

The name of the specified web-site template, if it is valid.

Remarks

This is a compile-time function. For more information, see Overview.

Example

No example.

webStaticFileStr

Validates that the specified web static file exists in the Application Explorer; if it does not, a compiler error occurs.

Syntax

```
str webStaticFileStr(str pagename)
```

Parameters

PARAMETER	DESCRIPTION
pagename	The name of the web static file to validate.

Return Value

The name of the specified web static file, if it is valid.

Remarks

This is a compile-time function. For more information, see Overview.

Example

```
{
    str s;
    ;

    s = webStaticFileStr(AXEP);
    print "string for web static file AXEP is " + s;
    pause;
}
```

webUrlItemStr

Validates that the specified web URL item exists in the Application Explorer; if it does not, a compiler error occurs.

Syntax

```
str webUrlItemStr(class weburlitem)
```

Parameters

PARAMETER	DESCRIPTION
weburlitem	The name of the web URL item to validate.

Return Value

The name of the specified web URL item, if it is valid.

Remarks

This is a compile-time function. For more information, see Overview.

Example

```
{
    str s;
    ;

    s = webUrlItemStr(EPAdmin);
    print "string for web url item EPAdmin is " + s;
    pause;
}
```

webWebPartStr

Validates that the specified web part exists in the Application Explorer; if it does not, a compiler error occurs.

Syntax

```
str webWebpartStr(str resourcename)
```

Parameters

PARAMETER	DESCRIPTION
resourcename	The name of the web part to validate.

Return Value

The name of the specified web part, if it is valid.

Remarks

This is a compile-time function. For more information, see Overview.

Example

```
{
    str s;
    ;

    s = webWebpartStr(AxWebParts_cab);
    print "string for web part AxWebParts_cab is " + s;
    pause;
}
```

workflowApprovalStr

Retrieves the name of a workflow approval in the Application Object Tree (Application Explorer) as a string.

Syntax

```
str workflowapprovalstr(approval approval)
```

Parameters

PARAMETER	DESCRIPTION
approval	The Application Explorer name of the workflow approval.

Return Value

A string that represents the Application Explorer name of the workflow approval.

Remarks

Use this function instead of literal text to retrieve a string that contains the workflow approval name. If the workflow approval does not exist, the function generates a syntax error at compile time. This is a compile-time function. For more information, see Overview.

Example

The following code example sets the variable *str s* to **MyWorkflowApproval** which is the name of the workflow approval in the Application Explorer.

```
static void MyWorkflowApprovalStrExample(Args _args)
{
    str s;
    ;
    s = workflowapprovalstr(MyWorkflowApproval);
    print s;
    pause;
}
```

workflowCategoryStr

Retrieves the name of a workflow category in the Application Object Tree (Application Explorer) as a string.

Syntax

```
str workflowcategorystr(category category)
```

Parameters

PARAMETER	DESCRIPTION
category	The Application Explorer name of the workflow category.

Return Value

A string that represents the Application Explorer name of the workflow category.

Remarks

Use this function instead of literal text to retrieve a string that contains the workflow category name. If the workflow category does not exist, the function generates a syntax error at compile time. This is a compile-time function. For more information, see Overview.

Example

The following code example sets the variable *str s* to **MyWorkflowCategory** which is the name of the workflow category in the Application Explorer.

```
static void MyWorkflowCategoryStrExample(Args _args)
{
    str s;
    ;
    s = workflowcategorystr(MyWorkflowCategory);
    print s;
    pause;
}
```

workflowTaskStr

Retrieves the name of a workflow task in the Application Object Tree (Application Explorer) as a string.

Syntax

```
str workflowtaskstr(task task)
```

Parameters

PARAMETER	DESCRIPTION
task	The Application Explorer name of the workflow task.

Return Value

A string that represents the Application Explorer name of the workflow task.

Remarks

Use this function instead of literal text to retrieve a string that contains the workflow task name. If the workflow task does not exist, the function generates a syntax error at compile time. This is a compile-time function. For more information, see Overview.

Example

The following code example sets the variable *str s* to **MyWorkflowTask** which is the name of the workflow task in the Application Explorer.

```
static void MyWorkflowTaskStrExample(Args _args)
{
    str s;
    ;
    s = workflowtaskstr(MyWorkflowTask);
    print s;
    pause;
}
```

workflowTypeStr

Validates that the specified workflow type exists in the Application Explorer; if it does not, a compiler error occurs.

Syntax

```
str workflowTypeStr(str workflow)
```

Parameters

PARAMETER	DESCRIPTION
workflow	The name of the workflow type to validate.

Return Value

The name of the workflow type.

Remarks

This is a compile-time function. For more information, see Overview.

Example

```
static void workFlowTypeStrExample(Args _args)
{
    str s;
    ;
    s = workFlowTypeStr(BudgetAccountEntryType);
    print s;
    pause;
}
```

X++ runtime function resources

2/18/2021 • 3 minutes to read • Edit Online

This topic describes the X++ run-time functions.

The X++ language provides nearly 200 system functions that aren't part of any class and are executed at run time. Run-time functions are used for data type conversions, mathematical operations, and so on. Here are some common run-time functions:

- **str2Int** – Creates an int value from a str value.
- **abs** – Creates a positive real value from a real value that is either positive or negative.
- **conFind** – Retrieves the location of an element in a container.

Call run-time functions from .NET

The logic of the X++ run-time functions is also implemented in the following .NET assembly.

```
Microsoft.Dynamics.AX.Xpp.Support.DLL
```

Inside this assembly, the X++ run-time functions are implemented as static methods of the following class.

```
Microsoft.Dynamics.AX.Xpp.PredefinedFunctions
```

Categories and functions

The following table lists and describes only the categories of X++ functions. These categories are intended to help you understand the many functions. However, the categories don't represent any formal construct.

CATEGORY	DESCRIPTION
Business	Functions that enter financial data and calculate formulas. For more information, see X++ Business Run-Time Functions.
Container	Functions that operate on the container data type of X++. For more information, see X++ Container Run-Time Functions.
Conversion	Functions that translate data of one type into data of another type. For more information, see X++ Conversion Run-Time Functions.
Date	Functions that operate on the date data type. For more information, see X++ Date Run-Time Functions.
Math	Functions that perform mathematical calculations. For more information, see X++ Math Run-Time Functions.

CATEGORY	DESCRIPTION
Reflection	Functions that access the metadata about objects and return other metadata about them. For more information, see X++ Reflection Run-Time Functions.
Session	Functions that change or report on the context of the current user connection. For more information, see X++ Session Run-Time Functions.
String	Functions that operate on the str data type. For more information, see X++ String Run-Time Functions.
Other	beep, newGuid, sleep

Business

For more information, see X++ Business Run-Time Functions.

cTerm	ddb	dg	fV
idg	intvMax	intvName	intvNo
intvNorm	pmt	pt	pv
rate	sln	syd	term

Container

For more information, see X++ Container Run-Time Functions.

- conDel
- conFind
- conIns
- conLen
- conNull
- conPeek
- conPoke

Conversion

For more information, see X++ Conversion Run-Time Functions.

any2Date	any2Enum	any2Guid	any2Int
any2Int64	any2Real	any2Str	anytodate
anytoenum	anytoguid	anytoint	anytoint64

anytoreal	anytostr	char2Num	date2Num
date2Str	datetime2Str	enum2str	guid2Str
int2Str	int642Str	num2Char	num2Date
num2Str	str2Date	str2Datetime	str2Enum
str2Guid	str2Int	str2Int64	str2Num
str2Time	time2Str	uint2Str	

Date

For more information, see X++ Date Run-Time Functions.

dayName	dayOfMth	dayOfWk	dayOfYr
endMth	mkDate	mthName	mthOfYr
nextMth	nextQtr	nextYr	prevMth
prevQtr	prevYr	systemDateGet	systemDateSet
timeNow	today	wkOfYr	year

Math

For more information, see X++ Math Run-Time Functions.

abs	acos	asin	atan
corrFlagGet	corrFlagSet	cos	cosh
decRound	exp	exp10	frac
log10	logN	max	min
power	round	sin	sinh
tan	tanh	trunc	

Reflection

For more information, see X++ Reflection Run-Time Functions.

classIdGet	dimOf	fieldId2Name	fieldId2PName
fieldName2Id	indexId2Name	indexName2Id	refPrintAll
tableId2Name	tableId2PName	tableName2Id	typeOf

Session

For more information, see X++ Session Run-Time Functions.

curExt	curUserId	funcName	getCurrentPartition
getCurrentPartitionRecId	getPrefix	sessionId	prmIsDefault
runAs	setPrefix		

String

For more information, see X++ String Run-Time Functions.

match	strAlpha	strCmp	strColSeq
strDel	strFind	strFmt	strIns
strKeep	strLen	strLine	strLTrim
strLwr	strNFind	strPoke	strPrompt
strRem	strRep	strRTrim	strScan
strUpr	subStr		

beep

Emits a short sound from the speakers on the computer.

```
void beep()
```

beep example

```
static void beepExample(Args _args)
{
        beep();
}
```

newGuid

Creates a globally unique identifier (GUID).

```
guid newGuid()
```

Return value

A GUID.

newGuid example

The following example creates a GUID.

```
static void newGuidExample(Args _arg)
{
    guid myGuid;

    myGuid = newguid();
    print strfmt("The GUID is: %1", myGuid);
}
```

sleep

Pauses the execution of the current thread for the specified number of milliseconds.

```
int sleep(int _duration)
```

Parameters

PARAMETER	DESCRIPTION
_duration	The number of milliseconds to pause.

sleep return value

The number of milliseconds that the thread actually paused.

Example

```
static void sleepExample(Args _arg)
{
    int seconds = 10;
    int i;

    i = sleep(seconds*1000);
    print "job slept for " + int2str(i/1000) + " seconds";
}
```

X++ business runtime functions

2/18/2021 • 8 minutes to read • Edit Online

This topic describes the business run-time functions.

These functions enter financial data and calculate formulas.

cTerm

Calculates the number of periods that are required for the current investment value to yield a target value.

Syntax

```
real cTerm(real interest, real future_value, real current_value)
```

Parameters

PARAMETER	DESCRIPTION
interest	The interest rate.
future_value	The target value.
current_value	The current investment value.

Return value

The number of periods that are required in order to reach *future_value*.

Remarks

The *current_value* and *future_value* parameters must have the same prefixed sign (plus or minus).

Example

```
static void cTermExample(Args _arg)
{
    real r;
    ;
    r = cTerm(10.0, 500.00, 100.00);
    print "The cTerm is " + num2Str(r, 2, 2, 1, 1);
    pause;
}
```

ddb

Calculates the accelerated depreciation of an asset.

Syntax

```
real ddb(real price, real scrap, real life, int period)
```

Parameters

PARAMETER	DESCRIPTION
price	The purchase price of the asset.
scrap	The residual value of the asset that has been written off.
life	The expected lifetime of the asset.
period	The period to calculate depreciation over.

Return value

The depreciation of the asset.

Remarks

The book value for a specific period is equal to the purchase price minus the accumulated depreciation for previous periods:

- Book value for Period 1 = Price
- Book value for Period 2 = Book value for Period 1 – Depreciation for Period 1
- Book value for Period n = Book value for Period (n–1) – Depreciation for Period (n–1)

There are three variations for the calculation of depreciation: If Period > Life:

- Depreciation = 0

If (Book value for Period n) – ((Book value for Period n) × 2 ÷ Life) < Residual value:

- Depreciation = (Book value for Period n) – Residual value

In all other cases: Depreciation = (Book value for Period n) × 2 ÷ Life The **syd** and **sln** functions also calculate the depreciation of an asset. The **syd** and **ddb** functions enables higher depreciation for the earlier years, whereas **sln** calculates a linear depreciation.

```
ddb(12000,2000,10,1); //Returns the value 2400.
ddb(12000,2000,10,3); //Returns the value 1536.
```

dg

Calculates the contribution ratio, which is based on the sales price and the purchase price. If the value of the *sale* parameter is **0.0**, the calculation can't be done.

Syntax

```
real dg(real sale, real purchase)
```

Parameters

PARAMETER	DESCRIPTION
sale	The sale price.
purchase	The purchase price.

Return value

The contribution ratio.

Remarks

```
dg(1000,300); //Returns the value 0.7.
dg(100,30); //Returns the value 0.7.
dg(20000, 11000); //Returns the value 0.45.
```

fV

Calculates the future value of an investment.

Syntax

```
real fV(real amount, real interest, real life)
```

Parameters

PARAMETER	DESCRIPTION
amount	The amount that was paid in during each period.
interest	The interest rate.
life	The number of investment periods.

Return value

The future value of the investment.

Remarks

```
fV(100,0.14,10); //Returns the value 1933.73.
fV(400,0.10,5); //Returns the value 2442.04.
```

idg

Calculates the sale price, based on the purchase price and the contribution ratio.

```
real idg(real purchase, real contribution_ratio)
```

Parameters

PARAMETER	DESCRIPTION
purchase	The purchase price.
contribution_ratio	The contribution ratio.

Return value

The sale price.

Remarks

If the contribution ratio is equal to `1.0`, the calculation can't be done. The `idg` function is the inverse of the `dg`

function.

```
idg(300,0.7); //Returns the value 1000.
idg(11000,0.45); //Returns the value 20000.
```

intvMax

Retrieves the number of intervals for the specified period when the period is divided into parts as specified by the *func* parameter.

```
int intvMax(date input_date, date ref_date, int func)
```

Parameters

PARAMETER	DESCRIPTION
input_date	The end of the period, which must be later than the *ref_date* parameter.
ref_date	The start of the period.
func	A **IntvScale** system enumeration value that indicates the division unit.

Remarks

Here are the possible values for the *func* parameter:

- None
- YearMonthDay
- YearMonth
- Year
- MonthDay
- Month
- Day
- YearQuarter
- Quarter
- YearWeek
- Week
- WeekDay

Example

```
static void intvMaxExample()
{
    date refDate = str2Date("4/9/2007", 213);
    date inputDate = str2Date("10/5/2007", 213);
    int numberOfIntervals;
    ;
    numberOfIntervals = intvMax(inputDate, refDate, intvScale::YearMonth);
    print numberOfIntervals;
    pause;
}
```

intvName

Returns the name of the interval that is the specified number of intervals ahead of the specified date.

```
str intvName(date input_date, int col, enum func)
```

Parameters

PARAMETER	DESCRIPTION
input_date	A date in the first interval.
col	The number of intervals ahead of the date that is specified by the *input_date* parameter.
func	An **intvScale** enumeration value.

Return value

The name of the interval.

Remarks

For example, if the *func* parameter is the **IntvScale::WeekDay** enumeration value, this method returns the name of the weekday. If the *func* parameter is the **IntvScale::Week** enumeration value, this method returns a string that contains the number of the week.

Example

```
static void intvNameExample(Args _args)
{
    date refDate = 2672010;
    str name;
    ;
    name = intvName(refDate, 3,  intvScale::WeekDay);
    Global::info(strfmt("%1 is the output, which indicates the day of the week 3 days after 26\7\2010.",
name));
}
/**** Infolog display.
Message (09:56:55 am)
Thu is the output, which indicates the day of the week 3 days after 2672010.
****/
```

intvNo

Calculates the number of intervals between two dates when you divide the time into the specified intervals.

Syntax

```
int intvNo(date input_date, date ref_date, int func)
```

Parameters

PARAMETER	DESCRIPTION
input_date	A date that indicates the end of the period

PARAMETER	DESCRIPTION
ref_date	A date that indicates the start of the period.
func	An **intvScale** enumeration value.

Return value

The number of intervals between the dates that are specified by the *ref_date* and *input_date* parameters.

Example

```
static void intvNoExample(Args _args)
{
    date inputDate = str2Date("1/1/2007", 213);
    date refDate = str2Date("3/1/2007", 213);
    int noOfIntervals;
    ;
    noOfIntervals = intvNo(refDate, inputDate, intvScale::Month);
    print noOfIntervals;
    pause;
    //noOfIntervals now holds the difference in months between March and January (2).
}
```

intvNorm

Returns the normalized date for the period.

Syntax

```
date intvNorm(date input_date, date ref_date, int func)
```

Parameters

PARAMETER	DESCRIPTION
input_date	The end of the period, which must be later than the date that is specified by the *ref_date* parameter.
ref_date	The start of the period.
func	An **intvScale** enumeration value that indicates the interval division unit.

Return value

The normalized date for the period.

Remarks

The returned date will equal the date of the first day in the interval in which the date that is specified by the *ref_date* parameter exists.

Example

```
static void example()
{
    print intvNorm(today(), today()-1, IntVScale::WeekDay);
    pause;
}
```

pmt

Calculates the amount that must be paid every period to repay a loan.

Syntax

```
real pmt(real principal, real interest, real life)
```

Parameters

PARAMETER	DESCRIPTION
principal	The amount that was originally borrowed.
interest	The interest that is applied each period to the amount that was borrowed.
life	The number of periods that the loan is repaid over.

Return value

The amount that must be paid every period.

Remarks

The *life* and *interest* parameters must be expressed in the same time units. The value of the *life* parameter must be more than **0.0**.

Example

```
pmt(4000,0.14,4); //Returns the value 1372.82.
pmt(10000,0.10,20); //Returns the value 1174.60.
```

pt

Retrieves the sum of a number plus a specified percentage of that number.

Syntax

```
real pt(real amount, real percentage)
```

Parameters

PARAMETER	DESCRIPTION
amount	The original number.
percentage	The percentage supplement.

Return value

The number that is equal to ((*amount* *× *percentage*) + *amount*).

Remarks

```
pt(2000.0,0.10); //Returns the value 2200.0.
pt(20.0,0.10); //Returns the value 22.0.
```

pv

Calculates the present value of an annuity, where an amount is received over multiple periods and the interest rate is deducted for each period.

Syntax

```
real pv(real amount, real interest, real life)
```

Parameters

PARAMETER	DESCRIPTION
amount	The amount that is paid during each period.
interest	The interest rate.
life	The number of times that the value that is specified by the *amount* parameter is paid.

Return value

The current value of an annuity.

Remarks

```
pv(300,0.14,4); //Returns the value 874.11.
```

rate

Calculates the interest that is required for the current investment value to attain the future value over the specified number of periods.

Syntax

```
real rate(real _future_value, real _current_value, real _terms)
```

Parameters

PARAMETER	DESCRIPTION
_future_value	The future value of the investment.
_current_value	The current value of the investment.
_terms	The number of periods that the investment spans.

Return value

The calculated interest rate.

Remarks

```
rate(10000,1000,20); //Returns the value 0.12.
```

sln

Retrieves the constant depreciation amount for the specified asset for each depreciation period.

Syntax

```
real sln(real price, real scrap, real life)
```

Parameters

PARAMETER	DESCRIPTION
price	The purchase price of the asset.
scrap	The scrap value of the asset.
life	The number of periods in the expected life of the asset.

Return value

The depreciation amount.

Example

```
static void slnExample(Args _arg)
{
    real r;
    ;
    r = sln(100.00, 50.00, 50.00);
    print r;
    pause;
}
```

syd

Calculates the depreciation of an asset over a specified period.

Syntax

```
real syd(real _price, real _scrap, real _life, int _period)
```

Parameters

PARAMETER	DESCRIPTION
_price	The purchase price of the asset.
_scrap	The scrap value of the asset.

PARAMETER	DESCRIPTION
_life	The expected life of the asset (the number of periods).
_period	The period to calculate depreciation for.

Return value

The amount of depreciation over the specified period.

Remarks

In contrast to the **sln** function, the **syd** function can allow for an accelerated depreciation of the asset. As with the **ddb** function, this enables higher depreciation during the early periods of the life of an asset.

Example

In the following examples, the periodic depreciation is calculated for an asset that has a purchase price of 10,000, a scrap value of 2,000, and a life of 5. In comparison, **sln(10000,2000,5)** would calculate 1600.00 for each period.

```
// Returns the value 2666.67 (for the 1st period).
syd(10000,2000,5,1);
// Returns the value 2133.33 (for the 2nd period).
syd(10000,2000,5,2);
// Returns the value 1600.00 (for the 3rd period).
syd(10000,2000,5,3);
// Returns the value 1066.67 (for the 4th period).
syd(10000,2000,5,4);
// Returns the value 533.33 (for 5th - and final- period).
syd(10000,2000,5,5);
```

term

Calculates the number of periods that an investment must run for.

Syntax

```
real term(real amount, real interest, real future_value)
```

Parameters

PARAMETER	DESCRIPTION
amount	The amount of the periodic investment.
interest	The interest rate for each period.
future_value	The future value that is anticipated for the investment

Return value

The number of periods that the investment must run for.

Example

```
static void termExample(Args _args)
{
    print term(400,0.08,5000);  //returns the value '9.01'.
    print term(100,0.14,3000);  //returns the value '12.58'.
    pause;
}
```

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

X++ container runtime functions

2/18/2021 • 4 minutes to read • Edit Online

This topic describes the container run-time functions.

These functions manipulate the contents of containers.

conDel

Removes the specified number of elements from a container.

Syntax

```
container conDel(container container, int start, int number)
```

Parameters

PARAMETER	DESCRIPTION
container	The container to remove elements from.
start	The one-based position at which to start removing elements.
number	The number of elements to delete.

Return value

A new container that doesn't include the removed elements.

Example

```
static void conDelExample(Args _args)
{
    container c = ["Hello world", 1, 3.14];
        // Deletes the first two items from the container.
        c = conDel(c, 1, 2);
}
```

conFind

Locates the first occurrence of an element or a sequence of elements in a container.

Syntax

```
int conFind (container container, anytype element,... )
```

Parameters

PARAMETER	DESCRIPTION
container	The container to search.

PARAMETER	DESCRIPTION
element	One or more elements to search for, separated by commas.

Remarks

If several elements are specified in the sequence, they must be separated by commas and specified in the correct sequence. The elements can be of any data type.

Return value

0 if the item was not found; otherwise, the sequence number of the item.

Example

```
static void conFindExample(Args _args)
{
    container c = ["item1", "item2", "item3"];
    int i;
    int j;
    i = conFind(c, "item2");
    j = conFind(c, "item4");
    print "Position of 'item2' in container is " + int2Str(i);
    print "Position of 'item4' in container is " + int2Str(j);
}
```

conIns

Inserts one or more elements into a container.

Syntax

```
container conIns (container container, int start, anytype element, ... )
```

Parameters

PARAMETER	DESCRIPTION
container	The container to insert elements into.
start	The position to insert elements at.
element	One or more elements to insert, separated by commas.

Return value

A new container that contains the inserted elements.

Remarks

The first element of the container is specified by the number **1**. To insert after the n element, the *start* parameter should be n+1. You can also use the **+=** operator to add values of any type to a container. For example, to create a container that contains the squared values of the first 10 loop iterations, use the following code.

```
int i;
container c;

for (i = 1; i < = 10; i++)
{
    c += i*i;
}
```

Example

```
static void conInsExample(Args _arg)
{
    container c;
    int i;

    c = conIns(c,1,"item1");
    c = conIns(c,2,"item2");
    for (i = 1 ; i <= conLen(c) ; i++)
    {
        // Prints the content of a container.
        print conPeek(c, i);
    }
}
```

conLen

Retrieves the number of elements in a container.

Syntax

```
int conLen(container container)
```

Parameters

PARAMETER	DESCRIPTION
container	The container to count the number of elements in.

Return value

The number of elements in the container.

Example

```
static void conLenExample(Args _arg)
{
    container c;
    int i;

    c = conins(["item1", "item2"], 1);
    for (i = 1 ; i <= conLen(c) ; i++)
    {
            print conPeek(c, i);
    }
}
```

conNull

Retrieves an empty container.

```
container conNull()
```

Remarks

Use this function to explicitly dispose of the contents of a container.

Return value

An empty container.

Example

```
static void conNullExample(Args _arg)
{
    container c = ["item1", "item2", "item3"];

    print "Size of container is " + int2str(conLen(c));
    // Set the container to null.
    c = conNull();
    print "Size of container after conNull() is " + int2Str(conLen(c));
}
```

conPeek

Retrieves a specific element from a container and converts it into another data type, if conversion is required.

Syntax

```
anytype conPeek(container container, int number)
```

Parameters

PARAMETER	DESCRIPTION
container	The container to return an element from.
number	The position of the element to return. Specify **1** to get the first element. An invalid position number, such as **-3**, **0**, or a number that is higher than the length of the container, might cause unpredictable errors.

Return value

The element in the container at the position that is specified by the *number* parameter. The **conPeek** function automatically converts the peeked item into the expected return type. Strings can automatically be converted into integers and real numbers, and integers and real numbers can be converted into strings.

Example

```
static void main(Args _args)
{
    container cnI, cnJ;
    int i, j;
    anytype aty;
    info("container cnI ...");
    cnI = ["itemBlue", "itemYellow"];
    for (i=1; i <= conLen(cnI); i++)
    {
        aty = conPeek(cnI, i);
        info(int2str(i) + " :  " + aty);
    }

    info("container cnJ ...");
    cnJ = conIns(cnI, 2, "ItemInserted");
    for (j=1; j <= conLen(cnJ); j++)
    {
        aty = conPeek(cnJ, j);
        info(int2str(j) + " :  " + aty);
    }
}
/***  Output pasted from InfoLog ...
Message (10:20:03 am)
container cnI ...
1 :  itemBlue
2 :  itemYellow
container cnJ ...
1 :  itemBlue
2 :  ItemInserted
3 :  itemYellow
***/
```

conPoke

Modifies a container by replacing one or more of the existing elements.

Syntax

```
container conPoke(container container, int start, anytype element, ...)
```

Parameters

PARAMETER	DESCRIPTION
container	The container to modify.
start	The position of the first element to replace.
element	One or more elements to replace, separated by commas.

Return value

A new container that includes the new elements.

Remarks

The first element of the container is specified by the number **1**.

Example

```
static void conPokeExample(Args _arg)
{
    container c1 = ["item1", "item2", "item3"];
    container c2;
    int i;
    void conPrint(container c)
    {
        for (i = 1 ; i <= conLen(c) ; i++)
        {
            print conPeek(c, i);
        }
    }

    conPrint(c1);
    c2 = conPoke(c1, 2, "PokedItem");
    print "";
    conPrint(c2);
}
```

X++ conversion runtime functions

2/18/2021 • 18 minutes to read • Edit Online

This topic describes the conversion run-time functions.

any2Date

Converts an **anytype** value to a **date** value.

```
date any2Date(anytype object)
```

Parameters

PARAMETER	DESCRIPTION
object	The value to convert to a date.

Return value

A **date** value.

Remarks

The *object* parameter can be of most data types, but useful output is obtained when it's of the **str** or **int** type. Inappropriate content generates a run-time error.

Example

```
static void any2DateExample(Args _args)
{
    date myDate;
    str s;
    int i;
    s = "2010 6 17"; // A string object, of yyyy mm dd.
    myDate = any2Date(s);
    Global::info(strFmt("%1  is output, from input of "2010 6 17"", myDate));
    i = 40361; // An int object, which represents the number of days from 1900/01/01.
    myDate = any2Date(i);
    Global::info(strFmt("%1  is output, from input of 40361", myDate));
}
/**** Infolog display.
Message (04:44:15 pm)
6/17/2010 is output, from input of "2010 6 17"
7/4/2010 is output, from input of 40361
****/
```

any2Enum

Converts an **anytype** value to the **Name** property value of an element in the target enum.

```
enum any2Enum(anytype object)
```

Parameters

PARAMETER	DESCRIPTION
object	The value to match the **Value** property of an element in the target enum.

Return value

The value of the **Name** property for whichever element in the target enum has a **Value** property that matches the input parameter.

Remarks

The *object* parameter can be of most data types, but useful data is obtained only when you use a parameter of the **str** or **int** type. This input *object* parameter refers to the **Value** property of an individual element in the target enum.

Example

```
static void any2EnumExample(Args _args)
{
    NoYes myNoYes;  // NoYes is an enum.
    int i;
    str s;
    i = 0;  // An int that will be converted.
    myNoYes = any2Enum(i);
    Global::info(strfmt("'%1' - is the output, from input of the %2 as int.", myNoYes, i));
    s = "1";  // A str that will be converted.
    myNoYes = any2Enum(s);
    Global::info(strfmt("'%1' - is the output, from input of the %2 as str.", myNoYes, s));
    /**** Infolog display.
    Message (01:05:32 pm)
    'No' - is the output, from input of the 0 as int.
    'Yes' - is the output, from input of the 1 as str.
    ****/
}
```

any2Guid

Converts the specified **anytype** object to a GUID object.

```
guid any2Guid(anytype object)
```

Parameters

PARAMETER	DESCRIPTION
object	The value to convert to a GUID object.

Return value

A GUID object.

any2Int

Converts an **anytype** value to an **int** value.

```
int any2Int(anytype object)
```

Parameters

PARAMETER	DESCRIPTION
object	The value to convert.

Return value

An **int** value.

Remarks

The *object* parameter can be of most data types, but useful data is obtained only when you use parameters of the **enum**, **real**, or **str** type.

Example

```
static void any2IntExample(Args _args)
{
    int myInt;
    str s;
    NoYes a;
    real r;
    s = "31";
    myInt = any2Int(s);
    Global::info(strfmt("%1 is the output, from input of 31 as a str value.", myInt));
    a = NoYes::No;
    myInt = any2Int(a);
    Global::info(strfmt("%1 is the output, from input of NoYes::No as an enum value.", myInt));
    r = 5.34e2;
    myInt = any2Int(r);
    Global::info(strfmt("%1 is the output, from the input of 5.34e2 as a real value.", myInt));
}
/**** Infolog display.
Message (02:23:59 pm)
31 is the output, from input of 31 as a str value.
0 is the output, from input of NoYes::No as an enum value.
534 is the output, from the input of 5.34e2 as a real value.
****/
```

any2Int64

Converts an **anytype** object to an **int64** object.

```
int64 any2Int64(anytype object)
```

Parameters

PARAMETER	DESCRIPTION
object	The **anytype** object to convert.

Return value

An **int64** object.

any2Real

Converts an **anytype** value to a **real** value.

```
real any2Real(anytype object)
```

Parameters

PARAMETER	DESCRIPTION
object	The value to convert.

Return value

A **real** value.

Remarks

The *object* parameter can be of most data types, but useful output is obtained for input elements of the **date**, **int**, **enum**, and **str** types.

Example

```
static void any2RealExample(Args _args)
{
    real myReal;
    str s;
    int i;
    NoYes a;
    s = "5.12";
    myReal = any2Real(s);
    Global::info(strfmt("%1 is the output from the input of 5.12 as a str object", myReal));
    i = 64;
    myReal = any2Real(i);
    Global::info(strfmt("%1 is the output from the input of 64 as an int object", myReal));
    a = NoYes::Yes;
    myReal = any2Real(a);
    Global::info(strfmt("%1 is the output from the input of NoYes::Yes as an enum object", myReal));
}
/****Infolog display.
Message (02:43:57 pm)
5.12 is the output from the input of 5.12 as a str object
64.00 is the output from the input of 64 as an int object
1.00 is the output from the input of NoYes::Yes as an enum object
****/
```

any2Str

Converts an **anytype** value to a **str** value.

```
str any2Str(anytype object)
```

Parameters

PARAMETER	DESCRIPTION
object	The value to convert.

Return value

A **str** value.

Remarks

The *object* parameter can be of most data types, but useful output is obtained from input elements of the **date**, **int**, and **enum** types.

Example

```
static void any2StrExample(Args _args)
{
    str myStr;
    anytype a;
    a = "Any to string";
    myStr = any2Str(a);
    Global::info(strFmt("%1 is output, from input of Any to string as a str value", myStr));
    a = NoYes::Yes;
    myStr = any2Str(a);
    Global::info(strFmt("%1 is output, from input of NoYes::Yes as an enumeration", myStr));
}
/****Infolog Display
Message (09:08:46 am)
Any to string is output, from input of Any to string as a str value
1 is output, from input of NoYes::Yes as an enumeration
****/
```

anytodate

See any2Date.

anytoenum

See any2Enum.

anytoguid

See any2Guid.

anytoint

See any2Int.

anytoint64

See any2Int64.

anytoreal

See any2Real.

anytostr

See any2Str.

char2Num

Converts a character in a string to the ASCII value of the character.

```
int char2Num(str text, int position)
```

Parameters

PARAMETER	DESCRIPTION
text	The string that contains the character.
position	The position of the character in the string.

Return value

The ASCII value of the character as an **int** object.

Remarks

```
char2Num("ABCDEFG",3); //Returns the numeric value of C, which is 67.
char2Num("ABCDEFG",1); //Returns the numeric value of A, which is 65.
```

date2Num

Converts a date to an integer that corresponds to the number of days since January 1, 1900.

```
int date2Num(date _date)
```

Parameters

PARAMETER	DESCRIPTION
_date	The date to convert.

Return value

The number of days between January 1, 1900, and the specified date.

Example

```
//Returns the value377.
date2Num(1311901);
static void date2NumExample(Args _arg)
{
    date d = today();
    int i;
    i = date2Num(d);
    print i;
}
```

date2Str

Converts the specified date to a string.

```
str date2Str(date date, int sequence, int day, int separator1, int month, int separator2, int year [, int
flags = DateFlags::None])
```

Parameters

PARAMETER	DESCRIPTION
date	The date to convert.
sequence	A three-digit number that indicates the sequence for the components of the date: **1** for day, **2** for month, and **3** for year.
day	An enumeration value that indicates the format for the day component of the date.
separator1	An enumeration value that indicates the separator to use between the first two components of the date.
month	An enumeration value that indicates the format for the month component of the date.
separator2	An enumeration value that indicates the separator to use between the last two components of the date.
year	An enumeration value that indicates the format for the year component of the date.
flags	A **DateFlags** enumeration value that indicates whether the language settings on the local computer should be used to calculate the proper left-to-right or right-to-left sequence in the returned string.

Return value

A string that represents the specified date.

Remarks

MorphX allocates valid values to the formatting parameters if the specified values aren't valid. To use the date format that the user specified in Regional Settings, use the **strFmt** or **date2Str** function and specify **-1** in all the formatting parameters. When the regional settings control the date format, the settings can change from user to user. If **-1** is used for either *separator* parameter, both separators default to Regional Settings. The *sequence* parameter values must be any three-digit number that contains exactly one occurrence of each the digits 1, 2 and 3. The digits 1, 2, and 3 represent day, month, and year, respectively. For example, **321** produces the sequence year, month, and day. Or the value can be **-1** to use Regional Settings. No enumeration type should be used for this parameter, because numbers such as 321 exceed the range of valid values for enumeration values, which is 0 through 250, inclusive. The default value of the *flags* parameter is the **DateFlags::None** enumeration value, which means no left-to-right or right-to-left sequence processing is done.

Example

The following example displays the current date in the sequence of year, month, and day.

```
static void Job2(Args _args)
{
    date currentDate = today();
    str s;
    int iEnum;
    s = date2Str
    (currentDate,
        321,
        DateDay::Digits2,
        DateSeparator::Hyphen, // separator1
        DateMonth::Digits2,
        DateSeparator::Hyphen, // separator2
        DateYear::Digits4
    );
    info("Today is:  " + s);
}
/** Example Infolog output
Message (12:36:21 pm)
Today is:  2009-01-13
**/
```

datetime2Str

Converts a **utcdatetime** value into a string.

```
str datetime2Str(utcdatetime datetime [, int flags = DateFlags::None])
```

Parameters

PARAMETER	DESCRIPTION
datetime	The **utcdatetime** value to convert.
flags	A **DateFlags** enumeration value that indicates whether to use local settings for right-to-left output.

Return value

A string that represents the **utcdatetime** value that was specified as the *datetime* parameter.

Remarks

Null date-time input

If the minimum **utcdatetime** value is specified for the *datetime* parameter, the **datetime2Str** function treats it as a null input value. This causes the function to return an empty string. The date-time **1900-01-01T00:00:00** is returned by the **DateTimeUtil::minValue** method. This minimum value is treated as null.

Right-to-left local settings

The default behavior of this function is to generate the string in left-to-right sequence, where the year portion is leftmost. However, the *flags* parameter value of the **DateFlags::FormatAll** enumeration value directs the function to generate the string in right-to-left sequence if the local settings are configured for right-to-left. The format of the **toStr** method of the **DateTimeUtil** class is unaffected by regional settings.

Example

```
static void jobTestDatetime2str( Args _args )
{
    utcdatetime utc2 = 1959-06-17T15:44:33;
    str s3;
    s3 = datetime2Str( utc2 );
    info( s3 );
}
```

enum2Str

Converts the specified enumerated text to a character representation.

```
str enum2Str(enum enum)
```

Parameters

PARAMETER	DESCRIPTION
enum	The enumerated text to convert.

Return value

The value of the enumeration as a string.

Example

The following example returns the string "Not included." This is the label for the **IncludeNot** value of the **ListCode** enumeration type.

```
static void enum2StrExample(Args _arg)
{
    ListCode l;
    l =  ListCode::IncludeNot;
    print enum2Str(l);
}
```

guid2Str

Converts the specified GUID object to the equivalent string.

```
str guid2String(guid _uuid)
```

Parameters

PARAMETER	DESCRIPTION
_uuid	The GUID object to convert.

Return value

The string equivalent of the specified GUID object.

Example

```
static void guid2StrExample()
{
    guid _guid;
    str stringGuid;
    _guid = Global::guidFromString("{12345678-1234-1234-1234-123456789abc}");
    print strfmt("GUID is %1", _guid);
    stringGuid = guid2str(_guid);
    info("String GUID is " + stringGuid);
}
/**** Output to Infolog
String GUID is {12345678-1234-1234-1234-123456789ABC}
****/
```

int2Str

Converts an integer to the equivalent string.

```
str int2Str(int integer)
```

Parameters

PARAMETER	DESCRIPTION
integer	The integer to convert.

Return value

A string representation of the integer.

Example

```
static void int2StrExample(Args _arg)
{
    print "This is int2Str, value is " + int2Str(intMax());
    print "This is int642Str, value is " + int642Str(int64Max());
}
```

int642Str

Converts the specified *integer* parameter to the equivalent text string.

```
str int642Str(int64 integer)
```

Parameters

PARAMETER	DESCRIPTION
integer	The int64 to convert to a string.

Return value

The equivalent text string of the *integer* parameter.

Example

```
static void example()
{
    print "This is int2Str, value is " + int2Str(intMax());
    print "This is int642Str, value is " + int642Str(int64Max());
}
```

num2Char

Converts an integer to the corresponding ASCII character.

```
str num2Char(int figure)
```

Parameters

PARAMETER	DESCRIPTION
figure	The integer to convert to a character.

Return value

The character that is represented by the specified integer.

Example

```
static void num2CharExample(Args _arg)
{
    str s;
    s = num2Char(42);
    // Prints an asterisk * -the character represented by 42.
    print s;
}
```

num2Date

Retrieves the date that corresponds to the specified number of days after January 1, 1900.

```
date num2Date(int _days)
```

Parameters

PARAMETER	DESCRIPTION
_days	The number of days after January 1, 1900 to return the date for. **Note:** The first valid date is January 1, 1901. Therefore, the **num2Date** function doesn't return a valid date unless *_days* is more than **365**.

Return value

The date that is the number of days that is specified by the *_days* parameter after January 1, 1900.

Remarks

```
num2Date(366); //Returns the date 01/01/1901 (1 January 1901).
```

num2Str

Converts a real number to a string.

```
str num2Str(real number, int character, int decimals, int separator1, int separator2)
```

Parameters

PARAMETER	DESCRIPTION
number	The real number to convert to a string.
character	The minimum number of characters that are required in the text.
decimals	The required number of decimal places.
separator1	A **DecimalSeparator** enumeration value.
separator2	A **ThousandSeparator** enumeration value.

Return value

A string that represents the number.

Remarks

For the *decimals* parameter, the maximum value is **16**. If a larger number is used, this method obtains a value for the *decimals* parameter from the local computer instead. In both cases, rounding occurs. Here are the possible enumeration values for the *separator1* parameter:

- **99** – Auto (the formatting settings of the user determine what decimal separator is used), enumeration value DecimalSeparator::Auto
- **1** – Dot (.), enumeration value DecimalSeparator::Dot
- **2** – Comma (,), enumeration value DecimalSeparator::Comma

Here are the possible values for the *separator2* parameter:

- **99** – Auto (the formatting settings of the user determine what thousand separator is used)
- **0** – None (no thousand separator), enumeration value ThousandSeparator::None
- **1** – Dot (.), enumeration value ThousandSeparator::Dot
- **2** – Comma (,), enumeration value ThousandSeparator::Comma
- **3** – Apostrophe ('), enumeration value ThousandSeparator::Apostrophe
- **4** – Space ( ), enumeration value ThousandSeparator::Space

Example

In the following code example, the first call to the **num2str** method provides **16** for the *decimals* parameter, and the second call provides **17**.

```
static void Job_Num2Str(Args _args)
{
    real realNum = 0.1294567890123456777; // 19 decimals places.
    info(Num2Str(realNum, 0, 16, DecimalSeparator::Dot, ThousandSeparator::Space)); // 16 decimal places
    info(Num2Str(realNum, 0, 17, DecimalSeparator::Dot, ThousandSeparator::Space)); // 17 decimal places
}
```

Output

The messages are in the following Infolog output. The first number in the output contains 16 decimal place digits, whereas the second number contains only two decimal place digits.

```
Message (10:18:12)
0.1294567890123457
0.13
```

str2Date

Converts the specified string to a **date** value.

```
date str2Date(str _text, str _sequence)
```

Parameters

PARAMETER	DESCRIPTION
_text	The string to convert to a **date** value.
_sequence	A three-digit integer that describes the positions of the day, month, and year in the string to convert.

Return value

A **date** value.

Remarks

Use the following values to specify the positions of the day, month, and year in the *_sequence* parameter:

- **Day:** 1
- **Month:** 2
- **Year:** 3

For example, if the sequence in the string is month, year, and then day, the *_sequence* parameter must be **231**. A **0** (zero) date is returned if the input parameters specify an invalid date. The following two examples specify an invalid date.

```
str2Date("31/12/44", 123) // Year must be four digits to reach the minimum of January 1 1901.
str2Date("31/12/2044", 213) // 213 means the month occurs first in the string, but 31 cannot be a month.
```

Example

```
static void str2DateExample(Args _arg)
{
    date d;
    d = str2Date("22/11/2007", 123);
    print d;
}
```

str2Datetime

Generates a **utcdatetime** value from the specified string of date and time information.

```
utcdatetime str2datetime( str text, int sequence )
```

Parameters

PARAMETER	DESCRIPTION
text	The string to convert to a **utcdatetime** value.
sequence	A three-digit number that describes the sequence of the date components in the *text* parameter.

Return value

A **utcdatetime** value that represents the specified date and time.

Remarks

The syntax requirements for the date portion of the *text* parameter are flexible. The variety of valid formats is the same as in the **date2str** function. Each of the following calls to **str2datetime** is valid, and all of them produce the same output.

```
utc3 = str2datetime( "1985/02/25 23:04:59" ,321 );
utc3 = str2datetime( "Feb-1985-25 11:04:59 pm" ,231 );
utc3 = str2datetime( "2 25 1985 11:04:59 pm" ,123 );
```

Each component of the date time is represented by a digit in the *sequence* parameter:

- **1** – Day
- **2** – Month
- **3** – Year

For example, year, month, day order is **321**. All valid values contain each of these three digits exactly one time. If the value of the *sequence* parameter isn't valid, the regional settings are used to interpret the input *text* parameter. If the input parameters describe an invalid date and time, an empty string is returned.

Example

```
static void JobTestStr2datetime( Args _args )
{
    utcdatetime utc3;
    str sTemp;
    utc3 = str2datetime( "1985/02/25 23:04:59" ,321 );
    sTemp = datetime2str( utc3 );
    print( "sTemp == " + sTemp );
}
```

str2Enum

Retrieves the enum element for which the localized **Label** property value matches the input string.

```
enum str2Enum(enum _type, str _text)
```

Parameters

PARAMETER	DESCRIPTION
_type	A variable that is declared of the **enum** type.
_text	The localized **Label** property text of the target element in the enum.

Return value

An element of the target enum, which also represents an int.

Remarks

The related function **enum2str** returns the value of a **Label** property from one element in the enum. The value that is returned by **enum2str** function can be the input for the *_type* parameter of the **str2enum** function. An appropriate value for the *_text* parameter is **enum2Str(BankAccountType::SavingsAccount)**. Each element of an enum has a **Name** property and a **Label** property. In a fresh install, the **Name** values are almost always English words. In the English edition, the **Label** property value is almost always the same as the **Name** value. However, in non-English editions, the **Label** values are localized and therefore don't match the **Name** values.

Example

To avoid string mismatches that are caused by localization to other spoken languages, we recommend that you use the **enum2str** function to generate the input into the **str2enum** function. The following example shows the appropriate way to use the **str2enum** function together with the **enum2str** function.

```
static void str2Enum_AcrossLangs(Args _arg)
{
    BankAccountType bat;
    str sEnumValueLabelLocalized;
    int nInt;
    // enum2str.
    sEnumValueLabelLocalized = enum2str(BankAccountType::SavingsAccount);
    info("Localized friendly string: "
        + sEnumValueLabelLocalized);
    // str2enum.
    bat = str2Enum(bat, sEnumValueLabelLocalized);
    nInt = bat;
    info("nInt = " + int2str(nInt));
    /********** Actual output:
    Message (04:32:12 pm)
    Localized friendly string: Savings account
    nInt = 1
    **********/
}
```

str2Guid

Converts a string to a GUID object.

```
Guid str2Guid(str text)
```

Parameters

PARAMETER	DESCRIPTION
guid	A string that represents a GUID.

Return value

A GUID that is represented by the input string.

Remarks

For example, a valid value for the *guid* parameter is **{12345678-1234-abCD-3456-123456789012}**, either with or without the braces.

str2Int

Converts a string to the equivalent integer.

```
int str2Int(str _text)
```

Parameters

PARAMETER	DESCRIPTION
_text	The string to convert to an integer.

Return value

The integer equivalent of the specified string.

Example

```
static void str2IntExample(Args _arg)
{
    int i;
    i = str2Int("1234567890");
    print "i = " + int2Str(i);
}
```

str2Int64

Converts a string into an **Int64** value.

```
int str2Int64(str text)
```

Parameters

PARAMETER	DESCRIPTION
text	The string to convert.

Return value

The **Int64** value of the specified string.

Example

```
static void str2Int64Example(Args _args)
{
    str myStr;
    str tooBig;
    Int64 myInt64;
    myStr = "1234567890";
    tooBig = int642str(int64Max()+1);
    myInt64 = str2Int64(mystr);
    print strfmt ("int64: %1",myInt64);
    myInt64 = str2Int64(tooBig);
    print strfmt ("Too big int64: %1",myInt64);
}
```

str2Num

Converts a string to a real number.

```
real str2Num(str _text)
```

Parameters

PARAMETER	DESCRIPTION
_text	The string to convert to a real number.

Return value

The real number if the specified string contains a valid number; otherwise, **0** (zero).

Remarks

The following examples show how this function is used.

```
str2Num("123.45") returns the value 123.45.
str2Num("a123") returns the value 0.0.
str2Num("123a") returns the value 123.00.
```

Scanning occurs from left to right and ends when a character can't be converted to part of a real number.

Example

```
static void str2NumToReal(Args _arg)
{
    real r;
    str s;
    r = str2Num("3.15");
    s = strFmt("r = %1", r);
    info(s);
}
/*** Infolog output.
Message_@SYS14327 (02:36:12 pm)
r = 3.15
***/

static void str2NumExponentialSyntax(Args _args)
{
    Qty qty1, qty2, qty3;
    qty1 = str2num('1e-3'); // Bad syntax by the user.
    qty2 = str2num('1.e-3');
    qty3 = str2num('1.0e-3');
    info(strfmt('Result: %1; Expected: %2', num2str(qty1, 0,3,2,0), '0.001'));
    info(strfmt('Result: %1; Expected: %2', num2str(qty2, 0,3,2,0), '0.001'));
    info(strfmt('Result: %1; Expected: %2', num2str(qty3, 0,3,2,0), '0.001'));
}
/*** Infolog output. The first result differs from expectations.
Message_@SYS14327 (02:20:55 pm)
Result: 1,000; Expected: 0.001
Result: 0,001; Expected: 0.001
Result: 0,001; Expected: 0.001
***/
```

str2Time

Converts a string to a **timeOfDay** value.

```
int str2Time(str _text)
```

Parameters

PARAMETER	DESCRIPTION
_text	The time to use to calculate the number of seconds since midnight.

Return value

The number of seconds between midnight and the *_text* parameter; otherwise, **-1**.

Remarks

```
str2Time("05:01:37") //Returns the value 18097.
str2Time("7 o'clock") //Returns the value -1.
```

Example

```
static void str2TimeExample(Args _arg)
{
    int i;
    i = str2Time("11:30");
    print i;
}
```

time2Str

Converts a **timeOfDay** value to a string that includes hours, minutes, and seconds.

```
str time2Str(int _time, int _separator, int _timeFormat)
```

Parameters

PARAMETER	DESCRIPTION
_time	A **timeOfDay** value.
_separator	A **TimeSeparator** enumeration value that indicates the characters between the hours, minutes, and seconds in the output string.
_timeFormat	A **TimeFormat** enumeration value that indicates whether a 12-hour clock or a 24-hour clock is used.

Return value

A string that represents the specified time.

Remarks

The value of the *_time* parameter is the number of seconds since midnight.

Example

```
static void TimeJob4(Args _args)
{
    timeOfDay theTime = timeNow();
    info( time2Str(theTime, TimeSeparator::Colon, TimeFormat::AMPM) );
}
/**
Message (04:33:56 pm)
04:33:56 pm
**/
```

uint2Str

Converts an integer to a string. The assumption is that the integer is unsigned.

```
str uint2Str(int integer)
```

Parameters

PARAMETER	DESCRIPTION
integer	The integer to convert.

Return value

The string equivalent to the specified unsigned integer.

Remarks

Use this function instead of the **int2str** function for very large integers, such as record IDs.

```
info(int2str(3123456789)); //returns -1171510507 as a string.
info(uint2str(3123456789)); //returns 3123456789 as a string.
```

X++ date runtime functions

2/18/2021 • 8 minutes to read • Edit Online

This topic describes the date run-time functions.

dayName

Retrieves the name of the day of the week that is specified by a number.

```
str dayName(int number)
```

Parameters

PARAMETER	DESCRIPTION
number	The number of a day in a week.

Return value

The day of the week specified by the number parameter.

Remarks

The valid values for the number parameter are **1** through **7**. Monday is represented by **1**, Tuesday by **2**, and Sunday by **7**.

Example

```
static void dayNameExample(Args _arg)
{
    str s;
    ;
    s = dayName(01);
    print "First day of the week's name is " + s;
    pause;
}
```

dayOfMth

Calculates the number of the day in the month for the specified date.

```
int dayOfMth(date date)
```

Parameters

PARAMETER	DESCRIPTION
date	The date to test.

Return value

An integer between 1 and 31 that indicates the day of the month for the specified date.

Remarks

```
dayOfMth(31122001) //returns 31.
```

Example

```
static void dayOfMthExample(Args _arg)
{
    date d = today();
    int i;
    ;
    i = dayOfMth(d);
    print "Today's day of the month is " + int2Str(i);
    pause;
}
```

dayOfWk

Calculates the number of day in the week for the specified date. **Note:** Monday is represented by **1**, Tuesday by **2**, and Sunday by **7**.

```
int dayOfWk(date date)
```

Parameters

PARAMETER	DESCRIPTION
date	A **date** value that indicates the year, month, and day.

Return value

The number of the specified day in the week.

Example

```
static void dayOfWkExample(Args _arg)
{
    date d = today();
    int i;
    ;
    i = dayOfWk(d);
    print "Today's day of the week is " + int2Str(i);
    pause;
}
```

dayOfYr

Calculates the number of days between January 1 and the specified date.

```
int dayOfYr(date _date)
```

Parameters

PARAMETER	DESCRIPTION
_date	A date that specifies the year, month, and day.

Return value

The number of days between January 1 and the specified date, inclusive.

Remarks

January 1 is **1**, and December 31 is either **365** or **366**.

Example

```
static void dayOfYrExample(Args _arg)
{
    date d = today();
    int i;
    ;
    i = dayOfYr(d);
    print "Today's day of the year is " + int2Str(i);
    pause;
}
```

endMth

Calculates the last date in the month of the specified date.

```
date endMth(date date)
```

Parameters

PARAMETER	DESCRIPTION
date	A **date** value that indicates a year, month, and day.

Return value

The **date** value of the last day in the specified month.

Remarks

```
endMth(0221988); //Returns the date 2921988 because 1988 is a leap year.
endMth(0221989); //Returns the date 2821989.
```

mkDate

Creates a date, based on three integers that indicate the day, month, and year, respectively.

```
date mkDate(int day, int month, int year)
```

Parameters

PARAMETER	DESCRIPTION
day	An integer that represents the day of the month.

PARAMETER	DESCRIPTION
month	An integer that represents the month of the year.
year	An integer that represents the year, which must be between 1900 and 2154.

Return value

A **date** value that is based on the values of the *day*, *month*, and *year* parameters.

Remarks

If the date isn't valid, this method returns a **0** (zero, 1/1/1900) date. Beginning with Dynamics AX 7.0(February 2016), shortcut values for the year, e.g. 75 for 1975, are not supported. If you provide a shortcut value for the year, a date of 1/1/1900 is returned.

Example

```
static void mkDateExample(Args _arg)
{
    date d;
    ;
    // Returns the date 0112005.
    d = mkDate(1, 1, 2005);
    print d;
    pause;
}
```

mthName

Retrieves the name of the specified month

```
str monthName(int number)
```

Parameters

PARAMETER	DESCRIPTION
number	The number of the month.

Return value

The name of the specified month.

Remarks

The valid values of the *number* parameter are **1** through **12**. January is represented by **1** and December by **12**.

Example

```
static void mthNameExample(Args _arg)
{
    str s;
    ;
    // MthName(6) returns the text string "June".
    s = mthName(6);
    print "Month name is " + s;
    pause;
}
```

mthOfYr

Retrieves the number of the month in the year for the specified date. **Note:** January is **1**, February is **2**, and December is **12**.

```
int mthOfYr(date date)
```

Parameters

PARAMETER	DESCRIPTION
date	A date that specifies a year, month, and day.

Return value

The number of the month in the year, for the month that is represented by the *date* parameter.

Example

```
static void mthOfYrExample(Args _arg)
{
    int i;
    ;
    i = mthOfYr(today());
    print "The number of the month in today's date is " + int2Str(i);
    pause;
}
```

nextMth

Retrieves the date in the following month that corresponds most closely to the specified date.

```
date nextMth(date date)
```

Parameters

PARAMETER	DESCRIPTION
date	The date to match in the following month.

Return value

The closest match to the specified date that is found in the next month.

Remarks

```
nextMth(2921996); //returns 29/03/1996.
nextMth(3111996); //returns 2921996, because 1996 is a leap year.
```

Example

```
static void nextMthExample(Args _arg)
{
    date d;
    ;
    d = nextMth(today());
    print "Closest date next month is "
    + date2Str(d, 2, 2, -1, 2, -1, 4);
    pause;
}
```

nextQtr

Retrieves the date in the following quarter that corresponds most closely to the specified date.

```
date nextQtr(date date)
```

Parameters

PARAMETER	DESCRIPTION
date	The date to match in the following quarter.

Return value

The closest match to specified date that is found in the next quarter.

Remarks

For example, **nextQtr(3111998)** returns **3041998**.

Example

```
static void nextQtrExample(Args _arg)
{
    date d;
    ;
    d = nextQtr(today());
    print "Closest date next quarter is "
        + date2Str(d, 2, 2, -1, 2, -1, 4);
    pause;
}
```

nextYr

Retrieves the date in the following year that corresponds most closely to the specified date.

```
date nextYr(date date)
```

Parameters

PARAMETER	DESCRIPTION
date	The date to match in the following year.

Return value

The closest match to the specified date that is found in the following year.

Remarks

For example, **nextyr(2921998)** returns **2821999**.

Example

```
static void nextYrExample(Args _arg)
{
    date d;
    ;
    d = nextYr(today());
    print "Closest date next year is "
        + date2Str(d, 2, 2, -1, 2, -1, 4);
    pause;
}
```

prevMth

Retrieves the date in the previous month that corresponds most closely to the specified date.

```
date prevMth(date date)
```

Parameters

PARAMETER	DESCRIPTION
date	The date to match in the previous month.

Return value

The closest match to the specified date that is found in the previous month.

Remarks

```
prevMth(3131996); //Returns the date 29/02/1996 because 1996 is a leap year.
prevMth(2821998); //Returns the date 28/01/1998.
```

prevQtr

Retrieves the date in the previous quarter that corresponds most closely to the specified date.

```
date prevQtr(date date)
```

Parameters

PARAMETER	DESCRIPTION
date	The date to match in the previous quarter.

Return value

The closest match to the specified date that is found in the previous quarter.

Remarks

```
prevQtr(3041998); //Returns the date 30/01/1998.
prevQtr(2951996); //Returns the date 29/02/1996, because 1996 is a leap year.
```

prevYr

Retrieves the date in the previous year that corresponds most closely to the specified date.

```
date prevYr(date date)
```

Parameters

PARAMETER	DESCRIPTION
date	The date to match in the previous year.

Return value

The closest match to the specified date that is found in the previous year.

Remarks

```
prevYr(2921996); //Returns the date 28/02/1995 because 1996 is a leap year.
prevYr(2821998); //Returns the date 28/02/1997.
```

systemDateGet

Retrieves the session date, if it has been set.

```
date systemDateGet()
```

Return value

The session date if it has been set; otherwise, the system date.

Remarks

Consider using **Session date and time** on the **Tools** menu to open the **Session date and time** page. This page can be used to actively set the session date. After this set action is detected by the system, subsequent calls to the **systemDateGet** function return the session date. The **today** function returns the system date. This function doesn't support time zones.

Example

The following example shows the date in the Infolog window.

```
static void Job_systemDateGet(Args _arg)
{
    info( date2Str(
        systemDateGet(),        // X++ language function.
        321,                    // 321 = ymd
        DateDay::Digits2,
        DateSeparator::Hyphen,  // separator1
        DateMonth::Digits2,
        DateSeparator::Hyphen,  // separator2
        DateYear::Digits4
    )
);
/********** Actual Infolog output
Message (03:46:00 pm)
2012-04-16
**********/
}
```

systemDateSet

Changes the system date.

```
date systemDateSet(date _date)
```

Parameters

PARAMETER	DESCRIPTION
_date	The new date for the system.

Return value

The new system date.

Remarks

This function doesn't affect the session date. This method changes the date, but the time will be set to **0** (zero).

Example

The following example sets the system date to today's date.

```
static void systemDateSetExample(Args _arg)
{
    date d = today();
    d = systemDateSet(d);
    print d;
}
```

timeNow

Retrieves the current system time.

```
int timeNow()
```

Return value

The number of seconds that have passed since midnight.

Example

```
static void timeNowExample(Args _arg)
{
    int i;
    ;
    i = timeNow();
    print "The number of seconds since midnight is " + int2Str(i);
    pause;
}
```

today

Retrieves the current date on the system.

```
date today()
```

Return value

The current date.

Example

```
static void todayExample(Args _arg)
{
    date d;
    ;
    d = today();
    print "Today's date is " + date2Str(d, 0, 2, -1, 2, -1, 4);
    pause;
}
```

wkOfYr

Calculates the week of the year that a date falls in, according to the ISO 8601 specification.

```
int wkOfYr(date _date)
```

Parameters

PARAMETER	DESCRIPTION
_date	The date to calculate the week of the year for.

Return value

The sequence number of the week that the *_date* parameter occurs in.

Example

The following code example compares the **wkOfYr** function with the **Global::weekOfYear** method. The function and the method produce different results.

```
// X++ job, under AOT > Jobs.
static void WeekTests3Job(Args _args)
{
int weekNum, i;
date dateTest;
str sMessages[];
//-------------------------------------------
sMessages[1] = "----- #1.  For Sunday, January 5, 2003 -----";
dateTest = 512003; // DayMonthYear  format.
weekNum = wkOfYr(dateTest);
sMessages[2] = int2str(weekNum) + " = wkOfYr funtion";
weekNum = Global::weekOfYear(dateTest);
sMessages[3] = int2str(weekNum) + " = Global::weekOfYear method";
//-------------------------------------------
sMessages[4] = " ";
sMessages[5] = "----- #2.  For Wednesday, August 20, 2003 -----";
dateTest = 2082003;
weekNum = wkOfYr(dateTest);
sMessages[6] = int2str(weekNum) + " = wkOfYr funtion";
weekNum = Global::weekOfYear(dateTest);
sMessages[7] = int2str(weekNum) + " = Global::weekOfYear method";
//-------------------------------------------
sMessages[8] = " ";
sMessages[9] = "----- #3.  For Sunday, December 28, 2003 -----";
dateTest = 28122003;
weekNum = wkOfYr(dateTest);
sMessages[10] = int2str(weekNum) + " = wkOfYr funtion";
weekNum = Global::weekOfYear(dateTest);
sMessages[11] = int2str(weekNum) + " = Global::weekOfYear method";
for (i=1; i<= 11; i++)
{
Global::info(sMessages[i]);
}
}
```

The previous example sent the following information to the Infolog for display. The output shows that there are differences between **wkOfYr** and **Global::weekOfYear**.

```
Message (01:59:13 pm) -----
#1. For Sunday, January 5, 2003 ----- 1 = wkOfYr function 2 = Global::weekOfYear method -----
#2. For Wednesday, August 20, 2003 ----- 34 = wkOfYr function 34 = Global::weekOfYear method -----
#3. For Sunday, December 28, 2003 ----- 52 = wkOfYr function 1 = Global::weekOfYear method
```

year

Retrieves the year from a **date** value.

```
int year(date _date)
```

Parameters

PARAMETER	DESCRIPTION
_date	The date to return the year from.

Return value

The year of the specified date.

Remarks

```
year(0221998); //Returns the value 1998.
```

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

X++ math runtime functions

2/18/2021 • 8 minutes to read • Edit Online

This topic describes the math run-time functions.

These functions perform mathematical calculations.

abs

Retrieves the absolute value of a real number. Examples:

- **abs(-100.0)** returns the value **100.0**.
- **abs(30.56)** returns the value **30.56**.

Syntax

```
real abs(real arg)
```

Parameters

PARAMETER	DESCRIPTION
arg	The number to get the absolute value of.

Return value

The absolute value of *arg*.

Example

```
static void absExample(Args _args)
{
    real r1;
    real r2;
    ;
    r1 = abs(-3.14);
    r2 = abs(3.14);
    if (r1 == r2)
    {
        print "abs of values are the same";
        pause;
    }
}
```

acos

Retrieves the arc cosine of a real number.

> **NOTE**
> Argument values that are outside the -1 to 1 range cause the following run-time error: "Argument for trigonometric function out of range."

Syntax

```
real acos(real arg)
```

Parameters

PARAMETER	DESCRIPTION
arg	The number to retrieve the arc cosine of.

Return value

The arc cosine of *arg*.

Example

```
static void acosExample(Args _args)
{
    real r;
    str  s;
    ;
    r = acos(0.0);
    s = strFmt("The arc cosine of 0.0 is %1 ", r);
    print s;
    pause;
}
```

asin

Retrieves the arc sine of a real number.

> **NOTE**
> Argument values that are outside the -1 to 1 range cause the following run-time error: "Argument for trigonometric function out of range."

Syntax

```
real asin(real arg)
```

Parameters

PARAMETER	DESCRIPTION
arg	The number to calculate the arc sine for.

Return value

The arc sine of the specified number.

Remarks

aSin(0.36) returns **0.37**.

atan

Retrieves the arc tangent of a real number.

Syntax

```
real atan(real arg)
```

Parameters

PARAMETER	DESCRIPTION
arg	The number to calculate the arc tangent for.

Return value

The arc tangent of the specified number.

Remarks

aTan(0.36) returns **0.35**.

Example

```
static void atanExample(Args _args)
{
    real r;
    ;
    r = atan(1.0);
    print strFmt("The Arc Tangent of 1.0 is %1", r);
    pause;
}
```

corrFlagGet

Retrieves the state of the correction flag for a real number.

Syntax

```
int corrFlagGet(real arg)
```

Parameters

PARAMETER	DESCRIPTION
arg	The flag to retrieve the state for.

Return value

A non-zero value if the flag is set; **0** (zero) if the flag is cleared.

Example

The following example displays **1**.

```
static void corrFlagGetExample(Args _args)
{
    real rr;
    rr = corrFlagSet(0.36,2);
    print(corrFlagGet(rr));
}
```

corrFlagSet

Controls the correction flag for a real number.

Syntax

```
real corrFlagSet(real real, int arg)
```

Parameters

PARAMETER	DESCRIPTION
real	The number in which to turn the correction flag on or off.
arg	**0** to turn the flag off; a non-zero value to turn the flag on.

Return value

0 if the flag is now off; a non-zero value if the flag is now on.

COS

Retrieves the cosine of a real number.

Syntax

```
real cos(real arg)
```

Parameters

PARAMETER	DESCRIPTION
arg	The number to find the cosine for.

Return value

The cosine of the specified number.

Remarks

The value of the *arg* parameter must be in radians.

Example

The following code example displays **0.76**.

```
static void cosExample(Args _arg)
{
    real r;
    ;
    r = cos(15);
    print strFmt("Cos of 15 is %1", r);
    pause;
}
```

cosh

Retrieves the hyperbolic cosine of a real number.

> **NOTE**
>
> Argument values that are outside the -250 to 250 range cause the following run-time error: "Argument for trigonometric function out of range."

Syntax

```
real cosh(real arg)
```

Parameters

PARAMETER	DESCRIPTION
arg	The hyperbolic number to calculate the cosine for.

Return value

The hyperbolic cosine of the specified number.

Remarks

The value of the *arg* parameter must be in radians.

Example

```
static void coshExample(Args _arg)
{
    real r;
    ;
    r = cosh(0.1);
    print "The hyperbolic cosine of 0.1 is " + num2Str(r, 2, 2, 1, 1);
    pause;
}
```

decRound

Rounds a number to the specified number of decimal places.

Syntax

```
real decRound(real figure, int decimals)
```

Parameters

PARAMETER	DESCRIPTION
figure	The number to round.
decimals	The number of decimal places to round to.

Return value

The value of the specified number, rounded to the specified number of decimal places.

Remarks

The value of the *decimals* parameter can be positive, 0 (zero), or negative.

- **decRound(1234.6574,2)** returns the value **1234.66**.
- **decRound(1234.6574,0)** returns the value **1235**.
- **decRound(1234.6574,-2)** returns the value **1200**.
- **decRound(12345.6789,1)** returns the value **12345.70**.
- **decRound(12345.6789,-1)** returns the value **12350.00**.

exp

Retrieves the natural antilogarithm of the specified real number.

Syntax

```
real exp(real arg)
```

Parameters

PARAMETER	DESCRIPTION
arg	The real number to calculate the natural antilogarithm for.

Return value

The natural antilogarithm of the specified real number.

Remarks

The calculated natural antilogarithm is the natural logarithm e raised to the power that is indicated by the *arg* parameter.

Example

```
static void expExample(Args _arg)
{
    real r1;
    real r2;
    ;
    r1 = exp(2.302585093);
    r2 = exp10(2.302585093);
    print strFmt("exp of 2.302585093 is %1", r1);
    print strFmt("exp10 of 230258 is %1", r2);
    pause;
}
```

exp10

Retrieves the base-10 antilogarithm of the specified real number.

Syntax

```
real exp10(real decimal)
```

Parameters

PARAMETER	DESCRIPTION
decimal	The real number to calculate the base-10 antilogarithm for.

Return value

The 10-based antilogarithm of the value of the *decimal* parameter.

Example

```
static void exp10Example(Args _arg)
{
    real r1;
    real r2;
    ;
    r1 = exp(2.302585093);
    r2 = exp10(2.302585093);
    print strFmt("exp of 2.302585093 is %1", r1);
    print strFmt("exp10 of 230258 is %1", r2);
    pause;
}
```

frac

Retrieves the decimal part of a real number.

Syntax

```
real frac(real decimal)
```

Parameters

PARAMETER	DESCRIPTION
decimal	The real number to retrieve the decimal part for.

Return value

The decimal part of the specified number.

Remarks

`frac(12.345)` returns the value `0.345`.

log10

Retrieves the 10-digit logarithm of a real number.

Syntax

```
real log10(real arg)
```

Parameters

PARAMETER	DESCRIPTION
arg	The number to calculate the logarithm for.

Return value

The base-10 logarithm of the specified number.

Remarks

`log10(200)` returns the value `2.30`.

logN

Retrieves the natural logarithm of the specified real number.

Syntax

```
real logN(real arg)
```

Parameters

PARAMETER	DESCRIPTION
arg	The number to calculate the natural logarithm for.

Return value

The natural logarithm of the specified number.

Remarks

`logN(45)` returns the value `3.81`.

max

Retrieves the larger of two specified values.

```
anytype max(anytype object1, anytype object2)
```

Parameters

PARAMETER	DESCRIPTION
object1	The first value.
object2	The second value.

Return value

The larger of the two values that are specified by the *object1* and *object2* parameters.

Remarks

- `max(12.0,12.1)` returns the value `12.1`.
- `max(2,33)` returns the value `33`.

min

Retrieves the smaller of two specified values.

```
anytype min(anytype object1, anytype object2)
```

Parameters

PARAMETER	DESCRIPTION
object1	The first value.
object2	The second value.

Return value

The smaller of the two values that are specified by the *object1* and *object2* parameters.

Remarks

`min(2,33)` returns the value `2`.

Example

```
static void minExample(Args _arg)
{
    anytype a;
    real r = 3.0;
    real s = 2.0;

    a = min(r, s);
    print num2Str(a, 1, 2, 1, 1) + " is less than the other number.";
}
```

power

Raises a real number to the power of another real number.

Syntax

```
real power(real arg, real exponent)
```

Parameters

PARAMETER	DESCRIPTION
arg	The number to calculate the power of.
exponent	The number to raise the number that is specified by the *arg* parameter to.

Return value

The real number that is the number specified by the *arg* parameter to the power of the number specified by the *exponent* parameter.

Remarks

- `power(5.0,2.0)` returns the value `25.0`.
- `power(4.0,0.5)` returns the value `2.0`.

round

Rounds a real number to the nearest multiple of another real number.

Syntax

```
real round(real _arg, real _decimals)
```

Parameters

PARAMETER	DESCRIPTION
_arg	The original number.
_decimals	The number that the value of the *_arg* parameter must be rounded to a multiple of.

Return value

The number that is a multiple of the value specified by the *_decimals* parameter and is closest to the value specified by the *_arg* parameter.

Remarks

To round a real number to a specified number of decimal places, use the decround function.

Remarks

- **round(123.45,5.00)** returns the value **125.00**.
- **round(7.45,1.05)** returns the value **7.35**.
- **round(23.9,5.0)** returns the value **25.00**.
- **round(26.1,5.0)** returns the value **25.00**.

sin

Retrieves the sine of a real number.

Syntax

```
real sin(real _arg)
```

Parameters

PARAMETER	DESCRIPTION
_arg	The number to calculate the sine for.

Return value

The sine of the specified real number.

Remarks

The value of the *_arg* parameter must be in radians.

Example

```
static void sinExample(Args _arg)
{
    real angleDegrees = 15.0;
    real angleRadians;
    real pi = 3.14;
    real r;
    ;
    angleRadians = pi * angleDegrees / 180;
    r = sin(angleRadians);
    print "sin of a "
        + num2Str(angleDegrees, 2, 2, 1, 1)
        + " degree angle is "
        + num2Str(r, 2, 10, 1, 1);
    pause;
}
```

sinh

Retrieves the hyperbolic sine of a real number.

Syntax

```
real sinh(real _arg)
```

Parameters

PARAMETER	DESCRIPTION
_arg	The number to calculate the hyperbolic sine for.

Return value

The hyperbolic sine of the specified real number.

Remarks

Values for the *_arg* parameter that are outside the -250 to 250 range cause the following run-time error: "Argument for trigonometric function out of range."

Example

The following example illustrates the **sinh** function.

```
static void sinhExample(Args _arg)
{
    real angleDegrees = 45.0;
    real angleRadians;
    real pi = 3.14;
    real r;
    ;
    angleRadians = pi * angleDegrees / 180;
    r = sinh(angleRadians);
    print "sinh of a "
    + num2Str(angleDegrees, 2, 2, 1, 1)
    + " degree angle is "
    + num2Str(r, 2, 15, 1, 1);
    pause;
}
```

tan

Retrieves the tangent of a real number.

Syntax

```
real tan(real arg)
```

Parameters

PARAMETER	DESCRIPTION
arg	The real number to calculate the tangent for.

Return value

The tangent of the specified real number.

Remarks

Values for the *arg* parameter that are outside the -250 to 250 range cause the following run-time error: "Argument for trigonometric function out of range."

Example

The following example illustrates the **tan** function.

```
static void tanExample(Args _arg)
{
    real r;
    ;
    r = tan(250);
    print strFmt("Tan of 250 is %1", r);
    pause;
}
```

tanh

Retrieves the hyperbolic tangent of a real number.

Syntax

```
real tanh(real _arg)
```

Parameters

PARAMETER	DESCRIPTION
_arg	The number to calculate the hyperbolic tangent for.

Return value

The hyperbolic tangent of the specified real number.

Example

The following example illustrates the **tanh** function.

```
static void tanhExample(Args _arg)
{
    real r;
    ;
    r = tanh(0.1);
    print "The hyperbolic tangent of angle 0.1 is "
    + num2Str(r, 2, 10, 1, 1);
    pause;
}
```

trunc

Truncates a real number by removing any decimal places.

Syntax

```
real trunc(real _decimal)
```

Parameters

PARAMETER	DESCRIPTION
_decimal	The number to truncate.

Return value

A number that is equivalent to the value of the *_decimal* parameter after the decimal places have been removed.

Remarks

This function always rounds numbers down to a complete integer.

Example

The following example truncates 2.7147 to 2.00.

```
static void truncExample(Args _arg)
{
    real r;
    ;
    r = trunc(2.7147);
    print strFmt("r = %1",  r);
    pause;
}
```

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

X++ reflection runtime functions

2/18/2021 • 7 minutes to read • Edit Online

This topic describes the reflection run-time functions.

classIdGet

Retrieves the numeric identifier (the class ID) of the class that the object that is initialized belongs to.

```
int classIdGet(class object)
```

Parameters

PARAMETER	DESCRIPTION
object	The object to get the class ID for.

Return value

The class ID of the specified object.

Example

```
static void classIdGetExample(Args _args)
{
    int i;
    WorkTimeCheck w;

    i = classIdGet(w);
    print "Class ID for object is " + int2Str(i);
}
```

dimOf

Retrieves the number of index elements that space has been allocated for in an X++ array.

```
int dimOf(anytype object)
```

Parameters

PARAMETER	DESCRIPTION
object	The array to determine the dimension size of.

Return value

If the value of the *object* parameter is an array, the number of elements in the array; otherwise, **0** (zero).

Remarks

The **dimOf** function is intended for X++ arrays that are declared as the following X++ primitive types:

- boolean

- date
- int
- int64
- real
- utcDateTime

An example is **int iAmounts[6];**. Arrays of enumeration values and extended data types are also supported if they are ultimately based on one of the preceding primitive data types (such as **int**). The **dimOf** function doesn't accept arrays of all X++ primitive types. Here are the array types that the **dimOf** function doesn't accept:

- **str**
- **container**
- **anytype**
- Arrays of class objects
- Instances of the **Array** class

Example

```
static void JobDimOfArrays(Args _args)
{
    int iAmounts[20], iCounts[];
    ABCModel enumAbcModel[22]; // Enum
    ABCModelType exdtAbcModelType[24]; // Extended data type
    anytype anyThings[26];
    str sNames[28];
    Array myArrayObj; // Class

    info("Start of job.");
    info("--(Next, normal int array, dimOf() accepts it.)");
    info(int2Str(dimOf(iAmounts)));
    info("--(Next, normal enum array, dimOf() accepts it.)");
    info(int2Str(dimOf(enumAbcModel)));
    info("--(Next, normal extended data type array (based on enum), dimOf() accepts it.)");
    info(int2Str(dimOf(exdtAbcModelType)));
    info("--(Next, dynamic int array, dimension not yet set.)");
    info(int2Str(dimOf(iCounts)));
    info("--(Next, dynamic int array, after dimension established.)");

    iCounts[13] = 13;
    info(int2Str(dimOf(iCounts)));
    info(" == == == == == (Next, array types that dimOf() does not support.)");
    info("--(Next, normal anytype array, dimOf() always returns 0.)");
    info(int2Str(dimOf(anyThings)));
    info("--(Next, an instance of class X++ Array, dimOf() always returns 0.)");

    myArrayObj = new Array(Types::Integer);
    myArrayObj.value(1,501);
    info(int2Str(dimOf(myArrayObj)));
    info("--(Next, the lastIndex method provides size information about Array instances.)");
    info(int2Str(myArrayObj.lastIndex()));
    info("--(Next, normal str array, dimOf() does not accept it, job is halted.)");
    info(int2Str(dimOf(sNames)));
    info("End of job.");

}
/************  Actual Infolog output
Message (11:10:06 am)
Start of job.
--(Next, normal int array, dimOf() accepts it.)
20
--(Next, normal enum array, dimOf() accepts it.)
22
--(Next, normal extended data type array (based on enum), dimOf() accepts it.)
24
--(Next, dynamic int array, dimension not yet set.)
0
--(Next, dynamic int array, after dimension established.)
16
== == == == == (Next, array types that dimOf() does not support.)
--(Next, normal anytype array, dimOf() always returns 0.)
0
--(Next, an instance of class X++ Array, dimOf() always returns 0.)
0
--(Next, the lastIndex method provides size information about Array instances.)
1
--(Next, normal str array, dimOf() does not accept it, job is halted.)
Error executing code: Illegal operation on this type of array. (C)JobsJobDimOfArrays - line 41
************/
/************  Pop-up error dialog box
"Internal error number 25 in script."
This error is caused by the code line...
info(int2Str(dimOf(iCounts)));
...before iCounts was assigned at any index.
***********/
```

fieldId2Name

Retrieves a string that represents the name of the field that is specified by a table ID number and a field ID number.

```
str fieldId2Name(int tableid, int fieldid)
```

Parameters

PARAMETER	DESCRIPTION
tableid	The ID number of the table. **Note:** Use the **tableName2Id** function to specify the ID of a table.
fieldid	The ID number of the field.

Return value

The name of the field.

Remarks

To return a printable version of the field name, use the **fieldId2PName** function.

Example

The following example sets **fn** to the name of the field in the Customer (CustGroup) table that has a field ID of 7.

```
static void fieldId2NameExample(Args _arg)
{
    str fn;
    fn = fieldId2Name(tableName2Id("Customer"),7);
}
```

fieldId2PName

Retrieves the printable name of the field that is specified by a table ID number and a field ID number.

```
str fieldId2PName(int tableid, int fieldid)
```

Parameters

PARAMETER	DESCRIPTION
tableid	The ID number of the table. **Note:** Use the **tableName2Id** function to specify the ID of a table.
fieldid	The ID number of the field. **Note:** Use the **fieldName2Id** function to specify the ID of a field.

Return value

The name of the field.

Example

```
static void fieldId2PNameExample(Args _arg)
{
    str name;
    tableid _tableId;
    fieldid _fieldid;

    _tableId = tableName2Id("Address");
    _fieldId = fieldName2Id(_tableId, "Name");
    name = fieldId2PName(_tableId, _fieldid);
    print name;
}
```

fieldName2Id

Retrieves the field ID of the table field that is specified by a table ID number and a field ID number.

```
int fieldName2Id(int tableid, str fieldname)
```

Parameters

PARAMETER	DESCRIPTION
tableid	The ID number of the table. **Note:** Use the **tableName2Id** function to specify the ID of a table.
fieldname	The name of the field.

Return value

The ID of the field that is specified by the *tableid* and *fieldname* parameters.

Example

```
static void fieldName2IdExample(Args _arg)
{
    int id;

    id = fieldName2Id(tableName2Id("Address"), "Name");
    // Returns 6. Name is the 6th field in the Address table.
    print id;
}
```

indexId2Name

Retrieves the name of an index.

```
str indexId2Name(int tableid, int indexid)
```

Parameters

PARAMETER	DESCRIPTION
tableid	The ID of the table that the index belongs to.
indexid	The ID of the index.

Return value

The name of the index.

Example

```
static void indexId2NameExample(Args _arg)
{
    str s;
    tableid id;
    indexid idx;

    id  = tableName2Id("Address");
    idx = indexName2Id(id, "AddrIdx");
    s = indexId2Name(id, idx);
    print "The result of calling indexId2Name is " + s;
}
```

indexName2Id

Retrieves the ID of an index.

```
int indexName2Id(int tableid, str indexname)
```

Parameters

PARAMETER	DESCRIPTION
tableid	The ID of the table that the index belongs to.
indexname	The name of the index.

Return value

The ID of the index.

Example

```
static void indexName2IdExample(Args _arg)
{
    indexid idx;
    tableid id;

    id  = tableName2Id("Address");
    idx = indexName2Id(id, "AddrIdx");
    print "Index ID for index name AddrIdx of table Address is " + int2Str(idx);
}
```

tableId2Name

Retrieves a string that contains the name of a table.

```
str tableId2Name(int _tableid)
```

Parameters

PARAMETER	DESCRIPTION
_tableid	The ID of the table.

Return value

The name of the table.

Example

```
static void tableId2NameExample(Args _arg)
{
    str s;
    tableid id;

    // Get the ID for table name Address.
    id = tableName2Id("Address");
    print "ID for table name Address is " + int2Str(id);

    // Get the name from the table ID.
    s = tableId2Name(id);
    print "Name for table ID " + int2Str(id) + " is " + s;

    // Get the printable name from the table ID.
    s = tableId2PName(id);
    print "Printable name for table ID " + int2Str(id) + " is " + s;
}
```

tableId2PName

Retrieves a string that contains the printable name (the label) of a table.

```
str tableId2PName(int _fieldid)
```

Parameters

PARAMETER	DESCRIPTION
_fieldid	The ID of the table.

Return value

The label of the table.

Example

```
static void tableId2NameExample(Args _arg)
{
    str s;
    tableid id;

    // Get the ID for table name Address.
    id = tableName2Id("Address");
    print "ID for table name Address is " + int2Str(id);

    // Get the name from the table ID.
    s = tableId2Name(id);
    print "Name for table ID " + int2Str(id) + " is " + s;

    // Get the printable name from the table ID.
    s = tableId2PName(id);
    print "Printable name for table ID " + int2Str(id) + " is " + s;
}
```

tableName2Id

Retrieves the ID of a table.

```
int tableName2Id(str _name)
```

Parameters

PARAMETER	DESCRIPTION
_name	The name of the table.

Return value

The ID of the table.

Example

```
static void tableName2IdExample(Args _arg)
{
    str s;
    tableid id;

    // Get the ID for the Address table name.
    id = tableName2Id("Address");
    print "ID for the Address table name is " + int2Str(id);

    // Get the name from the table ID.
    s = tableId2Name(id);
    print "Name for table ID " + int2Str(id) + " is " + s;

    // Get the printable name from the table ID.
    s = tableId2PName(id);
    print "Printable name for table ID " + int2Str(id) + " is " + s;
}
```

typeOf

Retrieves the type of an element.

```
enum typeOf(anytype _object)
```

Parameters

PARAMETER	DESCRIPTION
_object	The element to return the type for.

Return value

A **Types** system enumeration value.

Example

The following example tests whether the first element in a container, **c**, is another container that contains a single integer.

```
if(typeof(conpeek(c, 1)) != Types::Container ||
conlen(conpeek(c, 1)) != 1 ||
typeof(conpeek(conpeek(c, 1), 1)) != Types::Integer)
{
    // More code.
}
```

X++ session runtime functions

2/18/2021 • 5 minutes to read • Edit Online

This topic describes the session run-time functions.

curExt

Retrieves the extension that is used for the current company.

```
str curExt()
```

Return value

The extension for the current company.

Example

```
static void curExtExample(Args _arg)
{
    str s;
    // Sets s to the extension of the current company.
    s = curExt();
    print "Current extension is " + s;
}
```

curUserId

Retrieves the nonnumeric ID that represents the current user.

```
str curUserId()
```

Return value

The nonnumeric ID that represents the current user.

Example

```
static void curUserIdExample(Args _arg)
{
    str s;
    s = curUserId();
    print "Current user ID is " + s;
}
```

funcName

Retrieves a string that contains the current function context.

```
str funcName()
```

Return value

The name of the method that is executing this method.

Remarks

If execution is currently within the member of a table or class, the name of the method is prefixed with the name of that table or class.

Example

```
static void funcNameExample(Args _arg)
{
    print "Current function context is " + funcName();
}
```

getCurrentPartition

Retrieves the short name of the current partition.

```
str getCurrentPartition()
```

Return value

The short name of the current partition.

Remarks

The maximum length of the data partition name that is returned is eight characters.

Example

The following code example shows calls to, and output from, the **getCurrentPartition** function of the X++ language, and related functions or methods.

```
static public void Main(Args _args)  // X++ method.
{
    int64 iPartition;
    str sPartition;
    SelectableDataArea oSelectableDataArea;  // System ExDT.
    iPartition = getCurrentPartitionRecId();
    sPartition = getcurrentpartition();
    oSelectableDataArea = Global::getCompany( tableNum(BankAccountTable) );
    Global::info( strFmt(
            "getCurrentPartitionRecId =%1 , getCurrentPartition =%2 , getCompany =%3",
            iPartition, sPartition, oSelectableDataArea) );
}
/**** Pasted from Infolog window:
Message_@SYS14327 (03:42:38 pm)
getCurrentPartitionRecId =5637144576 , getCurrentPartition =initial , getCompany =ceu
****/
```

getCurrentPartitionRecId

Retrieves the **RecId** field of the current partition.

```
int64 getCurrentPartitionRecId()
```

Return value

The **RecId** field of the current data partition.

Remarks

To see a code example that relies on the **getCurrentPartitionRecId** function, see How to: Include a Filter for Partition in Direct Transact-SQL.

Example

The following code example shows calls to, and output from, the **getCurrentPartitionRecId** function of the X++ language, and related functions or methods.

```
static public void Main(Args _args)  // X++ method.
{
    int64 iPartition;
    str sPartition;
    SelectableDataArea oSelectableDataArea;  // System ExDT.
    iPartition = getCurrentPartitionRecId();
    sPartition = getcurrentpartition();
    oSelectableDataArea = Global::getCompany( tableNum(BankAccountTable) );
    Global::info( strFmt(
            "getCurrentPartitionRecId =%1 , getCurrentPartition =%2 , getCompany =%3",
            iPartition, sPartition, oSelectableDataArea) );
}
/**** Pasted from Infolog window:
Message_@SYS14327 (03:42:38 pm)
getCurrentPartitionRecId =5637144576 , getCurrentPartition =initial , getCompany =ceu
****/
```

getPrefix

Retrieves the current execution prefix after successive calls to the **setPrefix** function.

```
str getPrefix()
```

Return value

The current execution prefix.

Remarks

The prefix mechanism makes it more straightforward to write precise error messages about the transactions that an application performs. Because a hierarchical display is created in the Infolog, it can be easier to determine where each error came from.

Example

```
static void getPrefixExample(Args _arg)
{
    setPrefix("Prefix");
    setPrefix("Another prefix");
    print getPrefix();
}
```

sessionId

Retrieves the session number of the current session.

```
int sessionId()
```

Return value

The numeric ID of the current session.

Remarks

A session number is assigned when the client is started and connects to Application Object Server (AOS). Every call of this function during the life of the client returns the same integer value. The returned value is compatible with the **SessionID** extended data type. The **contains** methods return information about individual user sessions.

Example

```
static void sessionIdExample(Args _arg)
{
    int session;
    session = sessionId();
    print "This session ID is number " + int2Str(session);
}
```

prmIsDefault

Determines whether the specified parameter for the current method has the default value.

```
int prmIsDefault(anytype argument)
```

Parameters

PARAMETER	DESCRIPTION
Argument	The parameter to test.

Return value

1 if the default value for the parameter was used; otherwise, **0** (zero).

Example

```
static void prmIsDefaultExample(Args _arg)
{
    void fn(boolean b = true, int j = 42)
    {
        if (prmIsDefault(b) == 1)
        {
            print "First parameter is using the default value.";
        }
        else
        {
            print "First parameter is not using the default value.";
        }
    }
    fn();
    fn(false);
}
```

runAs

Enables the caller to run an X++ method in the security context of another user. This function is most often used with batch processing.

```
container runAs(
    str userId,
    int classId,
    str staticMethodName
    [,
    container params,
    str company,
    str language,
    str partition
    ])
```

Parameters

PARAMETER	DESCRIPTION
userId	The user to impersonate.
classId	The class to invoke in the impersonated session.
staticMethodName	The class method to invoke in the new user context.
params	The parameters to pass to the method; optional.
company	The company that is selected for the impersonated session; optional.
language	The language that is selected for the impersonated session; optional.
partition	The partition key of the type that is returned by the **getCurrentPartition** function; optional.

Return value

A container that holds the return value or values of the method that is called by the **runAs** function, if any values were returned.

Remarks

This function makes it possible to run code as another user. This capability presents a security threat. Therefore, this function runs under Code Access Security. Calls to this function on the server require permission from the **RunAsPermission** class. Each use of this application programming interface (API) should be threat-modeled. If a security vulnerability is discovered, validate input to this API. The debugger might ignore breakpoints that are located in a method that is called by using the **runAs** function. X++ code that is executed by the **runAs** function must run as Microsoft .NET Framework Common Intermediate Language (CIL). If CIL hasn't been generated for the target static method, an error message indicates that the method isn't found. The **PartitionKey** system type is the exact type of the *partition* parameter. **PartitionKey** is a string that has a maximum length of eight characters.

Example

The following example calls the **runDueDateEventsForUser** method in the **EventJobDueDate** class. The code runs in the security context of a user. Run this code by applying it to a method in a new class.

```
server static public void Main(Args _args)
{
    RunAsPermission perm;
    UserId runAsUser;
    SysUserInfo userInfo;
    userInfo = SysUserInfo::find();
    runAsUser = userInfo.Id;
    perm = new RunAsPermission(runAsUser);
    perm.assert();
    runAs(runAsUser, classnum(EventJobDueDate), "runDueDateEventsForUser");
    CodeAccessPermission::revertAssert();
}
```

setPrefix

Sets the prefix for the current execution scope.

```
int setPrefix(str _prefix)
```

Parameters

PARAMETER	DESCRIPTION
_prefix	The prefix for the current execution scope.

Return value

0 if the prefix was set successfully.

Remarks

The complete prefix for the execution can be fetched by using the **getPrefix** function. When the scope is left, the prefix is automatically reset to the previous level. The prefix mechanism makes it more straightforward to write precise error messages about the transactions that an application performs. For example, the **AA** method calls the **BB** method, and each method calls the **setPrefix** function. Messages that the **BB** method writes to the Infolog appear nested in a hierarchy. When the **BB** method ends, and control returns to the **AA** method, the prefix that was set by the **BB** method isn't attached to subsequent messages.

Example

```
static void setPrefixExample(Args _arg)
{
    int i;
    i = setPrefix("Prefix");
    print i;
}
```

X++ string runtime functions

2/18/2021 • 14 minutes to read • Edit Online

This topic describes the string run-time functions.

match

Searches for a string or expression in another string.

```
int match(str pattern, str text)
```

Parameters

PARAMETER	DESCRIPTION
pattern	The string or expression to search for.
text	The string to search.

Return value

1 if the pattern is located in the string; otherwise, **0** (zero).

Remarks

The search is case-insensitive. The following special characters can be used to create the pattern for the *pattern* parameter.

- ****: A backslash (\) nullifies, or escapes, the special treatment of special characters, so that a special character can be matched like a normal letter. A pair of backslashes is translated into one non-special backslash. Examples:
  - **match("ab$cd","ab$cd");** returns **0**.
  - **match("ab\$cd","ab$cd");** returns **0**. The backslash isn't escaped.
  - **match("ab\\$cd","ab$cd");** returns **1**. The backslash and dollar sign are escaped.
- **< or ^**: A left angle bracket (<) or a circumflex (^) at the start of an expression is used to match the start of a line. Examples:
  - **match("<abc","abcdef");** returns **1**.
  - **match("<abc","defabc");** returns **0**.
  - **match("^abc","abcdef");** returns **1**.
  - **match("^abc","defabc");** returns **0**.
- **> or $**: A right angle bracket (>) or a dollar sign ($) at the end of the expression is used to match the end of a line. Examples:
  - **match("abc>","abcdef");** returns **0**.
  - **match("abc>","defabc");** returns **1**.
- **? or .**: A question mark (?) or a period (.) matches any one character in the same position. Examples:
  - **match("abc.def","abc#def");** returns **1**.
  - **match("colou?r","colouXr");** returns **1**.

- **:x**: A colon (:) specifies a group of characters to match, as indicated by the character that immediately follows.
- **:a**: Sets the match to letters. Examples:
  - **match("ab:acd","ab#cd");** returns **0**.
  - **match("ab:acd","abxyzcd");** returns **0**.
  - **match("ab:acd","abxcd");** returns **1**.
- **:d**: Sets the match to numeric characters. Examples:
  - **match("ab:dcd","ab3cd");** returns **1**.
  - **match("ab:dcd","ab123cd");** returns **0**.
  - **match("ab:dcd","abcd");** returns **0**.
- **:n**: Sets the match to alphanumeric characters. Examples:
  - **match("ab:ncd","ab%cd");** returns **0**.
  - **match("ab:ncd","ab9cd");** returns **1**.
  - **match("ab:ncd","abXcd");** returns **1**.
- **:SPACE**: SPACE is the space character (" "). Sets the match to blanks, tabulations, and control characters such as Enter (new line). Examples:
  - **match("ab: cd","ab cd");** returns **1**.
  - **match("ab: cd","ab\ncd");** returns **1**.
  - **match("ab: cd","ab\tcd");** returns **1**.
  - **match("ab: cd","ab cd");** returns **0**. Only the first space is matched.
- *****: An expression that is followed by an asterisk ("*") requires a match for zero, one, or more occurrences of the preceding expression. Examples:
  - **match("abc*d","abd");** returns **1**.
  - **match("abc*d","abcd");** returns **1**.
  - **match("abc*d","abcccd");** returns **1**.
  - **match("abc*d","abxd");** returns **0**.
- **+**: An expression that is followed by a plus sign (**+**) requires a match for one or more occurrences of the preceding expression. Examples:
  - **match("abc+d","abd");** returns **0**.
  - **match("abc+d","abcd");** returns **1**
  - **match("abc+d","abcccd");** returns **1**.
  - **match("abc+d","abxd");** returns **0**.
- **-**: An expression that is followed by a minus sign (**-**) requires a match for zero or one occurrence of the preceding expression. In other words, the preceding expression is optional. Examples:
  - **match("colou-r","color");** returns **1**.
  - **match("colou-r","colour");** returns **1**.
- **[]**: Matches a single character with any character that is enclosed in the brackets. A range of characters can be specified by two characters that are separated by a minus sign (-). For example, **[a-z]** matches all letters between a and z, **[0-9]** matches a digit, and **[0-9a-f]** matches a hexadecimal digit. Examples:
  - **match("[abc]","apple");** returns **1**, because it matches the a in "apple."
  - **match("[abc]","kiwi");** returns **0**, because "kiwi" doesn't contain an a, b, or c.
  - **match("gr[ae]y","grey");** returns 1. This expression also matches "gray."
  - **match("gr[ae]y","graey");** returns **0**, because only one character between "gr" and "y" is matched.

- **[^]**: If the first character in the text that is enclosed in brackets is a circumflex (**^**), the expression matches all characters except the characters that are enclosed in the brackets. Examples:
  - **match("[^bc]at","bat");** returns **0**.
  - **match("[^bc]at","hat");** returns **1**.
  - **match("[^abc]","bat");** returns **1**. Anything except a, b, or c is matched. Therefore, the t is matched.

strAlpha

Copies only the alphanumeric characters from a string.

```
str strAlpha(str _text)
```

Parameters

PARAMETER	DESCRIPTION
_text	The string to copy from.

Return value

A new string that contains all the alphanumeric characters from the specified string.

Remarks

For example, **strAlpha("2+2=5 is this correct?")** returns the string **225isthiscorrect**.

Example

```
static void strAlphaExample(Args _arg)
{
    str s;
    ;
    s = strAlpha("?a*bc123.");
    print s;
    pause;
}
```

strCmp

Compares two text strings.

```
int strCmp(str text1, str text2)
```

Parameters

PARAMETER	DESCRIPTION
text1	The first string.
text2	The second string.

Return value

0 if the two strings are identical, **1** if the first string sorts earlier, or **-1** if the second string sorts earlier.

Remarks

The comparison performed by this method is case-sensitive.

```
print strCmp("abc", "abc"); //Returns the value 0.
print strCmp("abc", "ABC"); //Returns the value 1.
print strCmp("aaa", "bbb"); //Returns the value -1.
print strCmp("ccc", "bbb"); //Returns the value 1.
```

strColSeq

Converts all uppercase characters to lowercase characters, and converts all characters that have accents to the corresponding unaccented lowercase characters.

```
str strColSeq(str text)
```

Parameters

PARAMETER	DESCRIPTION
text	The string to copy and convert characters from.

Return value

The converted text string.

Remarks

The **strColSeq** function exists for backward-compatibility purposes. This function supports only the mapping for the following Western European characters:

- AàáâãäÀÁÂÃÄBCçÇDEèéêëÈÉÊËFGHIìíîïÌÍÎÏJKLMNñÑOòóôõöÒÓÔÕÖPQRSTUùúûüÙÚÛÜVWXYýÝZæøåÆØÅ
- aaaaaaaaaaabcccdeeeeeeeeefghiiiiiiiiijklmnnnooooooooooopqrstuuuuuuuuuvwxyyyz~¦Ç~¦Ç

For Unicode-compliant functionality, use the Win32 LCMapString application programming interface (API) via the **DLL** and **DLLFunc** classes.

Example

The following example prints **abcdeabcde**.

```
    static void strColSeqExample(Args _arg)
    {
            ;
            print strColSeq("");
            pause;
    }
```

strDel

Creates a copy of a string, from which the specified substring is removed.

```
str strDel(str _text, int _position, int _number)
```

Parameters

PARAMETER	DESCRIPTION
_text	The string to copy from.
_position	The position at which to begin ignoring characters during the copy operation.
_number	The number of characters to ignore. A minus sign in front of the *_number* parameter indicates that *_number*–1 characters before the character at *_position* should be removed together with the character at *_position*.

Return value

The remaining characters that are copied from the string.

Remarks

The **strDel** function is complementary to the **substr** function.

```
strDel("ABCDEFGH",2,3); //Returns the string "AEFGH".
strDel("ABCDEFGH",4,3); //Returns the string "ABCGH".
```

strFind

Searches a string for the first occurrence of one of the specified characters.

```
int strFind(str _text, str _characters, int _position, int _number)
```

Parameters

PARAMETER	DESCRIPTION
_text	The string to search.
_characters	The characters to search for.
_position	The position in the string where the search begins.
_number	A signed number that indicates the direction of the search and how many positions to search in the string.

Return value

The value of the position of the first occurrence of one of the specified characters.

Remarks

To search from the beginning of the string to the end, use **1** as the value of the *_position* parameter. If the value of the *_number* parameter is negative, the system searches the number of characters backward from the specified position. The search isn't case-sensitive. Here is an example.

```
strFind("ABCDEFGHIJ","KHD",1,10); //Returns the value 4 (the position where "D" was found).
strFind("ABCDEFGHIJ","KHD",10,-10); //Returns the value 8 (the position where "H" was found).
```

The **strFind** function is complementary to the **strNFind** function.

strFmt

Formats the specified string and substitutes any occurrences of n with the nth argument.

```
str strFmt(str _string, ...)
```

Parameters

PARAMETER	DESCRIPTION
_string	The strings to format.

Return value

The formatted string.

Remarks

If an argument isn't provided for a parameter, the parameter will be returned as "%n" in the string. The string conversion of values of the **real** type is limited to two decimal places. Values are rounded, not truncated. The **System.String::Format** method from the Microsoft .NET Framework can be used to gain additional functionality, as shown in the example.

Example

```
static void strFmtExampleJob(Args _arg)
{
    System.Double sysDouble;
    real r = 8.3456789;
    int  i = 42;
    utcDateTime utc = str2DateTime("2008-01-16 13:44:55" ,321); // 321 == YMD.
    str  s;
    ;
    s = strFmt("real = %1, int = %2, utcDateTime = %3, [%4]", r, i, utc);
    info("X1:  " + s);
    //
    sysDouble = r;
    s = System.String::Format("{0:##.####}", sysDouble);
    info("N1:  " + s);
    //
    s = System.String::Format("{0,6:C}", sysDouble); // $
    info("N2:  " + s);
    /**********  Actual Infolog output
    Message (02:16:05 pm)
    X1:  real = 8.35, int = 42, utcDateTime = 1/16/2008 01:44:55 pm, [%4]
    N1:  8.3457
    N2:   $8.35
    **********/
}
```

strIns

Builds a string by inserting one string into another.

```
str strIns(str _text1, str _text2, int _position)
```

Parameters

PARAMETER	DESCRIPTION
_text1	The string to insert the other string into.
_text2	The string to insert into the other string.
_position	The position where the first character of the *_text2* parameter should occur in the output string.

Return value

The combined text string.

Remarks

The **strIns** function is complementary to the **strDel** function. If the value of the *_position* parameter is more than the length of the original string, the string to insert is appended to the end of the original string.

```
strIns("ABFGH","CDE",3); //Returns the string "ABCDEFGH".
strIns("ABCD","EFGH",10); //Returns the string "ABCDEFGH".
```

strKeep

Builds a string by using only the characters from the first input string that the second input string specifies should be kept.

```
str strKeep(str _text1, str _text2)
```

Parameters

PARAMETER	DESCRIPTION
_text1	The string that contains the characters that can be used to build an output string.
_text2	The string that specifies which characters to keep for the output string.

Return value

A string of the characters that are kept.

Remarks

```
strKeep("ABBCDDEFGHB","BCD"); //Returns the string "BBCDDB".
strKeep("abcZcba","bc") //Returns the string "bccb".
```

The **strKeep** function is complementary to the **strRem** function.

strLen

Calculates the length of the specified string.

```
int strLen(str text)
```

Parameters

PARAMETER	DESCRIPTION
text	The string to calculate the length of.

Return value

The length of the specified string.

Remarks

```
strLen("ABC"); //Returns the value 3.
strLen("ABCDEFGHIJ"); //Returns the value 10.
```

strLine

Retrieves a single line from a string that spans multiple lines.

```
str strLine(str string, int count)
```

Parameters

PARAMETER	DESCRIPTION
string	A string that might span multiple lines.
count	The offset of the line to return.

Return value

A copied line of the string that is specified by the *string* parameter.

Remarks

The first line of the string has an offset of 0. You can assign multiple lines to one string by embedding the *\n* or *\r\n* characters in the string. Additionally, you can use the at sign (@) immediately before the opening quotation mark and use the Enter key to spread parts of the string value over multiple lines in the X++ code editor.

Example

```
str mytxt = "first-line\nsecond-line\nlast-line";
// Prints "second-line".
print strLine(mytxt,1);
// Prints "last-line".
print strLine(mytxt,2);
```

strLTrim

Removes leading blanks from a text string.

```
str strLTrim(str text)
```

Parameters

PARAMETER	DESCRIPTION
text	The string to delete the leading blanks from.

Return value

The string equivalent for the text that leading blanks have been removed from.

Remarks

The **strLTrim** function is complementary to the **strRTrim** function.

Example

```
// Returns the text string "ABC-DEFG".
strLTrim("   ABC-DEFG");
```

strLwr

Converts all letters in the specified string to lowercase.

```
str strLwr(str _text)
```

Parameters

PARAMETER	DESCRIPTION
_text	The string to convert to lowercase.

Return value

A copy of the specified string that contains only lowercase letter.

Remarks

The **strLwr** function is complementary to the **strUpr** function. The **strLwr** function uses the **LCMapString** function in the Win32 API.

Example

```
static void strLwrExample(Args _args)
{
    // Returns the text string "abcdd55efghij".
    print strLwr("Abcdd55EFGHIJ");
    pause;
}
```

strNFind

Searches part of a text string for the first occurrence of a character that isn't included in the specified list of characters.

```
int strNFind(str _text, str _characters, int _position, int _number)
```

Parameters

PARAMETER	DESCRIPTION
_text	The text string to search.
_characters	The list of characters to exclude from the search.
_position	The position in the string at which to begin the search.
_number	A signed number that indicates the direction of the search and how many positions to search. If a minus sign precedes *_number*, the system searches *_number* characters in reverse order from *_position*.

Return value

The position of the first occurrence of a character that isn't specified by the *_characters* parameter.

Remarks

The search isn't case-sensitive. To search from the beginning of the string to the end, use a value of **1** for the *_position* parameter. If a minus sign precedes the value of the *_number* parameter, the characters will be searched in reverse order, starting from the position that is specified by the *_position* parameter.

```
strNFind("ABCDEFGHIJ","ABCDHIJ",1,10); //Returns the value 5 (the position of "E");
strNFind("CDEFGHIJ","CDEFGIJ",10,-10); //Returns the value 6 (the position of "H").
strNFind("abcdef","abCdef",3,2) //Returns the value 0.
strNFind("abcdef", "abcef",3,2) //Returns the value 4.
```

The **strNFind** function is complementary to the **strFind** function.

strPoke

Overwrites part of a string with another string.

```
str strPoke(str _text1, str _text2, int _position)
```

Parameters

PARAMETER	DESCRIPTION
_text1	The original string.
_text2	The string to replace part of the original string with.
_position	The position of the original string at which to begin replacing the characters.

Return value

The new string.

Remarks

The new string can be longer than the original string. However, if the value of the *_position* parameter is more than the length of the string, the original string is returned without replacements.

```
strPoke("12345678","AAA",3); //Returns the string "12AAA678".
strPoke("abcde","4567",4); //Returns the string "abc4567".
strPoke("abcde", "4567", "10"); //Returns the string "abcde".
```

strPrompt

Appends a string with the specified number of period characters, followed by a colon and space character.

```
str strPrompt(str _string, _int len)
```

Parameters

PARAMETER	DESCRIPTION
_string	The original string.
_len	The desired final length of the string.

Return value

A string that looks like a prompt for user input.

Remarks

In atypical cases, where the value of the *_len* parameter is only slightly more than the length of the original string, the highest precedence is given to adding the trailing space. Next, precedence is given to the colon. The lowest precedence is given to the periods. Negative values for the *_len* parameter return the input string appended with a trailing space.

```
strPrompt("ab",-1); //Returns "ab ".
strPrompt("ab",3); //Returns "ab ".
strPrompt("ab",4); //Returns "ab: ".
strPrompt("ab",5); //Returns "ab.: ".
strPrompt("ab",6); //Returns "ab..: ".
```

Example

```
static void JobStrPromptDemo(Args _args)
{
    // Printed string is "[abc..: ]"
    print "[", strPrompt("abc", 7), "]";
    pause;
}
```

strRem

Removes the characters that are specified in one string from another string.

```
str strRem(str text1, str text2)
```

Parameters

PARAMETER	DESCRIPTION
text1	The string to remove characters from.
text2	The characters to exclude from the output string.

Return value

The remaining content of the original string.

Remarks

This function is case-sensitive.

```
strRem("abcd_abcd","Bc"); //Returns the string "abd_abd".
strRem("ABCDEFGABCDEFG","ACEG"); //Returns the string "BDFBDF".
```

This function is complementary to the **strKeep** function.

strRep

Repeats a string of characters.

```
str strRep(str _text, str _number)
```

Parameters

PARAMETER	DESCRIPTION
_text	The string to repeat.
_number	The number of times to repeat the string.

Return value

A new string that contains the contents of the original string that are repeated the specified number of times.

Example

The following example prints the text string **ABABABABABAB**.

```
static void strRepExample(Args _arg)
{
    str strL;
    ;
    strL = strRep("AB",6);
    print strL;
    pause;
}
```

strRTrim

Removes the trailing space characters from the end of a string.

```
str strRTrim(str _text)
```

Parameters

PARAMETER	DESCRIPTION
_text	The string to remove the trailing space characters from .

Return value

A copy of the specified string that doesn't include trailing space characters.

Remarks

```
strRTrim("ABC-DEFG- "); //Returns the string "ABC-DEFG-".
strRTrim(" CD "); //Returns " CD".
```

The **strRTrim** function is complementary to the **strLTrim** function.

strScan

Searches a text string for an occurrence of another string.

```
int strScan(str _text1, str _text2, int _position, int _number)
```

Parameters

PARAMETER	DESCRIPTION
_text1	The string to search in.
_text2	The string to find.
_position	The first position in the *_text1* parameter at which to perform a comparison.
_number	The number of positions in the *_text1* parameter to retry the comparison for. If a minus sign precedes the *_number* parameter, the system searches the number of characters in reverse order from the specified position.

Return value

The position at which the specified string was found in the string; otherwise, **0** (zero).

Remarks

The comparisons aren't case-sensitive. Values for the *_position* parameter that are less than **1** are treated as **1**. The direction of the scan is controlled by the sign that is specified in the *_number* parameter. A positive sign indicates that each successive comparison will start one position closer to the end of the string. A negative sign indicates that each comparison will start one position closer to the start of the string.

```
strScan("ABCDEFGHIJ","DEF",1,10); //Returns the value 4.
strScan ("ABCDEFGHIJ","CDE",10,-10); //Returns the value 3.
```

strUpr

Converts all the letters in a string to uppercase.

```
str strUpr(str _text)
```

Parameters

PARAMETER	DESCRIPTION
_text	The string to convert to uppercase letters.

Return value

A copy of the specified string that contains only lowercase letters.

Remarks

The **strUpr** function is complementary to the **strLwr** function. The **strUpr** function uses the **LCMapString()** function in the Win32 API.

Example

The following example will print **ABCDD55EFGHIJ**.

```
static void strUprExample(Args _args)
{
    print strUpr("Abcdd55EFGhiJ");
    pause;
}
```

subStr

Retrieves part of a string.

```
str subStr(str _text, int _position, int _number)
```

Parameters

PARAMETER	DESCRIPTION
_text	The original string.
_position	The position in the original string where the part to retrieve begins.
_number	A signed integer that indicates the direction and number of positions to retrieve from the original string. If a minus sign precedes *_number*, the system selects the substring backward from the specified position.

Return value

A substring of the original string.

Remarks

If a minus sign precedes the value of the *_number* parameter, the substring will be selected backward from the specified position.

```
subStr("ABCDEFGHIJ",3,5); //Returns the string "CDEFG".
subStr("ABCDEFGHIJ",7,-4); //Returns the string "DEFG".
subStr("abcdef"),2,99) //Returns the string "cdef".
subStr("abcdef",2,3) //Returns the string "bcd".
subStr("abcdef",2,-3); //Returns the string "ab".
```

System tables

2/18/2021 • 79 minutes to read • Edit Online

This topic describes the system tables.

Common

The Common table is the base class for all tables. It does not contain any data. It is primarily used in X++ code to refer to any table in a polymorphic way.

Methods

METHOD	DESCRIPTION
aosValidateDelete	Validates on the server that the specified record can be deleted from a table.
aosValidateInsert	Validates on the server that the specified record can be inserted.
aosValidateRead	Validates on the server that the specified record can be read.
aosValidateUpdate	Validates on the server that the specified record can be updated.
buf2con	Packs the table buffers of an xRecord instance into an X++ container.
canSubmitToWorkflow	Indicates whether submission to workflow is possible.
caption	Gets and sets the caption property of a table.
checkInvalidFieldAccess	Gets and sets invalid field access.
checkRecord	Gets and sets the property that indicates whether to check mandatory fields.
checkRestrictedDeleteActions	Gets and sets the property that indicates whether a record can be deleted.
clear	Removes all rows from the table buffer.
company	Gets and sets the property that indicates a legal entity for the record.
con2buf	Unpacks a container into the table buffers.
concurrencyModel	Gets and sets the default concurrency model to use to update records.
context	Gets and sets the context property.

METHOD	DESCRIPTION
data	Retrieves a row from the table.
dataSource	Retrieves the data source of the table.
dbOpInTransaction	Makes sure that database operations are correctly closed if they fail.
defaultField	Populates default values in a field in the table.
defaultRow	Populates default values in fields in the table in the non-interactive case.
delete	Deletes the current record from the table.
disableCache	Gets and sets the property that indicates whether caching is disabled.
dispose	Releases resources that are used by the xRecord object.
doClear	Removes all rows from the table buffer and bypasses any additional logic in the clear method of the table.
doDelete	Deletes the current record from the table and bypasses any additional logic in the delete method of the table.
doInsert	Inserts the record into the table and bypasses any additional logic in the insert method of the table.
doUpdate	Updates the current record and bypasses any additional logic in the update method of the table.
doValidateDelete	Performs the action to validate that a record can be deleted.
equal	Determines whether the specified object is equal to the current one.
fieldAccessRight	Returns the field access right.
fieldBufferAccessRight	Returns the field access right for the current record.
fieldState	Sets or returns the state of a field in the table buffer.
getAllowRedefault	Returns the list of fields that are allowed to re-default.
getDefaultingDependencies	Returns the container that holds defaulting dependencies.
getExtension	Returns the table extension.
getFieldValue	Gets the value of the specified field from a table buffer.
getInstanceRelationType	Returns the table name that corresponds to the InstanceRelationType ID.

METHOD	DESCRIPTION
getPhysicalTableName	Return the physical table name, which, in the case of the SQL Temp DB table, is the table instance name.
getPresenceFieldData	Retrieves the PresenceInfo value from the specified field.
getSQLStatement	Gets the SQL statement that is used to return records from the database.
getTableInInstanceHierarchy	
getTableType	Indicates the type of the table.
helpField	Retrieves a string that contains the Help text for the specified field.
initValue	Initializes a field to the default value.
inputStatus	Sets or returns the current input status of the table buffer.
insert	Inserts the record into the table.
interactiveContext	Sets or returns the current interactive context of the table buffer.
isFieldDataRetrieved	Checks whether the data of the given field has been retrieved.
isFieldSet	Checks whether a field has a Set or Defaulted state.
isFormDataSource	Indicates whether the data source is a form.
isNewRecord	Returns true if the record is a new record that hasn't been persisted yet.
isPartOfUOWSaveChanges	
isTempDb	Indicates whether the type of the table is SQL TempDB.
isTmp	Indicates whether this is a temporary table.
joinChild	Finds the join child of the current record.
joinParent	Finds the join parent of the current record.
linkPhysicalTableInstance	Checks whether there is a link for the physical table instance for the record.
merge	Merges the current table with the specified table.
modifiedField	Modifies the specified field to the original.

METHOD	DESCRIPTION
modifiedFieldValue	Modifies the specified field to the original value.
orig	Retrieves the original values of the current record.
overwriteSystemfields	Gets and sets the property that indicates whether system fields can be overwritten.
postLoad	Is executed after a record is read.
queryTimedOut	Indicates whether the query exceeded the time limit for execution.
queryTimeout	Gets and sets the property that indicates the time limit for the execution of a query.
readCommittedLock	
readPast	Gets and sets the property that indicates whether to skip rows that are locked by other processes when a record is read.
recordLevelSecurity	Gets and sets the property that indicates whether to apply security on a record level.
relatedTable	Sets or returns the related buffer of a link of a table buffer.
hasRelatedTable	Indicates whether a foreign key constraint buffer is linked with the table.
renamePrimaryKey	Renames the foreign keys in other tables according to the change of the corresponding primary key value in this table.
reread	Rereads the record from the table.
RowCount	Retrieves the number of rows in the table.
selectForUpdate	Gets and sets the property that indicates whether to select records for update when they are read.
selectLocked	Indicates whether to select locked records.
selectRefRecord	Selects the record by referenced field ID.
selectWithRepeatableRead	Gets and sets the property that indicates whether repeatable read is enabled.
setConnection	Sets the user connection for this table.
setCrossPartition	Sets or resets cross-partitioning for the table.
setFieldValue	Sets the field value in the record buffer.

METHOD	DESCRIPTION
setSQLTracing	Enables or disables SQL tracing mode.
setTempDB	
setTmp	Sets the table so that it is not persisted to the database.
setTmpData	Sets the contents of the temporary table to the specified data.
setXDSContext	Sets new XDS context.
skipDatabaseLog	Gets and sets the property that indicates whether to skip database log requests.
skipDataMethods	Gets and sets the property that indicates whether to discard overloaded methods.
skipDeleteActions	Gets and sets the property that indicates whether to skip delete actions on the table.
skipDeleteMethod	Gets and sets the property that indicates whether to discard overloaded methods.
skipEvents	Provides an option to turn off calling the Application.event* methods for the lifetime of an xRecord object.
skipPostLoad	Gets and sets the property that indicates whether to skip executing the xRecord.postLoad method on the table.
skipTTSCheck	Gets and sets the property that indicates whether to skip the check to determine whether the record is selected for update.
suppressWarnings	Gets and sets the property that indicates whether to suppress warnings for this pointer.
tableAccessRight	Returns the table access right.
tableBufferAccessRight	Returns the table access right for the current record.
toolTipField	Retrieves the HelpText value for the specified field.
toolTipRecord	Retrieves the ToolTip value for the current record.
ttsabort	Aborts a transaction that was started by a call to the ttsbegin method.
ttsbegin	Starts a transaction that can be either committed by the ttscommit method or aborted by the ttsabort method.
ttscommit	Commits a transaction that was started by a call to the ttsbegin method.

METHOD	DESCRIPTION
update	Updates the current record.
validateDelete	Determines whether the current record is valid and ready to be deleted from the database.
validateField	Determines whether the specified field is valid.
validateFieldValue	
validateRelations	
validateWrite	Determines whether the current record is valid and ready to be written.
validTimeStateUpdateMode	Sets a valid time state update mode on the cursor.
wasCached	Specifies the location from which the data was retrieved.
write	Updates a record if it exists; otherwise, inserts a record.
xml	Retrieves an XML string that represents the current object.
takeOwnershipOfTempDBTable	
useExistingTempDBTable	

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
CreatedBy	String	CreatedBy		
CreatedDateTime	UtcDateTime	CreatedDateTime		
CreatedTransactionId	Int64	CreatedTransactionId		
dataAreaId	String	DataAreaId		
DEL_CreatedTime	Int	DEL_CreatedTime		
DEL_ModifiedTime	Int	DEL_ModifiedTime		
ModifiedBy	String	ModifiedBy		
ModifiedDateTime	UtcDateTime	ModifiedDateTime		
ModifiedTransactionId	Int64	ModifiedTransactionId		
Partition	Int64	Partition		

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
RelationType	Int64	RelationType		
RowNumber	Int	RowNumber		
SequenceNum	Int64	SequenceNum		
TableId	Int	TableId		
UnionAllBranchId	Int	UnionAllBranchId		

Relations

RELATION	TABLE
dataAreaId	DataArea

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
RecId	No	

Inheritance Hierarchy

xRecord Class Common Table

DataArea

The DataArea table contains a list of companies that have been created in the database.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
alwaysNative	Enum		boolean	
id	String	DataAreaId		ID for an area of data
isVirtual	Enum		boolean	
name	String	UserIdStr		Name
Partition	Int64	Partition		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
timeZone	Enum		Timezone	

Relations

RELATION	TABLE
id	DataArea
Partition	Partitions

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
Id	No	
IdOnly	Yes	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table DataArea Table

DatabaseLog

The DatabaseLog table stores configuration information for the SysDatabaseLog table.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
createdBy	String	CreatedBy		
createdDateTime	UtcDateTime	CreatedDateTime		
dEL_CreatedTime	Integer	DEL_CreatedTime		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
dEL_ModifiedTime	Integer	DEL_ModifiedTime		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
domainId	String	DomainId		ID for the domain
logField	Integer	FieldId		ID for the field
logTable	Integer	TableId		ID for the table
logType	Enum		DatabaseLogType	
modifiedBy	String	ModifiedBy		
modifiedDateTime	UtcDateTime	ModifiedDateTime		
RecId	Int64	RecId		
recVersion	Integer	RecVersion		

Field Groups

FIELD GROUP	FIELDS
logFieldRelation	

Relations

RELATION	TABLE
Relation_DatabaseLog	DEL_DomainInfo

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
Loglist	No	
RecId	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

DEL_AccessRightsList

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
accessType	Enum		AccessType	
accessTypeFkeyUse	Enum		AccessType	
createdBy	String	CreatedBy		
createdDateTime	UtcDateTime	CreatedDateTime		
dEL_CreatedTime	Integer	DEL_CreatedTime		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
dEL_ModifiedTime	Integer	DEL_ModifiedTime		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
domainId	String	DomainId		ID for the domain
elementName	String	UtilElementName		Name of the application element.
groupId	String	UserGroupId		ID for the user group
id	Int			
modifiedBy	String	ModifiedBy		
modifiedDateTime	UtcDateTime	ModifiedDateTime		
parentId	Int			
RecId	Int64	RecId		
recordType	Enum		AccessRecordType	

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
recVersion	Integer	RecVersion		

Relations

RELATION	TABLE
Relation_AccessRightsList1	UtilIdElements
Relation_AccessRightsList2	UtilIdElements
Relation_AccessRightsList3	DEL_DomainInfo
Relation_AccessRightsList4	DEL_UserGroupInfo

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
Element	Yes	
Group	No	
RecId	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table DEL-AccessRightsList Table

DEL_CompanyDomainList

The CompanyDomainList table contains associations between the DomainInfo and DataArea tables. Security rights are granted per domain.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
companyId	String	SelectableDataArea		ID for the company you can select
createdBy	String	CreatedBy		
createdDateTime	UtcDateTime	CreatedDateTime		

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
dEL_CreatedTime	Integer	DEL_CreatedTime		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
dEL_ModifiedTime	Integer	DEL_ModifiedTime		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
domainId	String	DomainId		ID for the domain
modifiedBy	String	ModifiedBy		
modifiedDateTime	UtcDateTime	ModifiedDateTime		
RecId	Int64	RecId		
recVersion	Integer	RecVersion		

Relations

RELATION	TABLE
companyId	DataArea
domainId	DEL_DomainInfo

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
Company	No	
Domain	No	
RecId	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table DEL_CompanyDomainList Table

DEL_DomainInfo

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
id	String	DomainId		ID for the domain
name	String	UserIdStr		Name
RecId	Int64	RecId		
recVersion	Integer	RecVersion		

Relations

RELATION	TABLE
id	DEL_DomainInfo

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
Id	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table DEL_DomainInfo Table

DEL_UserGroupInfo

The UserGroupInfo table contains the list of available user groups.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
id	String	UserGroupId		ID for the user group
name	String	UserIdStr		Name

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
RecId	Int64	RecId		
recVersion	Integer	RecVersion		

Relations

RELATION	TABLE
id	UserGroupInfo

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
Id	No	

Inheritance Hierarchy

xRecord Class Common Table DEL_UserGroupInfo Table

DEL_UserGroupList

The UserGroupList table contains the list of users associated with each user groups.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
createdBy	String	CreatedBy		
createdDateTime	UtcDateTime	CreatedDateTime		
dEL_CreatedTime	Integer	DEL_CreatedTime		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
dEL_ModifiedTime	Integer	DEL_ModifiedTime		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
groupId	String	UserGroupId		ID for the user group
modifiedBy	String	ModifiedBy		
modifiedDateTime	UtcDateTime	ModifiedDateTime		

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
userId	String	UserId		ID for the user

Relations

RELATION	TABLE
Relation_UserGroupList1	DEL_UserGroupInfo
Relation_UserGroupList2	UserInfo

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
GroupId	No	
RecId	No	
UserId	No	

Inheritance Hierarchy

xRecord Class Common Table DEL_UserGroupList Table

ModelSecPolRuntimeEx

The ModelSecPolRuntimeEx table stores the runtime metadata that is necessary to apply security policies.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
ConstrainedTable	String			
ContextString	String			
ContextType	Int			
DEL_ElementHandle	Int			
DEL_IsEnabled	Int			
DEL_LayerId	Int			
ElementHandle	Int			
IsDirty	Int			

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
IsEnabled	Int			
IsModeled	Int			
LayerId	Int			
ModeledQueryDebugInfo	String			
ModeledQueryPackData	Container			
Name	String			
Operation	Int			
PrimaryTableAOTName	String			
QueryObjectAOTName	String			
RecId	Int64	RecId		
recVersion	Integer	RecVersion		

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
ConstrainedTableIdx	Yes	ConstrainedTable
RecIDIdx	No	RecId

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table ModelSecPolRuntimeEx Table

ModelSecPolRuntimeView

The ModelSecPolRuntimeView view shows the runtime metadata for the currently active security policies.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
ConstrainedTable	String			
ContextString	String			
ContextType	Int			
ElementHandle	Int			
IsDirty	Int			
IsModeled	Int			
LayerId	Int			
ModeledQueryDebugInfo	String			
ModeledQueryPackData	Container			
Name	String			
Operation	Int			
PrimaryTableAOTName	String			
QueryObjectAOTName	String			
RecId	Int64	RecId		
recVersion	Integer	RecVersion		

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table ModelSecPolRuntimeView Table

Partitions

The Partitions table contains the list of data partitions in the system.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
createdBy	String	CreatedBy		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
createdDateTime	UtcDateTime	CreatedDateTime		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
dEL_CreatedTime	Integer	DEL_CreatedTime		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
dEL_ModifiedTime	Integer	DEL_ModifiedTime		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
modifiedBy	String	ModifiedBy		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
modifiedDateTime	UtcDateTime	ModifiedDateTime		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
name	String	UserIdStr		Name (This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
PartitionKey	String	PartitionKey		Partition Key (This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
RecId	Int64	RecId		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
recVersion	Integer	RecVersion		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))

Field Groups

FIELD GROUP	FIELDS
AutoIdentification	

Relations

RELATION	TABLE
PartitionKey	Partitions

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
PartitionIdx	No	
RecIDIdx	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table Partitions Table

PrintJobHeader

The PrintJobHeader table contains information regarding the current print job

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
createdBy	String	CreatedBy		
createdDateTime	UtcDateTime	CreatedDateTime		
dataAreaId	String	DataAreaId		
dEL_CreatedTime	Integer	DEL_CreatedTime		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
deviceName	String			
format	Enum		PrintFormat	
jobDescription	String			
jobStatus	Enum		PrintJobStatus	
jobType	String			
numberOfPages	Int			
Partition	Int64	Partition		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
printedBy	String	UserId		ID for the user
printedDate	date			
printedTime	Int			
printerInfo	Container			
printFromPage	Int			
printNumcopies	Int			
printOnServer	Enum		boolean	

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
printToPage	Int			
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
unlimitedPageHeight	Enum		boolean	

Relations

RELATION	TABLE
dataAreaId	DataArea
Partition	Partitions
printedBy	UserInfo

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
CreatedBy	Yes	
CreatedDate	Yes	
JobType	Yes	
RecId	No	

Inheritance Hierarchy

xRecord Class Common Table PrintJobHeader Table

PrintJobPages

The PrintJobPages table contains information regarding the currently printing page of a print job

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
dataAreaId	String	DataAreaId		
numberOfLines	Int			
pageContents	Container			
pageNo	Int			
pagesHeaderRecId	Int64	RecId		Unique ID for the record in the database

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
Partition	Int64	Partition		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
RecId	Int64	RecId		
recVersion	Integer	RecVersion		

Relations

RELATION	TABLE
dataAreaId	DataArea
pagesHeaderRecId	PrintJobHeader
Partition	Partitions
Relation_PrintJobPages1	PrintJobHeader

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
PageNo	No	
RecId	No	

Inheritance Hierarchy

xRecord Class Common Table PrintJobPages Table

SecurableObject

The SecurableObject table contains all security artifacts reference by the security framework.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
ChildName	String	SecurableChildName		The child name of the securable object.
Name	String	SecurableName		The name of the securable object.

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
Type	Enum		SecurableType	

Field Groups

FIELD GROUP	FIELDS
AutoLookup	

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
NameChildTypeIdx	No	
RecIDIdx	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SecurableObject Table

SecurityDuty

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
Identifier	String	SecurityDutyIdentifier		
Name	String	SecurityDutyName		
Description	String	SecurityDutyDescription		

Field Groups

FIELD GROUP	FIELDS
AutoIdentification	Name

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
RecIDIdx	No	RecId
IdentifierIdx	No	Identifier
NameIdx	Yes	Name

Inheritance Hierarchy

xRecord Class Common Table SecurityDuty Table

SecurityEntryPointInferredTables

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
EntryPointName	String	SecurableName		
Type	Enum		SecurableType	
TableName	String	SecurableName		
AllowEdit	Enum		boolean	
AllowCreate	Enum		boolean	
AllowDelete	Enum		boolean	
ValidTimeStateUpdate	Enum		ValidTimeStateUpdate	

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
RecIDIdx	No	RecId
EntryPointTableIdx	No	EntryPointName, Type, TableName, ValidTimeStateUpdate

Inheritance Hierarchy

xRecord Class Common Table SecurityEntryPointInferredTables Table

SecurityEntryPointLink

The SecurityEntryPointLink table contains the entry point to securable object mapping that has been specified on the AOT nodes of menu items and web menu items.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
EntryPoint	Int64	RecId		Unique ID for the record in the database
PermissionOwner	Int64	RecId		Unique ID for the record in the database
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
ValidFrom	UtcDateTime			
ValidTo	UtcDateTime			

Relations

RELATION	TABLE
Relation_SecurityEntryPointLink1	SecurableObject
Relation_SecurityEntryPointLink2	SecurableObject

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
EntryPointIdx	No	
RecIDIdx	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SecurityEntryPointLink Table

SecurityPermission

The SecurityPermission table contains the list of permissions that have been specified on the AOT nodes of forms, reports, security code permissions, and service operations.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
Access	Enum		AccessRight	

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
Group	Enum		AccessRight	
Owner	Int64	RecId		Unique ID for the record in the database
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
SecurableObject	Int64	RecId		Unique ID for the record in the database
ValidFrom	UtcDateTime			
ValidTo	UtcDateTime			

Relations

RELATION	TABLE
Relation_SecurityPermission1	SecurableObject
Relation_SecurityPermission2	SecurableObject

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
OwnerGroupObjectIdx	No	
RecIDIdx	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SecurityPermission Table

SecurityPrivilege

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
Identifier	String	SecurityPrivilegeIdentifier		

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
Name	String	SecurityPrivilegeName		
Description	String	SecurityPrivilegeDescription		

Field Groups

FIELD GROUP	FIELDS
AutoIdentification	Name

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
RecIDIdx	No	RecId
IdentifierIdx	No	Identifier
NameIdx	Yes	Name

Inheritance Hierarchy

xRecord Class Common Table SecurityPrivilege Table

SecurityRole

The SecurityRole table reflects the list of roles defined by the security AOT role node.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
AllowCurrentRecords	Enum		AccessRight	
AllowFutureRecords	Enum		AccessRight	
AllowPastRecords	Enum		AccessRight	
AotName	String	SecurityRoleAotName		The name of the role in the AOT.
ContextString	String			
DEL_AllowCurrentRecords	Enum		AccessRight	
DEL_AllowFutureRecords	Enum		AccessRight	
DEL_AllowPastRecords	Enum		AccessRight	

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
DEL_IsEnabled	Enum		boolean	
Description	String	SecurityRoleDescription		Description of the security role.
IsEnabled	Enum		boolean	
Name	String	SecurityRoleName		The name of the security role.
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
UserLicenseType	Enum		UserLicenseType	

Field Groups

FIELD GROUP	FIELDS
AutoIdentification	

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
AotNameIdx	No	
NameIdx	Yes	
RecIDIdx	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SecurityRole Table

SecurityRoleAssignmentRule

Rules for dynamically assigning users to role

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
MembershipRuleDescription	String	MembershipRuleDescription		Description of the automatic role membership rule
MembershipRuleName	String	MembershipRuleName		Name of the automatic role membership rule
Partition	Int64	Partition		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
RuleQuery	Container			
SecurityRole	Int64	RecId		Unique ID for the record in the database
ValidFrom	UtcDateTime			
ValidTo	UtcDateTime			

Relations

RELATION	TABLE
Partition	Partitions
SecurityRole	SecurityRole
SecurityRoleRelationShip	SecurityRole

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
AlternateKey	No	
RecIDIdx	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not

authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SecurityRoleAssignmentRule Table

SecurityRoleDutyExplodedGraph

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
SecurityRole	Int64	RecId		
SecurityDuty	Int64	RecId		

Relations

RELATION	TABLE
SecurityRole	SecurityRole
SecurityDuty	SecurityDuty

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
RecIDIdx	No	RecId
RoleDutyIdx	No	SecurityRole, SecurityDuty

Inheritance Hierarchy

xRecord Class Common Table SecurityRoleDutyExplodedGraph Table

SecurityRoleExplodedGraph

The SecurityRoleExplodedGraph table contains all role relationships, direct or indirect, as defined by the AOT sub role nodes of the security role nodes.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
RefCount	Int			
SecurityRole	Int64	RecId		Unique ID for the record in the database

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
SecuritySubRole	Int64	RecId		Unique ID for the record in the database

Relations

RELATION	TABLE
Relation_SecurityRole1	SecurityRole
Relation_SecurityRole2	SecurityRole
SecurityRole	SecurityRole
SecuritySubRole	SecurityRole

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
RecIDIdx	No	
RoleSubRoleIdx	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SecurityRoleExplodedGraph Table

SecurityRolePermissionOverride

The SecurityRolePermissionOverride table contains the list of permissions that have been specified on the security role AOT nodes.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
Access	Enum		AccessRight	
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
SecurableObject	Int64	RecId		Unique ID for the record in the database

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
SecurityRole	Int64	RecId		Unique ID for the record in the database
ValidFrom	UtcDateTime			
ValidTo	UtcDateTime			

Relations

RELATION	TABLE
Relation_SecurityRolePermissionOverride1	SecurityRole
Relation_SecurityRolePermissionOverride2	SecurableObject

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
RecIDIdx	No	
RoleObjectIdx	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SecurityRolePermissionOverride Table

SecurityRolePrivilegeExplodedGraph

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
SecurityRole	Int64	RecId		
SecurityPrivilege	Int64	RecId		

Relations

RELATION	TABLE
SecurityRole	SecurityRole
SecurityPrivilege	SecurityPrivilege

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
RecIDIdx	No	RecId
RolePrivilegeIdx	No	SecurityRole, SecurityPrivilege

Inheritance Hierarchy

xRecord Class Common Table SecurityRolePrivilegeExplodedGraph Table

SecurityRoleRuntime

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
SecurityRole	Int64	RecId		
Name	String	SecurableName		
ChildName	String	SecurableChildName		
Type	Enum		SecurableType	
CreateAccess	Int			
ReadAccess	Int			
UpdateAccess	Int			
DeleteAccess	Int			
CorrectAccess	Int			
InvokeAccess	Int			
PastCreateAccess	Int			
PastReadAccess	Int			
PastUpdateAccess	Int			
PastDeleteAccess	Int			
PastCorrectAccess	Int			
PastInvokeAccess	Int			
CurrentCreateAccess	Int			
CurrentReadAccess	Int			

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
CurrentUpdateAccess	Int			
CurrentDeleteAccess	Int			
CurrentCorrectAccess	Int			
CurrentInvoke	Int			
FutureCreateAccess	Int			
FutureReadAccess	Int			
FutureUpdateAccess	Int			
FutureDeleteAccess	Int			
FutureCorrectAccess	Int			
FutureInvokeAccess	Int			

Field Groups

FIELD GROUP	FIELDS
AutoIdentification	Name, ChildName, Type

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
RecIDIdx	No	RecId
RoleIDIdx	Yes	SecurityRole
SecurableObjectIdx	Yes	Type, Name, ChildName

Inheritance Hierarchy

xRecord Class Common Table SecurityRoleRuntime Table

SecurityRoleTaskGrant

The SecurityRoleTaskGrant table contains the list of role to duty mappings and role to privilege mappings as defined by the AOT security role node.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
RecId	Int64	RecId		
recVersion	Integer	RecVersion		

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
SecurityRole	Int64	RecId		Unique ID for the record in the database
SecurityTask	Int64	RecId		Unique ID for the record in the database

Relations

RELATION	TABLE
Relation_SecurityRoleTaskGrant1	SecurityRole
Relation_SecurityRoleTaskGrant2	SecurityTask

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
RecIDIdx	No	
RoleTaskIdx	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SecurityRoleTaskGrant Table

SecuritySegregationOfDutiesConflict

The SecuritySegregationOfDutiesConflict table stores information about segregation of duties conflicts that result from attempted assignments of users to roles, and resolutions to the conflicts provided by authorized users.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
AssignmentMode	Enum		RoleAssignmentMode	
createdBy	String	CreatedBy		
createdDateTime	UtcDateTime	CreatedDateTime		

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
dEL_CreatedTime	Integer	DEL_CreatedTime		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
DEL_ExistingTask	Int64	RecId		
dEL_ModifiedTime	Integer	DEL_ModifiedTime		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
DEL_NewTask	Int64	RecId		
ExistingDuty	Int64	RecId		
ExistingRole	Int64	RecId		Unique ID for the record in the database
ExistingTask	Int64	RecId		Unique ID for the record in the database
modifiedBy	String	ModifiedBy		
modifiedDateTime	UtcDateTime	ModifiedDateTime		
NewDuty	Int64	RecId		
NewRole	Int64	RecId		Unique ID for the record in the database
NewTask	Int64	RecId		Unique ID for the record in the database
Partition	Int64	Partition		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
ReasonForOverride	VarString	SegregationOfDuties OverrideComment		Comment explaining the reason for overriding the segregation of duties violation
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
Resolution	Enum		SegregationOfDuties Resolution	
SegregationOfDuties Rule	Int64	RecId		Unique ID for the record in the database
User	String	UserId		ID for the user

Relations

RELATION	TABLE
ExistingDuty	SecurityDuty
ExistingRole	SecurityRole
ExistingRoleRelationship	SecurityRole
ExistingTaskRelationship	SecurityTask
NewDuty	SecurityDuty
NewRole	SecurityRole
NewRoleRelationship	SecurityRole
NewTaskRelationship	SecurityTask
Partition	Partitions
Relation_SecuritySegregation7	UserInfo
SecuritySODRuleRelationship	SecuritySegregationOfDutiesRule
SegregationOfDutiesRule	SecuritySegregationOfDutiesRule
User	UserInfo

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
AlternateKey	No	
ExistingDutyIdx	Yes	ExistingDuty
ExistingRoleIdx	Yes	
ExistingTaskIdx	Yes	
NewDutyIdx	Yes	NewDuty
NewRoleIdx	Yes	
NewTaskIdx	Yes	
RecIDIdx	No	
UserInfoIdx	Yes	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SecuritySegregationOfDutiesConflict Table

SecuritySegregationOfDutiesRule

The SecuritySegregationOfDutiesRule table stores the rules governing segregation of duties.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
DEL_FirstSecurityTask	Int64	RecId		
DEL_SecondSecurityTask	Int64	RecId		
FirstDuty	Int64	RecId		
FirstSecurityTask	Int64	RecId		Unique ID for the record in the database
Mitigation	String	SecurityMitigation		Mitigation for the risk associated with violating the segregation of duties rule

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
Name	String	SegregationOfDuties RuleName		Name of the segregation of duties rule
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
Risk	String	SecurityRisk		Risk associated with violating the segregation of duties rule
SecondDuty	Int64	RecId		
SecondSecurityTask	Int64	RecId		Unique ID for the record in the database
Severity	Enum		SegregationOfDuties Severity	
ValidFrom	UtcDateTime			
ValidTo	UtcDateTime			

Field Groups

FIELD GROUP	FIELDS
AutoIdentification	

Relations

RELATION	TABLE
FirstDuty	SecurityDuty
FirstSecurityTaskRelationship	SecurityTask
SecondDuty	SecurityDuty
SecondSecurityTaskRelationship	SecurityTask

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
AlternateKey	No	
FirstSecurityDuty	Yes	FirstDuty

INDEX	ALLOW DUPLICATES	FIELDS
NameIdx	No	
RecIDIdx	No	
SecondSecurityDuty	Yes	SecondDuty
SecondSecurityTask	Yes	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SecuritySegregationOfDutiesRule Table

SecuritySubRole

The SecuritySubRole table contains all sub roles that have been specified on the security role AOT nodes.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
SecurityRole	Int64	RecId		Unique ID for the record in the database
SecuritySubRole	Int64	RecId		Unique ID for the record in the database
ValidFrom	UtcDateTime			
ValidTo	UtcDateTime			

Relations

RELATION	TABLE
Relation_SecuritySubRole1	SecurityRole
Relation_SecurityTaskPermission2	SecurityRole
SecurityRole	SecurityRole

RELATION	TABLE
SecuritySubRole	SecurityRole

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
RecIDIdx	No	
RoleSubRoleIdx	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SecuritySubRole Table

SecuritySubTask

The SecuritySubTask table contains the duty to privilege mappings that have been specified on the security duty AOT nodes.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
SecuritySubTask	Int64	RecId		Unique ID for the record in the database
SecurityTask	Int64	RecId		Unique ID for the record in the database
ValidFrom	UtcDateTime			
ValidTo	UtcDateTime			

Relations

RELATION	TABLE
Relation_SecuritySubTask1	SecurityTask
Relation_SecuritySubTask2	SecurityTask

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
RecIDIdx	No	
TaskSubTaskIdx	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SecuritySubTask Table

SecurityTask

The SecurityTask table contains the list of duties and privileges that have been defined by the AOT security duty and security privilege nodes.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
AotName	String	SecurityTaskAotName		The name of the task in the AOT.
Description	String	SecurityTaskDescription		Description of the process cycle, duty, or privilege.
IsEnabled	Enum		boolean	
IsPermissionSet	Enum		boolean	
Name	String	SecurityTaskName		The name of the process cycle, duty, or privilege.
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
Type	Enum		SecurityTaskType	

Field Groups

FIELD GROUP	FIELDS
AutoIdentification	

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
AotNameIdx	No	
NameIdx	Yes	
RecIDIdx	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SecurityTask Table

SecurityTaskEntryPoint

The SecurityTaskEntryPoint table contains the list of privilege to entry point mappings that have been specified on the AOT security privilege node.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
EntryPoint	Int64	RecId		Unique ID for the record in the database
PermissionGroup	Enum		AccessRight	
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
SecurityTask	Int64	RecId		Unique ID for the record in the database
ValidFrom	UtcDateTime			
ValidTo	UtcDateTime			

Relations

RELATION	TABLE
Relation_SecurityTaskEntryPoint1	SecurityTask
Relation_SecurityTaskEntryPoint2	SecurableObject

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
RecIDIdx	No	
TaskEntryPointIdx	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SecurityTaskEntryPoint Table

SecurityTaskExplodedGraph

The SecurityTaskExplodedGraph table contains the duty to privilege mappings that have been specified on the security duty AOT nodes.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
RefCount	Int			
SecuritySubTask	Int64	RecId		Unique ID for the record in the database
SecurityTask	Int64	RecId		Unique ID for the record in the database

Relations

RELATION	TABLE
Relation_SecurityTaskExplodedGraph1	SecurityTask
Relation_SecurityTaskExplodedGraph2	SecurityTask

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
RecIDIdx	No	
SubTaskIdx	Yes	

INDEX	ALLOW DUPLICATES	FIELDS
TaskSubTaskIdx	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SecurityTaskExplodedGraph Table

SecurityTaskPermission

The SecurityTaskPermission table is obsolete.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
Access	Int			
Level	Int			
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
SecurableObject	Int64	RecId		Unique ID for the record in the database
SecurityTask	Int64	RecId		Unique ID for the record in the database

Relations

RELATION	TABLE
Relation_SecurityTaskPermission1	SecurityTask
Relation_SecurityTaskPermission2	SecurableObject

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
RecIDIdx	No	
TaskObjectIdx	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SecurityTaskPermission Table

SecurityTaskPermissionOverride

The SecurityTaskPermissionOverride table contains the list of permissions that have been specified on the security privilege AOT nodes.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
Access	Enum		AccessRight	
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
SecurableObject	Int64	RecId		Unique ID for the record in the database
SecurityTask	Int64	RecId		Unique ID for the record in the database
ValidFrom	UtcDateTime			
ValidTo	UtcDateTime			

Relations

RELATION	TABLE
Relation_SecurityTaskPermissionOverride1	SecurityTask
Relation_SecurityTaskPermissionOverride2	SecurableObject

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
RecIDIdx	No	
TaskObjectIdx	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the

AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SecurityTaskPermissionOverride Table

SecurityUserRole

The SecurityUserRole table contains the user to role mappings.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
AssignmentMode	Enum		RoleAssignmentMode	
AssignmentStatus	Enum		RoleAssignmentStatus	
Partition	Int64	Partition		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
SecurityRole	Int64	RecId		Unique ID for the record in the database
User	String	UserId		ID for the user
ValidFrom	UtcDateTime			
ValidTo	UtcDateTime			

Relations

RELATION	TABLE
Partition	Partitions
Relation_SecurityRole	SecurityRole
Relation_SecurityUserRole3	UserInfo
SecurityRole	SecurityRole

RELATION	TABLE
User	UserInfo

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
RecIDIdx	No	
UserRoleIdx	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SecurityUserRole Table

SecurityUserRoleCondition

The SecurityUserRoleCondition table contains the list of companies that constrain a user to role mappings. If there are no entries for a particular user to role mapping then the user is granted the permissions of that role for all companies.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
ControllingKey	int64			
DataArea	String	DataAreaId		ID for an area of data
Partition	Int64	Partition		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
SecurityUserRole	Int64	RecId		Unique ID for the record in the database

Relations

RELATION	TABLE
DataArea	DataArea
Partition	Partitions
Relation_SecurityUserRoleCondition1	SecurityUserRole
Relation_SecurityUserRoleCondition2	DataArea
SecurityUserRole	SecurityUserRole

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
RecIDIdx	No	
UserRoleDataAreaIdx	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SecurityUserRoleCondition Table

SqlDescribe

The SqlDescribe table is used to store the table and field metadata. The SqlDataDictionary::tablemetadata method populates this table by using a back end database query.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
array	Int			
fieldId	Integer	FieldId		ID for the field
fieldType	Enum		Types	
flags	Int			
name	String	UtilElementName		Name of the application element.
nullable	Enum		boolean	
numericPrecision	Int			

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
numericScale	Int			
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
rightJustify	Enum		boolean	
shadow	Enum		boolean	
sqlName	String	UtilElementName		Name of the application element.
strSize	Int			
tabId	Integer	TableId		ID for the table

Field Groups

FIELD GROUP	FIELDS
fieldIdRelation	

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
Field	No	
RecId	No	
SqlName	Yes	

Inheritance Hierarchy

xRecord Class Common Table SqlDescribe Table

SqlDictionary

The SqlDictionary table describes the current state of the database with respect to the table and field metadata. The table also contains view and table dependency information. The database synchronization engine uses the SqlDictionary table to determine the actions that are required to synchronize the AOT with the database.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
array	Int			
fieldId	Integer	FieldId		ID for the field
fieldType	Enum		Types	

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
flags	Int			
name	String	UtilElementName		Name of the application element.
nullable	Enum		boolean	
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
rightJustify	Enum		boolean	
shadow	Enum		boolean	
sqlName	String	UtilElementName		Name of the application element.
strSize	Int			
tabId	Integer	TableId		ID for the table

Field Groups

FIELD GROUP	FIELDS
fieldIdRelation	

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
Field	No	
RecId	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SqlDictionary Table

SqlParameters

The SqlParameters table stores database related information in the form of parameter and value pairs. This table is not used in Microsoft Dynamics Ax 2009.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
id	Int			
iParm	Int			
iValue	Int			
parm	String			
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
value	String			

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
Parm	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SqlParameters Table

SqlStatistics

The SqlStatistics table stores related database statistics for the user. This table is not used in Microsoft Dynamics Ax 2009.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
dEL_ModifiedTime	Integer	DEL_ModifiedTime		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
indexId	Integer	IndexId		ID for the index
modifiedBy	String	ModifiedBy		
modifiedDateTime	UtcDateTime	ModifiedDateTime		

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
objectType	Enum		SqlStatType	
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
tabId	Integer	TableId		ID for the table
userId	String	UserId		ID for the user
value1	Int			
value10	Int			
value11	Int			
value12	Int			
value2	Int			
value3	Int			
value4	Int			
value5	Int			
value6	Int			
value7	Int			
value8	Int			
value9	Int			

Field Groups

FIELD GROUP	FIELDS
indexIdRelation	

Relations

RELATION	TABLE
Relation_SqlStats1	SqlStatistics
Relation_SqlStats2	UserInfo
tabId	SqlStatistics
userId	UserInfo

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
Id	No	
RecId	No	

Inheritance Hierarchy

xRecord Class Common Table SqlStatistics Table

SqlStorage

The SqlStorage table contains information about table space and its Oracle attributes.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
id	Int			
indexId	Integer	IndexId		ID for the index
objectType	Int			
override	Enum		boolean	
parm	String			
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
tabId	Integer	TableId		ID for the table
value	String			

Field Groups

FIELD GROUP	FIELDS
indexIdRelation	

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
Id	No	

Inheritance Hierarchy

xRecord Class Common Table SqlStorage Table

SqlSyncInfo

The SqlSyncInfo table captures messages and DDL statements during the database synchronization process. Once the synchronization process is complete the information in the table is deleted.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
ID	Int			
LogType	Enum		SqlSyncLogType	
MessageType	Enum		SqlSyncMessageType	
ParentID	Int			
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
Sequence	Int			
SyncTable	Enum		boolean	
TableName	String			
Text	String			
WarningOk	Enum		boolean	

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
TableName	Yes	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateReadUpdateDelete. The Application Object Server authorizes each create, read, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SqlSyncInfo Table

Subquery

The Subquery table is used by position based paging functionality.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
dataAreaId	String	DataAreaId		

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
RecId	Int64	RecId		
recVersion	Integer	RecVersion		

Relations

RELATION	TABLE
dataAreaId	DataArea

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
RecId	No	

Inheritance Hierarchy

xRecord Class Common Table Subquery Table

SysActiveTempTable

The SysActiveTempTable table provides data about the temporary database tables that are currently created. The table is used by the framework to manage the lifetime of these tables.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
createdDateTime	UtcDateTime	CreatedDateTime		
dEL_CreatedTime	Integer	DEL_CreatedTime		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
InstanceId	String	UtilElementName		Name of the application element.
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
RelationTypeId	Int64	RecId		Unique ID for the record in the database
ServerId	Int64	RecId		Unique ID for the record in the database

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
SessionId	Int64	RecId		Unique ID for the record in the database

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
InstanceIdIdx	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SysActiveTempTable Table

SysBCProxyUserAccount

The SysBCProxyUserAccount table stores the business connector proxy information that is entered through the SysBcAliasForm security form. This table always contains one record.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
networkAlias	String	NetworkAlias		
networkDomain	String	NetworkDomain		
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
sid	String	Sid		

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
RecID	No	
Sid	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not

authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SysBCProxyUserAccount Table

SysBreakpointList

The SysBreakpointList table contains a list of developers that have breakpoints in MorphX.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
createdBy	String	CreatedBy		
createdDateTime	UtcDateTime	CreatedDateTime		
dEL_CreatedTime	Integer	DEL_CreatedTime		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
machineName	String	NetworkDomain		
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
userId	String	UserId		ID for the user
version	Int			

Relations

RELATION	TABLE
userId	UserInfo

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
RecId	No	
UserId	Yes	

Inheritance Hierarchy

xRecord Class Common Table SysBreakpointList Table

SysBreakpoints

The SysBreakpoints table contains a list of all the breakpoints in MorphX.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
codePath	Container			
lineNo	Int			
listRecId	Int64	RecId		Unique ID for the record in the database
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
status	Int			
version	Int			

Relations

RELATION	TABLE
listRecId	SysBreakpointList
Relation_SysBreakpoints1	SysBreakpointList

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
ListRecId	No	

Inheritance Hierarchy

xRecord Class Common Table SysBreakpoints Table

SysCacheFlush

The SysCacheFlush table contains data that is used for synchronization of caches across multiple AOS servers.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
ClearData	Container			
FlushData	Container			
FlushVersion	Int			
modifiedDateTime	UtcDateTime	ModifiedDateTime		
RecId	Int64	RecId		

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
recVersion	Integer	RecVersion		
Scope	String	GlobalObjectCacheScope		Name of an instance in the global object cache.

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
CacheScopeIdx	No	
RecIDIdx	No	

Inheritance Hierarchy

xRecord Class Common Table SysCacheFlush Table

SysClientAccessLog

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
ClientComputer	String	UserIdStr		Name
createdBy	String	CreatedBy		
createdDateTime	UtcDateTime	CreatedDateTime		
EventsContainer	Container			
Partition	Int64	Partition		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
SessionId	Int			

Relations

RELATION	TABLE
Partition	Partitions

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
CreatedByIdx	Yes	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateDelete. The Application Object Server authorizes each create and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SysClientAccessLog Table

SysClientSessions

The SysClientSessions contains the data for the client sessions that are currently active in the system.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
clientComputer	String	UserIdStr		Name
clientType	Int			
DataPartition	String	PartitionKey		Partition Key (This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
DEL_company	String			
DEL_Login_time	Int			
helpLanguage	String	InstalledLanguageId		
LoginDateTime	UtcDateTime			
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
ServerId	Int			
SessionId	Int			
sessionType	Int			
sid	String	Sid		

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
Status	Int			
userId	String	UserId		ID for the user
userLanguage	String	InstalledLanguageId		
Version	Int			

Relations

RELATION	TABLE
DataPartition	Partitions
Relation_SysClientSessions1	SysServerSessions
Relation_SysClientSessions2	UserInfo
ServerId	SysServerSessions
userId	UserInfo

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
ServerId	Yes	
SessionId	No	
Status	Yes	Status
Status_ClientType_UserId	Yes	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SysClientSessions Table

SysConfig

The SysConfig table contains license and configuration information.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
configType	Enum		ConfigType	
createdBy	String	CreatedBy		
createdDateTime	UtcDateTime	CreatedDateTime		
dEL_CreatedTime	Integer	DEL_CreatedTime		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
dEL_ModifiedTime	Integer	DEL_ModifiedTime		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
expiration	String			
id	Int			
modifiedBy	String	ModifiedBy		
modifiedDateTime	UtcDateTime	ModifiedDateTime		
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
shadowValue	String			(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
timestamp	String			(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
userCount	Int			
value	String			

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
ConfigType	No	
RecId	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SysConfig Table

SysEncryptionKey

The SysEncryptionKey table stores the encryption key that is used to encrypt the EP query string and post the data parameters.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
createdBy	String	CreatedBy		
createdDateTime	UtcDateTime	CreatedDateTime		
dEL_CreatedTime	Integer	DEL_CreatedTime		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
dEL_ModifiedTime	Integer	DEL_ModifiedTime		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
Key	Container			
modifiedBy	String	ModifiedBy		
modifiedDateTime	UtcDateTime	ModifiedDateTime		
RecId	Int64	RecId		
recVersion	Integer	RecVersion		

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
RecID	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SysEncryptionKey Table

SysGlobalConfiguration

The SysGlobalConfiguration table stores system level global setting that can be used to configure specific components.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
Name	String			
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
ServerId	String			
SettingLevel	Int			
Value	String			

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
NameIdx	No	
RecIDIdx	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SysGlobalConfiguration Table

SysInheritanceRelations

The SysInheritanceRelations framework helper table for table inheritance. The table stores table inheritance hierarchy related information.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
MainTableId	Int			
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
RelatedTableId	Int			

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
Main	No	
RelatedMain	No	

Inheritance Hierarchy

xRecord Class Common Table SysInheritanceRelations Table

SysLastValue

The SysLastValue table is storage for the usage data that is recorded as users navigate the system.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
company	String	SelectableDataArea		ID for the company you can select
designName	String	UtilElementName		Name of the application element.
elementName	String	UtilElementName		Name of the application element.
isKernel	Enum		boolean	
Partition	Int64	Partition		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
RecId	Int64	RecId		

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
recordType	Enum		UtilElementType	
recVersion	Integer	RecVersion		
userId	String	UserId		ID for the user
value	Container			

Relations

RELATION	TABLE
isVirtual_Extern	DataArea
Partition	Partitions
Relation_SysLastValue1	UserInfo
Relation_SysLastValue2	DataArea
userId	UserInfo

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
RecordType	Yes	
UserId	No	

Inheritance Hierarchy

xRecord Class Common Table SysLastValue Table

SysModel

The SysModel table contains information about installed models on the system.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
createdBy	String	CreatedBy		
createdDateTime	UtcDateTime	CreatedDateTime		
Layer	Int64	LayerRecid		The ID of the layer.
modifiedBy	String	ModifiedBy		
modifiedDateTime	UtcDateTime	ModifiedDateTime		

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
State	String			

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
RecIDIdx	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SysModel Table

SysModelElement

The SysModelElement table lists the ModelElements that the installation holds.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
AxId	Integer	UtilElementId		Unique internal identification number of the application object.
ElementType	Int64	ModelElementType		The ID of an ElementType
Name	String			
Origin				
ParentId	Integer	UtilElementParentId		The unique internal identification number of a parent application object
ParentModelElement	Int64	ParentModelElement Recid		The ID of a parent model element
PartOfInheritance	Int			

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
RootModelElement	Int64	RootModelElementRecid		The ID of a root model element

Relations

RELATION	TABLE
ElementType	SysModelElementType
Relation_SysModelElementType	SysModelElementType

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
RecIDIdx	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SysModelElement Table

SysModelElementData

The SysModelElementData table provides the Layer specific data for any SysModelElement.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
createdBy	String	CreatedBy		
createdDateTime	UtcDateTime	CreatedDateTime		
Layer	Int64	LayerRecid		The ID of the layer.
LegacyId	Int			
ModelElement	Int64	ModelElementRecid		The ID of a ModelElement
ModelId	Integer	ModelId		The ID of the model.

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
modifiedBy	String	ModifiedBy		
modifiedDateTime	UtcDateTime	ModifiedDateTime		
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
SaveCount	Int			

Relations

RELATION	TABLE
Layer	SysModelLayer
ModelElement	SysModelElement
ModelId	SysModel
Relation_SysModel	SysModel
Relation_SysModelElement	SysModelElement
Relation_SysModelLayer	SysModelLayer

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
RecIDIdx	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SysModelElementData Table

SysModelElementDataOld

The SysModelElementDataOld table provides the Layer specific data for any SysModelElementOld.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
createdBy	String	CreatedBy		

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
createdDateTime	UtcDateTime	CreatedDateTime		
Layer	Int64	LayerRecid		The ID of the layer.
LegacyId	Int			
ModelElement	Int64	ModelElementRecid		The ID of a ModelElement
ModelId	Integer	ModelId		The ID of the model.
modifiedBy	String	ModifiedBy		
modifiedDateTime	UtcDateTime	ModifiedDateTime		
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
SaveCount	Int			

Relations

RELATION	TABLE
Layer	SysModelLayerOld
ModelElement	SysModelElementOld
ModelId	SysModelOld
Relation_SysModelElementOld	SysModelElementOld
Relation_SysModelLayerOld	SysModelLayerOld
Relation_SysModelOld	SysModelOld

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
RecIDIdx	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SysModelElementDataOld Table

SysModelElementLabel

The SysModelElementLabel table contains the label text for a given language.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
Comment	String			
Id	Int			
LabelId	String			
Language	String			
Module	String			
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
Text	String			

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
ModuleLangIDIdx	No	
RecIDIdx	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SysModelElementLabel Table

SysModelElementLabelOld

The SysModelElementLabelOld table contains the label text for a given language.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
Comment	String			

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
Id	Int			
LabelId	String			
Language	String			
Module	String			
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
Text	String			

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
RecIDIdx	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SysModelElementLabelOld Table

SysModelElementOld

The SysModelElementOld table lists the ModelElements that the installation holds.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
AxId	Integer	UtilElementId		Unique internal identification number of the application object.
ElementType	Int64	ModelElementType		The ID of an ElementType
Name	String			
Origin				

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
ParentId	Integer	UtilElementParentId		The unique internal identification number of a parent application object
ParentModelElement	Int64	ParentModelElement Recid		The ID of a parent model element
PartOfInheritance	Int			
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
RootModelElement	Int64	RootModelElementRecid		The ID of a root model element

Relations

RELATION	TABLE
ElementType	SysModelElementTypeOld
Relation_SysModelElementTypeOld	SysModelElementTypeOld

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
RecIDIdx	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SysModelElementOld Table

SysModelElementSource

The SysModelElementSource table contains the Source Text for all SysModelElements that have source.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
Layer	Int64	LayerRecid		The ID of the layer.

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
ModelElement	Int64	ModelElementRecid		The ID of a ModelElement
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
Source	Container			

Relations

RELATION	TABLE
Layer	SysModelElementData
Relation_SysModelElementData	SysModelElementData

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
RecIDIdx	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SysModelElementSource Table

SysModelElementSourceOld

The SysModelElementSourceOld table contains the Source Text for all SysModelElementsOld that have source.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
Layer	Int64	LayerRecid		The ID of the layer.
ModelElement	Int64	ModelElementRecid		The ID of a ModelElement
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
Source	Container			

Relations

RELATION	TABLE
Layer	SysModelElementDataOld
Relation_SysModelElementDataOld	SysModelElementDataOld

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
RecIDIdx	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SysModelElementSourceOld Table

SysModelElementType

The SysModelElementType table specifies the possible SysModelElement types. Its Recid is backwards compatible with the UtilRecordType enum for the 'old' element types.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
Name	String	ModelElementTypeName		The name of the element type.
ParentType	Int64	ModelElementType		The ID of an ElementType
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
TreeNodeName	String			

Relations

RELATION	TABLE
Name	SysModelElementType

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
RecIDIdx	No	
TypeNameIdx	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SysModelElementType Table

SysModelElementTypeOld

The SysModelElementTypeOld table specifies the possible SysModelElementOld types. Its Recid is backwards compatible with the UtilRecordType enum for the 'old' element types.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
Name	String	ModelElementTypeName		The name of the element type.
ParentType	Int64	ModelElementType		The ID of an ElementType
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
TreeNodeName	String			

Relations

RELATION	TABLE
Name	SysModelElementType

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
RecIDIdx	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not

authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SysModelElementTypeOld Table

SysModelLayer

The SysModelLayer table lists the possible LayerId and Name. If Model data exists in a layer it reports the aggregated version number for that layer.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
IsDirty	Int			
Layer	Enum		UtilEntryLevel	
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
VersionNumber	Int			

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
RecIDIdx	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SysModelLayer Table

SysModelLayerOld

The SysModelLayerOld table lists the possible LayerId and Name. If Model data exists in a layer it reports the aggregated version number for that layer.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
Layer	Enum		UtilEntryLevel	
RecId	Int64	RecId		
recVersion	Integer	RecVersion		

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
VersionNumber	Int			

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
RecIDIdx	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SysModelLayerOld Table

SysModelManifest

The SysModelManifest table contains the manifest information about deployed models, such as Description, Publisher and Version of a model

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
Category	Integer	ModelManifestCategoryRecId		The ID of the model category
Description	String	ModelDescription		The description of the model.
DisplayName	String	ModelDisplayName		The display name of the model.
Model	Int64	ModelRecid		The ID of the model.
Name	String	ModelName		The name of the model in the model store.
Publisher	String	ModelPublisher		The publisher of the model.
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
Signed	Int			
VersionBuildNo	Int			

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
VersionMajor	Int			
VersionMinor	Int			
VersionRevision	Int			

Field Groups

FIELD GROUP	FIELDS
AutoIdentification	

Relations

RELATION	TABLE
Category	SysModelManifestCategory
Model	SysModel
Relation_SysModel	SysModel
Relation_SysModelManifestCategory	SysModelManifestCategory

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
ModelNameIdx	No	
RecIDIdx	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SysModelManifest Table

SysModelManifestCategory

The SysModelManifestCategory table contains the category aspect of the manifest information for deployed models.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
Name	String	ModelManifestCategoryName		The name of the model category.
RecId	Int64	RecId		
recVersion	Integer	RecVersion		

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
NameIdx	No	
RecIDIdx	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SysModelManifestCategory Table

SysModelManifestCategoryOld

The SysModelManifestCategoryOld table contains the category aspect of the manifest information for deployed models.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
Name	String	ModelManifestCategoryName		The name of the model category.
RecId	Int64	RecId		
recVersion	Integer	RecVersion		

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
RecIDIdx	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not

authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SysModelManifestCategoryOld Table

SysModelManifestOld

The SysModelManifestOld table contains the manifest information about deployed models, such as Description, Publisher and Version of a model.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
Category	Integer	ModelManifestCategoryRecId		The ID of the model category
Description	String	ModelDescription		The description of the model.
DisplayName	String	ModelDisplayName		The display name of the model.
Model	Int64	ModelRecidOld		The ID of the model (old).
Name	String	ModelName		The name of the model in the model store.
Publisher	String	ModelPublisher		The publisher of the model.
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
Signed	Int			
VersionBuildNo	Int			
VersionMajor	Int			
VersionMinor	Int			
VersionRevision	Int			

Field Groups

FIELD GROUP	FIELDS
AutoIdentification	

Relations

RELATION	TABLE
Category	SysModelManifestCategoryOld
Model	SysModelOld
Relation_SysModelManifestCategoryOld	SysModelManifestCategoryOld
Relation_SysModelOld	SysModelOld

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
ModelNameIdx	No	
RecIDIdx	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SysModelManifestOld Table

SysModelOld

The SysModelOld table contains information about installed models on the system.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
createdBy	String	CreatedBy		
createdDateTime	UtcDateTime	CreatedDateTime		
Layer	Int64	LayerRecid		The ID of the layer.
modifiedBy	String	ModifiedBy		
modifiedDateTime	UtcDateTime	ModifiedDateTime		
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
State	String			

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
RecIDIdx	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SysModelOld Table

SysOccConfiguration

The SysOccConfiguration table stores the global concurrency model setting and updates the conflict exception login policy.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
AutoUpdateRecVersion	Enum		boolean	
GlobalOccMode	Enum		GlobalOccMode	
LogHandledUpdateConflicts	Enum		boolean	
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
UniqueIndex	Int			
UseReadUncommitedForAll	Enum		boolean	

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
UniqueIndex	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

SysRecordLevelSecurity

The SysRecordLevelSecurity table contains all the record level security restrictions that are configured by the system administrator. The restrictions are persisted on a per company, per group basis.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
_unused	Enum		boolean	
companyId	String	SelectableDataArea		ID for the company you can select
createdBy	String	CreatedBy		
createdDateTime	UtcDateTime	CreatedDateTime		
dEL_CreatedTime	Integer	DEL_CreatedTime		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
DEL_groupId	String	UserGroupId		ID for the user group
dEL_ModifiedTime	Integer	DEL_ModifiedTime		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
modifiedBy	String	ModifiedBy		
modifiedDateTime	UtcDateTime	ModifiedDateTime		
Partition	Int64	Partition		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
restriction	Container			

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
SecurityRole	Int64	RecId		Name of the security role
tabId	Integer	TableId		ID for the table

Relations

RELATION	TABLE
companyId	DataArea
DEL_groupId	UserGroupInfo
Partition	Partitions
Relation_SecurityRole	SecurityRole
SecurityRole	SecurityRole

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
RecId	No	
Role	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SysRecordLevelSecurity Table

SysServerSessions

The SysServerSessions table is used to store information about the active AOS Servers in the system.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
AOSAccount	String			
AOSId	String			
DEL_LastUpdateTime	Int			

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
DEL_Login_time	Int			
Instance_Name	String			
LastUpdateDateTime	UtcDateTime			
LoadBalance	Int			
LoginDateTime	UtcDateTime			
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
ServerId	Int			
Status	Int			
Version	Int			
Workload	Int			

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
LoadBalance	Yes	
ServerId	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SysServerSessions Table

SysSetbasedHelper

The SysSetbasedHelper framework helper table for table inheritance set-based operations.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
CandidateRecId	Int64	RecId		Unique ID for the record in the database

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
RecId	Int64	RecId		
recVersion	Integer	RecVersion		

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
CandidateRecIDIdx	No	
RecIDIdx	No	

Inheritance Hierarchy

xRecord Class Common Table SysSetbasedHelper Table

SystemSequences

The SystemSequences table holds the next available record ID block for each table.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
cycle	Enum		boolean	
dataAreaId	String	DataAreaId		
id	Int			
maxVal	Int64	RecId		Unique ID for the record in the database
minVal	Int64	RecId		Unique ID for the record in the database
name	String			
nextVal	Int64	RecId		Unique ID for the record in the database
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
tabId	Integer	TableId		ID for the table

Relations

RELATION	TABLE
dataAreaId	DataArea

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
Id	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SystemSequences Table

TableCollectionList

The TableCollectionList table stores the mapping between table collections and virtual companies.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
Partition	Int64	Partition		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
tableCollection	String	UtilElementName		Name of the application element.
virtualDataArea	String	VirtualDataArea		ID for a virtual company

Relations

RELATION	TABLE
isVirtual_Extern	DataArea
parentId_Extern	UtilElements
Partition	Partitions

RELATION	TABLE
Relation_CollectionList1	UtilElements
Relation_CollectionList2	DataArea

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
VirtualDataArea	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table TableCollectionList Table

TimeZonesList

The TimeZonesList table contains the list of the time zones that are supported.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
EnumName	String			
EnumPosition	Int			
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
TimeZoneKeyName	String			
TzEnum	Enum		Timezone	

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
EnumPosition	No	
TzEnum	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object

Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table TimeZonesList Table

TimeZonesRulesData

The TimeZonesRulesData table contains the GMT offsets and daylight saving time information for all time zones that are supported.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
Bias	Int			
DBias	Int			
DDay	Int			
DDayOfWeek	Int			
DHour	Int			
DMinute	Int			
DMonth	Int			
DSecond	Int			
DYear	Int			
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
RuleId	Int			
SBias	Int			
SDay	Int			
SDayOfWeek	Int			
SHour	Int			
SMinute	Int			
SMonth	Int			
SSecond	Int			

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
SYear	Int			
TzEnum	Enum		Timezone	
Year	Int			

Relations

RELATION	TABLE
Relation_TimeZonesRulesData1	TimeZonesList
TzEnum	TimeZonesList

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
RuleId	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table TimeZonesRulesData Table

UserDataAreaFilter

The UserDataAreaFilter table contains a list of selectable companies for a user. It is populated by invoking the populateSelectableCompanies method on the SecurityRights class.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
DataArea	String	DataAreaId		ID for an area of data
Partition	Int64	Partition		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
RecId	Int64	RecId		
recVersion	Integer	RecVersion		

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
User	String	UserId		ID for the user

Relations

RELATION	TABLE
DataArea	DataArea
Partition	Partitions
Relation_DataArea	DataArea
Relation_User	UserInfo
User	UserInfo

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
DataAreaIdx	No	
RecIdIdx	No	

Inheritance Hierarchy

xRecord Class Common Table UserDataAreaFilter Table

UserInfo

The UserInfo table contains a list of users and their active directory and default information.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
accountType	Enum		UserAccountType	
autoInfo	Int			
autoLogOff	Int			
autoUpdate	Int			
clientAccessLogLevel	Int			
company	String	SelectableDataArea		ID for the company you can select
compilerWarningLevel	Enum		CompilerWarningLevel	
confirmDelete	Int			

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
confirmUpdate	Int			
credentialRecId	int64			
debuggerPopup	Int			
debugInfo	Int			
defaultPartition	Enum		boolean	(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
DEL__unused1	String			
DEL__unused2	String			
DEL_defaultModelId	Int			
DEL_osAccountName	String			
DEL_password	String			
DEL_startupMenu	String	UtilElementName		Name of the application element.
enable	Enum		boolean	
enabledOnce	Enum		boolean	
externalId	String			
externalIdType	Enum		ExternalIdType	
externalUser	Enum		boolean	
filterByGridOnByDefault	Enum		boolean	
formFontName	String			
formFontSize	Int			
garbagecollectlimit	Int			
generalInfo	Int			

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
globalExcelExportFilePath	String			(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
globalExcelExportLocation	Int			
globalExcelExportMode	Int			
globalFormOpenMode	Int			
globalListPageLinkMode	Int			
helplanguage	String	InstalledLanguageId		
historyLimit	Int			
homePageRefreshDuration	Int			
id	String	UserId		ID for the user
IdentityProvider	String	NetworkDomain		
infologLevel	Int			
issuerRecId	int64			
language	String	InstalledLanguageId		
messageLimit	Int			
name	String	UserIdStr		Name
networkAlias	String	NetworkAlias		
networkDomain	String	NetworkDomain		
notifyTimeZoneMismatch	Enum		boolean	

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
Partition	Int64	Partition		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
preferredCalendar	Enum		PreferredCalendar	
PreferredLocale	String	PreferredLocale		
preferredTimeZone	Enum		Timezone	
propertyFontName	String			
propertyFontSize	Int			
querytimeLimit	Int			
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
reportBottomMargin	String			
reportFontName	String			
reportFontSize	Int			
reportLeftMargin	String			
reportRightMargin	String			
reportTopMargin	String			
showAOTLayer	Int			
showModelNameInAOT	Int			
showStatusLine	Int			
showToolbar	Int			
sid	String	Sid		
startupProject	String	UtilElementName		Name of the application element.
statuslineInfo	Int			

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
toolbarInfo	Int			
traceInfo	Int			

Field Groups

FIELD GROUP	FIELDS
AutoLookup	id, accountType, name, networkAlias, networkDomain, enable
AutoReport	

Relations

RELATION	TABLE
id	UserInfo
isVirtual_Extern	DataArea
Partition	Partitions
Relation_UserInfo	DataArea

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
Id	No	
IdOnly	Yes	
Sid	Yes	
SidOnly	Yes	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateDelete. The Application Object Server authorizes each create and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table UserInfo Table

UserInfoStartupModel

The UserInfoStartupModel table holds the preferred startup model for each layer for each user.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
Layer	Enum		UtilEntryLevel	
ModelId	Int64	ModelRecid		The ID of the model.
Partition	Int64	Partition		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
UserId	String	UserGroupId		ID for the user group

Relations

RELATION	TABLE
ModelId	SysModelManifest
Partition	Partitions
Relation_SysModelManifest	SysModelManifest
Relation_UserInfo	UserInfo
Relation_UserInfoStartupModel3	DEL_UserGroupInfo
UserId	UserInfo

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
RecIDIdx	No	
UserID_Layer_Idx	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table UserInfoStartupModel Table

UtilElements

The UtilElements table contains the application that is shown in the AOT.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
baseVersion	Int			
code	Container			
createdBy	String	CreatedBy		
createdDateTime	UtcDateTime	CreatedDateTime		
dEL_CreatedTime	Integer	DEL_CreatedTime		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
dEL_ModifiedTime	Integer	DEL_ModifiedTime		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
modifiedBy	String	ModifiedBy		
modifiedDateTime	UtcDateTime	ModifiedDateTime		
name	String	UtilElementName		Name of the application element.
parentId	Int			
RecId	Int64	RecId		
recordType	Enum		UtilElementType	
recVersion	Integer	RecVersion		
saveCount	Int			
source	Container			
utilLevel	Enum		UtilEntryLevel	
version	Int			

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
name	No	recordType, parentId, name, utilLevel
RecId	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table UtilElements Table

UtilElementsOld

The UtilElementsOld table contains the application model stored in the application folder. It is used during the upgrade process.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
baseVersion	Int			
code	Container			
createdBy	String	CreatedBy		
createdDateTime	UtcDateTime	CreatedDateTime		
dEL_CreatedTime	Integer	DEL_CreatedTime		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
dEL_ModifiedTime	Integer	DEL_ModifiedTime		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
modifiedBy	String	ModifiedBy		
modifiedDateTime	UtcDateTime	ModifiedDateTime		
name	String	UtilElementName		Name of the application element.

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
parentId	Int			
RecId	Int64	RecId		
recordType	Enum		UtilElementType	
recVersion	Integer	RecVersion		
saveCount	Int			
source	Container			
utilLevel	Enum		UtilEntryLevel	
version	Int			

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
name	No	recordType, parentId, name, utilLevel
RecId	No	

Inheritance Hierarchy

xRecord Class Common Table UtilElementsOld Table

UtilIdElements

The UtilIdElements table contains the application model shown in the AOT.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
baseVersion	Int			
code	Container			
createdBy	String	CreatedBy		
createdDateTime	UtcDateTime	CreatedDateTime		
dEL_CreatedTime	Integer	DEL_CreatedTime		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
dEL_ModifiedTime	Integer	DEL_ModifiedTime		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
id	Int			
modifiedBy	String	ModifiedBy		
modifiedDateTime	UtcDateTime	ModifiedDateTime		
name	String	UtilElementName		Name of the application element.
parentId	Int			
RecId	Int64	RecId		
recordType	Enum		UtilElementType	
recVersion	Integer	RecVersion		
saveCount	Int			
utilLevel	Enum		UtilEntryLevel	
version	Int			

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
Id	No	
name	No	recordType, parentId, name, utilLevel
RecId	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table UtilIdElements Table

UtilIdElementsOld

The UtilIdElementsOld table contains the application model stored in the application folder. It is used during the upgrade process.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
baseVersion	Int			
code	Container			
createdBy	String	CreatedBy		
createdDateTime	UtcDateTime	CreatedDateTime		
dEL_CreatedTime	Integer	DEL_CreatedTime		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
dEL_ModifiedTime	Integer	DEL_ModifiedTime		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
id	Int			
modifiedBy	String	ModifiedBy		
modifiedDateTime	UtcDateTime	ModifiedDateTime		
name	String	UtilElementName		Name of the application element.
parentId	Int			
RecId	Int64	RecId		
recordType	Enum		UtilElementType	
recVersion	Integer	RecVersion		
saveCount	Int			
utilLevel	Enum		UtilEntryLevel	
version	Int			

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
Id	No	
name	No	recordType, parentId, name, utilLevel
RecId	No	

Inheritance Hierarchy

xRecord Class Common Table UtilIdElementsOld Table

UtilModels

The UtilModels table contains information about models that are installed on the system.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
CategoryId	Int			
Description	String	ModelDescription		The description of the model.
DisplayName	String	ModelDisplayName		The display name of the model.
Id	Int			
InstallMode	Int			
Layer	Enum		UtilEntryLevel	
MarkedForRemoval	Int			
ModelGroupId	Int			
Name	String	ModelName		The name of the model in the model store.
Publisher	String	ModelPublisher		The publisher of the model.
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
Signed	Int			
State	String			

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
VersionBuildNo	Int			
VersionMajor	Int			
VersionMinor	Int			
VersionRevision	Int			

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
ModelIDIdx	No	
ModelNameIdx	No	
RecIDIdx	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table UtilModels Table

VirtualDataAreaList

The VirtualDataAreaList table stores the mapping between real companies and virtual companies.

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
id	String	SelectableDataArea		ID for the company you can select
Partition	Int64	Partition		(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))
RecId	Int64	RecId		
recVersion	Integer	RecVersion		

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
virtualDataArea	String	VirtualDataArea		ID for a virtual company

Field Groups

FIELD GROUP	FIELDS
virtualDataAreaRelation	

Relations

RELATION	TABLE
isVirtual_Extern	DataArea
Partition	Partitions
Relation_DataAreaList1	DataArea
Relation_DataAreaList2	DataArea

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
Id	No	
RecId	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table VirtualDataAreaList Table

VSAssembly

The VSAssembly table contains synchronization information that describes the last time an assembly that is stored under the Visual Studio Projects node in the AOT was deployed..

Fields

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
DeployTo	Enum		DeployTo	
Name	String	AssemblyName		

FIELD	TYPE	EXTENDED TYPE	ENUMERATION TYPE	DESCRIPTION
ProjectName	String	ProjectName		
ProjectType	String	ProjectType		
RecId	Int64	RecId		
recVersion	Integer	RecVersion		
ServerId	Int			
UpdatedDate	UtcDateTime			

Indexes

INDEX	ALLOW DUPLICATES	FIELDS
NameIdx	No	

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table VSAssembly Table

Extensibility home page

2/18/2021 • 3 minutes to read • Edit Online

Dynamics 365 Finance, Supply Chain, and Commerce are extensively customized by partners, value added resellers (VARs), and even some customers. The ability to customize the product is a strength that has historically been supported through overlayering of the application code. The move to the cloud, together with more agile servicing and frequent updates, requires a less intrusive customization model, so that updates are less likely to affect custom solutions. This new model is called *extensibility* and has replaced customization through overlayering.

Extensibility is the only customization framework in Finance, Supply Chain, and Commerce. Overlayering isn't supported.

Introduction

These introductory topics contain general information about customization. This information includes information about when the transition occurs from customization through overlayering to a purely extension-based model. These topics also explain how to log extensibility requests to Microsoft, and provide answers to frequently asked questions (FAQ).

- Application extensibility plans
- Extensibility requests
- Extensibility FAQ

What's new

Read What's new or changed for extensibility for extensibility-related updates that have been made since July 2017.

Getting started

The topics in this section will help you start to build extensions. They will also help you migrate solutions that are currently based on overlayered code to extension-based solutions. This section includes hands-on labs that walk you through simple customizations.

- Migrate from overlayering to extensions
- Customize model elements through extension
- Customize through extension and overlayering

Fundamentals on extensions

This section includes fundamentals, principles, and practices for making extensions. The guiding principles in these topics discuss how customization must be approached through extensions. These principles include naming guidelines. Additionally, these topics discuss the foundation framework, such as extensions and chain of command.

- Intrusive customizations
- Class extension model in X++
- Class extension - Method wrapping and Chain of Command
- Naming guidelines for extensions

- Relax model restrictions to refactor overlayering into extensions

How do I create extensions?

This section includes "How do I?" topics that explain how to customize specific object types or code. Most of these topics are brief and to the point. Because there are many topics here, it might be practical to search for a specific topic.

Data types

- Add values to enums through extension
- Modify extended data types (EDTs) through extension

Classes

- Register subclasses for factory methods
- Respond by using EventHandlerResult
- Extend the RunBase class
- Customize application startup by using delegates

Tables

- Modify existing fields in a table through extension
- Add fields to tables through extension
- Add indexes to tables through extension
- Add relations to tables through extension
- Modify table properties through extension
- Add methods to tables through extension
- Perform business actions throughout the lifecycle of table records

Forms

- Add a new data source to a form
- Change the captions of forms through extension
- Modify the properties of form controls through extension

Others

- Extending decimal point precision for selected data types
- Add new inventory dimensions through extension

Reports

- Extend the list of Electronic reporting (ER) functions
- Customize App Suite reports by using extensions

Blog posts

Information about customization is also shared through blogs where various topics are discussed. This section includes reference to some of these blogs.

- Extending Dynamics 365 for Finance and Operations
- Extension methods
- Extensible base enumerations
- Static event subscription
- Subscribing to onValidatingWrite
- Embrace the extensions mindset with Dynamics 365 for Finance and Operations
- Extensible X++ - Method Signatures

How do I create an extensible solution?

This section includes some best practices on how to create/make your solution extensible, so that consumers of your code can extend your solution.

- Write extensible code
- Classes
- Methods
- Forms
- Extended data types
- Extensible enums
- Delegates
- Tables
- Attributes that make methods extensible

Breaking changes

When you make your solution extensible, you also help guarantee that you won't break those extension points later.

- For pointers that can help you avoid breaking your consumers, see Breaking changes.
- The compatibility checker tool can detect metadata breaking changes against a given baseline release or update, helping to ensure backward compatibility.

Application extensibility roadmap

2/18/2021 • 2 minutes to read • Edit Online

Reducing implementation and upgrade effort is a major initiative for the development team. The benefits of this initiative are to enable you to quickly take advantage of new innovations from Microsoft and your partners, reduce the total cost of ownership, and improve quality. A major part of this initiative is to change the customization approach for the product. In Dynamics AX 2012, several extension capabilities were added to the product. For example, the ability to do event-based customization using methods pre-and post-events was introduced. Extension capabilities have continued to grow in the evolution to the new application.

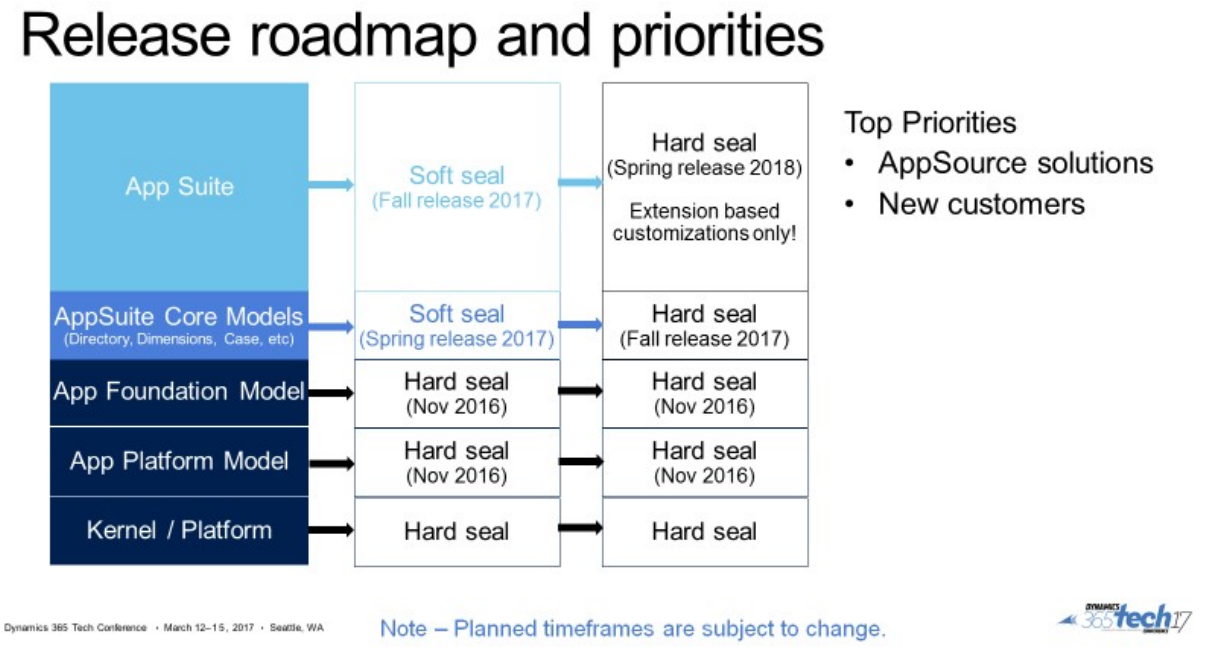
Extension-based customizations have several advantages over the legacy approach of overlayering-based customizations, especially when it comes to reducing implementation and upgrade effort.

- Overlayering-based customizations require code upgrade, recompile time, and extensive testing. This limits the ability to seamlessly apply hot fixes. These costs can be an inhibitor for customers to upgrade to newer versions containing innovations from Microsoft and partners.
- Extension-based customizations also improve the development experience. Models containing overlayered customizations must be in the same package as the base objects. This results in longer compile cycles and larger package distributions. Extensions are also much easier to unit test in isolation from the base object.
- Reducing upgrade costs through extension-based customizations reduces the support matrix for partners as fewer release combinations will need to be supported.

For these reasons, we have gradually been sealing the product models, so they only support extension-based customizations. **AppPlatform** and **AppFoundation** were the first. These models were sealed for overlayering in Platform update 3 (November 2016). Binary updates are now provided to these models on a monthly basis, achieving our goals of reducing upgrade cost and delivering innovation to our customers at a faster cadence.

With Microsoft Dynamics 365 for Finance and Operations release 8.0, we have sealed all product models. Now only extension based customizations are supported.

The following illustration shows the roadmap we followed as we moved to extensions, away from overlayering.



NOTE

A soft seal results in a compiler warning upon overlayering. A hard seal results in a compiler error upon overlayering.

The Modern support policy provides three years of support for a release. Given this, overlayered code will continue to be supported for three years starting November 2017 on the Microsoft Dynamics 365 for Finance and Operations, Enterprise edition 7.3 release. However, this code will not be moved forward to subsequent product releases until the overlayered code is moved to extensions.

There is a substantial amount of work for Microsoft, partners, and customers to accomplish this goal. Workshops, office hours, Help topics, and additional resources are available for training and collaboration in this ecosystem. Internally, we are ready to build more extensibility features in both the core platform and the application. We're working closely with partners with applications on AppSource to define patterns as they migrate to extensions.

The benefits of reducing upgrade friction and enabling innovation uptake will be worth the effort to remove overlayering.

Extensibility requests

2/18/2021 • 6 minutes to read • Edit Online

Finance and Operations applications exclusively use extensions to customize the product. We're aware that this change impacts our entire partner ecosystem. We recommend that you read the resources listed on the Extensibility home page. These resources answer many questions and prepare you for building solutions using extensions.

You will discover that some customizations, which were possible with overlayering, cannot be done through extensions. To enable the same business requirements without overlayering, we have added many extension capabilities and expect to add more going forward. For some customizations that were done with overlayering, you will need to log requests, to make us aware of what you need.

What we are doing

We've been working toward an extension-based customization model for some time. Over the past several releases we have been gradually sealing models. As of Dynamics 365 for Finance and Operations release 8.0, this completes the sealing. From this release forward, only extension-based customizations are allowed.

In future releases, we will be adding even more extensibility capabilities to enable independent software vendors (ISVs) and value-added resellers (VARs) to deliver complete business solutions. We will prioritize these on a customer-by-customer basis with frequent releases.

How do I log extensibility requests?

If you discover a customization that you cannot implement as an extension, you must log a request to Microsoft to ensure appropriate extension support is added to the product for your scenario.

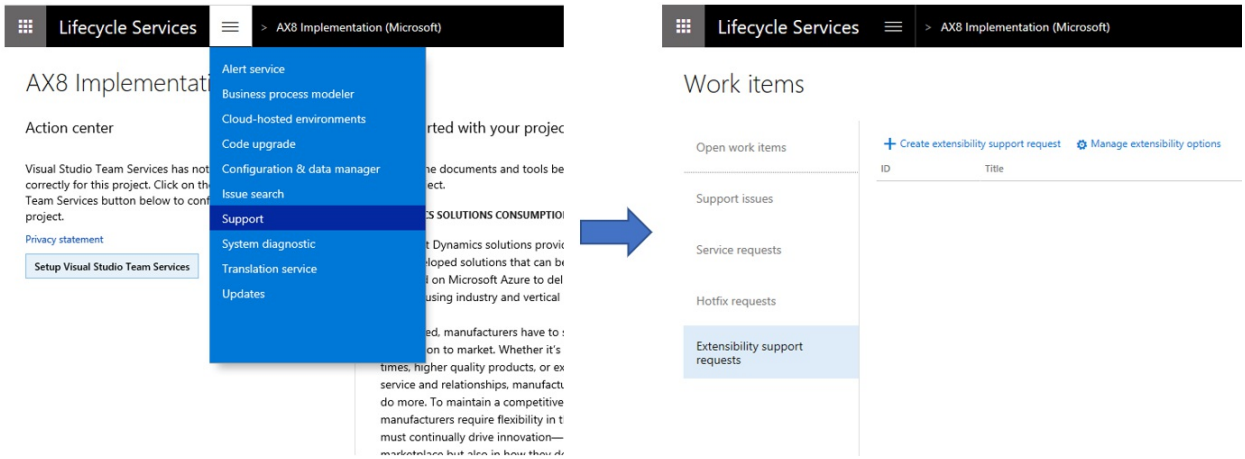
Before logging the request, there are a few things to consider:

- Could the requirement be met with existing extensibility features? Building solutions with extensions requires different design and implementation patterns.
- How important is the requirement to the customer and/or business analyst?
- Will the implementation be upgrade friendly for the long term?

Learn more about extensibility by reading the resources listed on the Extensibility home page and related resources.

Extensibility requests are logged using a specific project in LCS. Logged requests are collected under that same project. We recommend that you log related requests under the same LCS project as this helps maintain a holistic view on all requests for a specific solution or an implementation. Microsoft then further identifies logged requests by the organization name that is associated with the LCS account.

In your LCS project, at the top of the page, select the hamburger icon and then click **Support** menu item.



You can view the list of logged extensibility support requests and their status. Click the request ID to review details of the logged request.

New requests that are logged are briefly assigned a status of **Pending** while the request is copied to the Microsoft tracking databases. Next, an ID is assigned to the request and the status is updated to **Active**. When a request is processed by Microsoft, the status of the request will be updated to **Closed**. Click the request ID to view resolution date and description information. Requests that have been closed are released with monthly application updates.

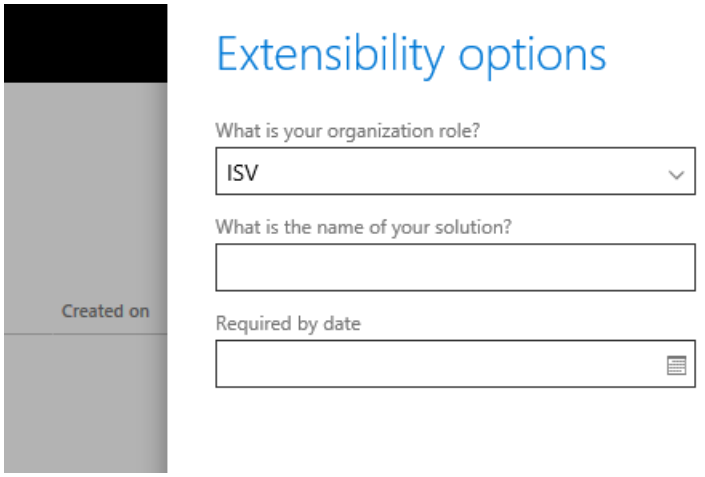
> **NOTE**
>
> At this time, there is no state available to indicate feedback from Microsoft on when requests have been planned.

The **Extensibility support request** form includes two actions:

- **Manage extensibility options**
- **Create extensibility support request**

When you click **Manage extensibility options**, you can view all of the information that is shared between requests. This information includes, requests for either an **ISV** or **VAR** solution and if the requests are specific for a **Customer** implementation project. If the role selected **ISV** or **VAR**, a solution name must be specified. The name should be recognizable and correlate with AppSource solutions. The **Required by date** indicates the last date that requests can be made to be available for your development.

> **IMPORTANT**
>
> Note that Microsoft does not guarantee that all requests will be provided by the date given. However, the required date provides an indication that will be considered when planning for requests at Microsoft.



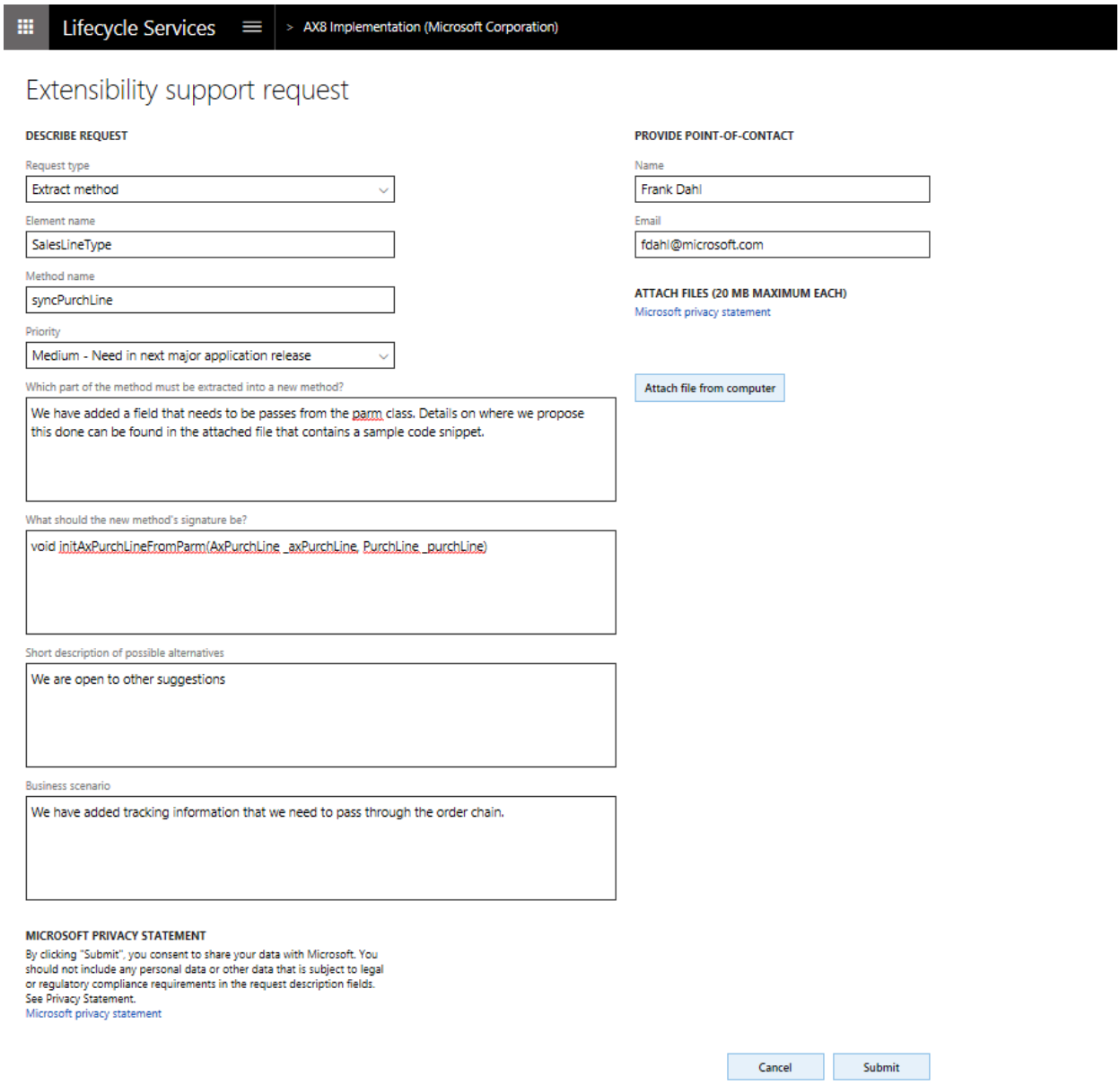
Extensibility support options can be updated to reflect any changes after a request is created. After you have made your updates to the request, click **Update** to notify Microsoft of your changes.

> **NOTE**
> There is currently no option with the tool to record what a Customer implementation includes regarding ISV or VAR solutions.

The action 'Create extensibility support request' is used for creating, or logging, new extensibility requests. When you log an extensibility request, provide detailed information about what you need to become enabled for extensibility, and include information on what it is you need to extend. This will help Microsoft to be efficient in addressing your requests. You are welcome to propose how Microsoft could enable the functionality that you need in the standard application in a way that effectively addresses your needs.

When you select the request type, determine how your request aligns with the request types that Microsoft uses to categorize requests. Each request type changes the form to include specific fields related to the request type. This helps guide the process to make the request actionable for Microsoft. Be sure to provide accurate names when naming elements and methods. Microsoft rarely enables requests by adding inline delegates, so when possible, consider other types of requests. Common application request types include **extract method**, **extensible enum**, **construct with throw**, and **method change**. Additionally, there is **platform request** and **metadata change** for proposing changes, including general platform improvements. The request type **method signature change** is typically for a breaking change. It is unlikely that a breaking change can be accommodated under a monthly update as it will require a more major release version to drive such changes.

Click **Attach file from computer** to upload documents that you can attach to requests. You can use the attachments to supply code snippets that provide additional details for the request. We recommend that you be as specific as possible with your requests.



Log a request for each instance. Do not bundle multiple requests into one. If multiple requests are related, consider adding a document or description that includes request ID's so that any work on the requests is considered in context.

The requests include a point of contact. This is needed for times when the logged extensibility request is not actionable for one reason of another. The requests may require discussion regarding, for example different design options. Microsoft will use this contact information to drive such interactions. Click **Submit** when you are ready to submit the request to Microsoft. Because requests can't be edited after they are submitted, verify the data before you submit. Requests that are accidentally submitted with incomplete or inaccurate date can be removed using the designated action after clicking the ID on the request. Requests that are submitted to Microsoft will temporarily show as **Pending** until the request is created within the Microsoft tracking databases. This will assign an ID to the request and the state will become **Active**. This status update indicates that the request is now visible to Microsoft.

Make sure to read through the privacy statement before you log any requests.

> **NOTE**
>
> We will not release extensibility requests as hotfixes.

Extensibility requests are exclusive for the application. We are not planning to accommodate extensibility requests for Dynamics AX 2012 or earlier releases.

When will my extensibility requests be enabled?

Extensibility requests are logged to a backlog. Microsoft engineers prioritize all requests, and then work on them in priority order. Please note that Microsoft is not ensuring that all requests will be fulfilled. In particular, requests that are intrusive by nature will not be supported, as they will prevent seamless upgrade.

How will extensibility requests be made available to deploy?

After Dynamics 365 for Finance and Operations release 8.0, we plan to release frequent application updates with new extensibility requests. This will follow the same release cadence as platform updates.

Still have questions?

Read the Extensibility FAQ and the other resources listed on the Extensibility home page.

Extensibility FAQ

2/18/2021 • 5 minutes to read • Edit Online

Will source code be available after the hard seal?

Yes, source code will be available after the hard seal. It's required for effective implementation and debugging.

How do I contact Microsoft if I have an extensibility request?

There is a special extensibility request form on the Lifecycle Services (LCS) site.

Where can I ask questions about extensibility patterns?

You can gain access to the Operations Extensibility group in Yammer. Operations Extensibility is an active group that has a significant amount of partner engagement. You get access via the Connect site by signing an NDA.

Where can I find documentation about extensibility patterns?

Documentation about extensibility patterns is available on the Extensibility home page.

Where can I get information about extensibility training?

We will announce training sessions in multiple ways. AppSource partners might receive direct invitations for some sessions. We will also announce workshops in the Operations Extensibility Yammer group and other forums.

What is the goal of sealing the application?

The application is being sealed as a step toward reducing upgrade costs in the ecosystem, so that customers can stay current on new releases. Customers can take advantage of new innovations that come from Microsoft and partners.

Extension packages enable better performance at design time, faster build automation, and unit testing. They also provide more efficient distribution and installation of models from independent software vendors (ISVs) and customers across different systems.

What is Microsoft working on to support this move?

There are several areas where the product team is working to improve the extensibility of the product. This work ranges from platform changes that have broad impact to refactored application code that provides additional hook points. For details, see the Operations Extensibility Yammer group and the product release plans.

After the application is sealed, what should customers do in a critical situation if they must make a quick change?

This scenario is very similar to a scenario where a critical bug fix is required, and the same process should be followed. As a required first step, you must create a case for support.

Can I overlayer an ISV solution after the hard seal of the application

code?

We recommend that ISVs also seal their models. This step helps achieve the broader goal of reducing upgrade costs.

Will I be able to overlayer an on-premises solution?

On-premises solutions will follow the same patterns as cloud solutions. Therefore, no overlayering of Microsoft code will be supported.

How often will Microsoft provide external updates so that partners can see what extensibility enhancements have been made?

We plan to provide monthly updates of platform and application after Microsoft Dynamics 365 for Finance and Operations release 8.0.

Why wasn't my extensibility request accepted?

Some extensibility requests break changes. Some of the more common potentially breaking requests are listed here along with potential workarounds. In addition, read Creating extensions to understand the existing platform extension capabilities and Tips for logging extensibility requests to learn more about how to create solid requests if a capability doesn't exist in the latest release.

Why can't EDT.StringSize be made extensible?

- Request: Make EDT.StringSize changeable via extension.
- Problem: When a table string field (FieldX) is of type "parent EDT" and is associated (through table relations) with another table's field (FieldY) of type EDT2 (EDT2 is derived from "parent EDT"). If FieldY could have a larger string by allowing EDT2.StringSize to increase, FieldX would not be able to handle the new string size.
- Workaround: Create a new EDT and use that for the table field FieldY.

Why can't a unique table index be made extensible?

- Request: Make unique table indexes changeable via extension, for example by allowing an extra field to be added.
- Problem: If a unique table index changes and any data does not conform to the new index, then it would be a breaking change. Also, any query would affect it since it can now retrieve a non-unique record. For example, if a Person table had a key of "Name" and select person where name="Chris" works, but if BirthDate was added to the key, now there could be multiple records returned for "Chris".
- Workaround: Add "soft" constraints in the validateWrite or validateInsert methods.

Why can't CountryRegionCode be made extensible? (*it already is*)

- Request: Make CountryRegionCode changeable via extension.
- Problem: Starting with Platform update 14, changes to CountryRegionCode are supported if the CountryRegionCode property already has a value. Empty CountryRegionCode properties cannot be changed because that change is more restrictive (the element would now only be available for certain countries/regions) and therefore would be a breaking change.
- Workaround: Use the existing CountryRegionCode extension capability when the element is already country/region specific.

Why can't the Table Field properties AllowEdit, AllowEditOnCreate, Mandatory, or IgnoreEDTRelation be made extensible?

- Request: Make Table Field properties AllowEdit, AllowEditOnCreate, Mandatory, and/or IgnoreEDTRelation changeable via extension.
- Problem: The ability to change the "Allow Edit", "Allow Edit On Create", "Mandatory", and "IgnoreEDTRelation"

properties on Table Fields would result in breaking changes. Changing a field to allow editing changes the intent of the field. Not allowing a field to be edited can break existing behavior. Changing a relation breaks the original intent of that relation, which is a breaking change. Making a field mandatory can result in breaking existing behavior.

- Workaround: Add new Table Fields via extension and control those as needed.

Why can't Security Privileges be made extensible?

- Request: Make Security Privilege changeable via extension.
- Problem: The ability to change the Security Privilege would result in breaking changes because this is the lowest level of security metadata.
- Workaround: Create a new Security Privilege if needed and use that.

Why should I avoid calling and extending APIs that are marked with InternalUseOnlyAttribute?

Throughout the application, an effort has been made to avoid breaking changes to APIs made by customers, partners, or ISVs. When a class or method has the **InternalUseOnlyAttribute** applied to it, this means that the API is for internal use only and could change without warning. If customers, partners, or ISVs use or extend an API with **InternalUseOnlyAttribute**, this could create issues because the API could change at any time, which would require changes in their extensions before an update can be applied. This could result in urgent changes and the need to recompile. Developers should not depend on these classes and methods remaining unchanged.

Calls to classes and methods with the **InternalUseOnlyAttribute** will result in compiler warnings. Starting in Platform update 20 to Platform update 24, targeting classes and methods with **InternalUseOnlyAttribute** using Chain of Command will result in compiler errors. In Platform update 25 and later, we plan to continue to issue compiler warnings.

Migrate from overlayering to extensions

2/18/2021 • 10 minutes to read • Edit Online

Introduction

When the application was first released, we strongly recommended that extensions be used instead of overlayering for customization. Overlayering-based customizations have been migrated from release to release through code migrations, and many customizations of application code are still based on the overlayering of code. For most partners, at least some of their solution is still based on overlaying, and some partners will have lots of overlayering across their solutions.

The amount of work that is required to change an implementation from overlayered code to extensions depends on the code itself. Some overlayered code can be changed relatively seamlessly. However, for some changes, you must rethink the customization to find an appropriate way to accomplish it through extension. Therefore, it can be a major undertaking to change complete solutions where multiple places have overlayered code. Such an undertaking requires an investment in the solution. The upside of this investment is a more seamless upgrade process, because customization is now based on application programming interfaces (APIs) through extensions. Additionally, a lengthy code upgrade process is no longer required as it was for overlayered code. More importantly, daily servicing of a running environment offers many benefits. The core application and extensions no longer have to be compiled together, and patching can be done by deploying precompiled assemblies. Therefore, customers can apply patches to their system in a relatively seamless manner, and the amount of downtime is minimized. However, there is work that must be done before this result can be achieved.

Although there are multiple ways to approach this task, we have gained experience through our close work with independent software vendors (ISVs) and value-added resellers (VARs) that have already started to migrate from overlayering to extensions. In this topic, we share some of this experience.

First things first

The task ahead is substantial, and we want to make sure that our shared investment pays dividends. Keep the goal in mind as you work through your customizations. When customization is done correctly, your solution has these qualities:

- It has no intrusive customizations.
- It supports side-by-side deployment with other ISV solutions.
- It's resilient to changes in Microsoft code.
- It's resilient to changes in other ISV solutions.
- It can be upgraded automatically to future versions.

This type of customization represents a fundamental shift of approach. Previously, the primary objective was to implement the functional requirements on the current version. This objective was acceptable, because we knew that manual work was required in order to upgrade the solution. Previously, great engineers minimized the manual upgrade cost. Now, *every* engineer must implement solutions that require *zero effort* to upgrade.

Staying on the right path

Cars are designed to be safe. However, they can't yet prevent accidents. Accident prevention remains the driver's responsibility. Similarly, the development toolset is designed for extensibility. However, the toolset can't yet prevent every type of intrusive customization. As an engineer, it's your responsibility to avoid intrusive customizations.

Sometimes, you might find that you can reach your functional goal only by implementing intrusive customizations. In this case, you should reach out to Microsoft to find a correct solution. You should not force

your way forward. Otherwise, customers who, for example, experience an outage of their service after an automated upgrade might realize that your solution isn't future-proof after all.

Because a future-proof solution represents a competitive advantage, it's worth the extra effort to do it correctly.

Obtain an overview of your code

When you create an overview of your code, first consider analyzing each of your solutions independently instead of analyzing them all together. This approach might be practical even if different teams work on the individual solutions. By choosing one team that you will engage before the other teams, you can gain some experience. Experience is valuable, because it not only helps you analyze and plan the work, but also helps the team ramp up and become familiar with the extensibility model. Therefore, the experience that you and your team gain can become valuable "lessons learned" that you can apply to later solutions.

We have gained practical experience both with ISVs that take each ISV solution in turn, and with ISVs that work as VARs and take customer solutions later.

No matter how the work is pieced together in solutions, you can use the Customization Analysis Report (CAR) to get information about what has been overlayered. This report is generated when you submit your solutions to the Code Migration tool on Microsoft Dynamics Lifecycle Services (LCS). The report is in Microsoft Excel format and includes a list of all the places that have overlayered code. You can use the report to both analyze and categorize all overlayered instances in your solution.

To obtain an overview, you might find it helpful to categorize each overlayered instance. The category that you apply to an overlayered instance should represent the approximate effort that is required in order to change the customization to extensions. Some customizations will be easily changed to extensions. However, for other customizations, the change will be more difficult.

From our experience working with numerous ISVs, we have found that the following categories are a good starting point.

CATEGORY	DESCRIPTION
Extensible enums	You can add new enum values by using extensions. For more information, see Add values to enums through extension.
Construct with throw	Most construct methods are simple and can be extended by using post-event handlers. However, some construct methods are more complex and throw an exception when no class is created.
Exposing members	Member variables that have the **private** access modifier in their definition can't be accessed through extensions unless they become exposed through public methods. You can request that we add access to members through extensions that currently have not been exposed for this. Note that access to protected members is generally enabled through extension classes.
Data manipulation methods that don't raise DataEvents	In some places in the application, data methods such as **insert()** and **update()** don't call **super()**. Therefore, the methods don't raise DataEvents to add extensions to. Microsoft plans to refactor the standard application so that it includes additional methods that enable extensions in these places. If you submit a request for Microsoft to add this, add any of the affected methods that you must currently overlayer, if those methods haven't already been accounted for.

CATEGORY	DESCRIPTION
Extract method	This category is for code changes in the middle of methods, which can't be made through chain of command. When you request a method extraction, be sure to specify which lines of a method to extract, and what the signature of the new method must be.
SQL statement operations	SQL statements that are written directly in the application code don't enable extensions. When you make a request to extend these SQL statements, be sure to explicitly specify what you must extend, such as a field list, **where** clauses, or ordering.
Metadata overlayering	Provide the Application Object Tree (AOT) path of the element where you believe the metadata (property value) must be changed. Metadata changes can't be made through the current extension capabilities.
Method overlayering	This category is for customizations where a method is overlayered. You should consider converting the overlayed method to an extension, so that changes are clean by extension not substitution.
Method signature changes	The capability to change method signatures through overlayering will be discontinued. Other patterns for achieving similar results are required. You can request changes to the standard signature to support extensibility. Be sure to include information about additional parameters that are required.
Inventory dimensions	You can no longer add dimensions by editing the macro and recompiling the standard application. Another approach will be offered that involves predefined dimensions that are deployed at runtime. This approach drives changes to existing customizations where new dimensions are added.
Extensibility platform	Some customizations might not be possible through extensions unless new platform features are added. If you determine that customizations can't currently be done through extensions, open an extensibility request that explains the scenario and what is required.
Reports	Customizations of report designs have limited support for extensibility. In general, a new report must be created. Data provider classes can also be customized so that they include additional information. In some places, the standard application must be changed to enable this type of customization.
Other	This category is for overlayering instances that don't fit into any other category.

By categorizing all overlaying code, you gain an overview of what must be changed.

Analyzing for impact and estimating work

In a typical approach to assessing work impact, you break down tasks into something tangible. This approach also applies to this work. The categories that were discussed in the previous section help frame similar

customizations, and a first pass on estimates can be built by coming up with an overall estimate for each of these categories and similar categories that make up your solution. The group of customizations in a given category often has a few extremes that stand out, and it might be appropriate to establish estimates for these customizations individually.

Consider that some customizations will require either a request to Microsoft to enable extensibility or significant refactoring of the customization so that it can be done through extensibility. Both of these scenarios will increase the estimates for migrating the solution.

Customization that drives what are referred to as intrusive changes is often more complex to convert to extensions. For these changes, you must consider what is the correct way to approach the customization. Here are some examples of these changes:

- Customizations that request inline delegates.
- Customizations of complex classes or methods such as **SalesLinetype**.
- Changes to method signatures.
- Additions of inventory dimensions.
- Changes to report definitions and report data provider classes.
- Intrusive changes to forms.

For changes that require different approaches to make the customizations extension-based, you might have to log requests to Microsoft to enable extensibility. When creating your migration schedule, you'll need to take into account the delay of waiting for updates from Microsoft.

What is supported, and what requires an extensibility request?

When you review a customization, be sure to consider different options for converting it to an extension. Be sure to consider whether a method is hookable, or whether it can be a class extension or form event. Review most of the currently available application code that is available to you.

You might conclude that a change to the standard application is required in order to enable the required extension. In this case, you must log an extensibility request. The request is then put into the backlog at Microsoft so that it can be addressed. Don't log extensibility requests by opening a request for a hotfix, because Microsoft doesn't release extensibility requests as hotfixes.

Be sure to supply enough contextual information in your extensibility requests. For example, a request for an inline delegate might come from the current customization approach. However, to better accommodate extension, the requirement that led to this customization might be better served by a structural change to the standard application. We appreciate suggestions of this type, because they help move the application toward a better platform for building different customizations.

Planning the migration

Be sure to start planning the migration of your solutions early. This planning is important because it helps you make sure that you have room in your schedules to identify and log extensibility requests, and that you have room for the time delay before these requests become available in product releases. Additionally, acknowledge that your developers might have to build new skills, and make sure that you cater to any required learning as part of the migration plan.

Your solution might contain intrusive customizations that aren't easily accommodated through extensions. You should consider whether the business value of these customizations outweighs the effort of building them through extensions. In some cases, partners have decided to discontinue parts of their solutions, because they found that it was impractical to rebuild those parts through extensions, and those parts weren't critical to the solutions.

Some smaller fixes that you're customizing across the application might not be core for your solution, but they are important for the customers that you engage with. In these cases, you must decide whether you prefer to ask Microsoft to implement similar capabilities in the standard application. You can enter an extensibility request for this purpose. For example, if customers want to simplify standard business processes in the system, you might suggest that we add options for disabling steps of the process in the standard application.

Customize model elements through extension

2/18/2021 • 16 minutes to read • Edit Online

In this tutorial, you'll become familiar with the Fleet Management Extension model. This model contains elements that extend the functionality of the Fleet Management application. You can customize model elements by creating *extensions*. Unlike the overlayering capabilities of Microsoft Dynamics AX 2012, extensions don't overlay the baseline model elements. Instead, extensions are compiled as a separate assembly that adds to or customizes the model and the associated business logic. You can extend metadata, for example, by adding a field to a table or adding a control to a form, and also extend or customize business logic by defining event handlers and plug-in classes. You can now author event handlers on several pre-defined events on tables, forms, form data sources, form controls, and others. Plug-ins are also a new extensibility concept that enables replacing or extending the business logic of the application.

Prerequisites

This tutorial requires you to access the environment using Remote Desktop, and that you are provisioned as an administrator on the instance.

Understanding the Fleet Management model

The Fleet Management application provides a rental car company a system for managing vehicles, customers, and vehicle reservations. The application is designed for use by the Fleet Clerk and Fleet Manager personas.

Fleet Clerk

The Clerk is the front desk employee who handles the face-to-face and over-the-phone interactions with customers. The Clerk is primarily concerned with entering customer information into the application, creating vehicle reservations for customers, upselling the reservation by offering vehicle accessories, and processing vehicle returns upon completion of a vehicle rental. The Clerk spends the vast majority of their time using the **Fleet Management Workspace** to prepare for interactions with customers by anticipating their needs and providing a pleasant and memorable experience, while interacting with the customer.

Fleet Manager

The Manager is the back office employee who handles setting business requirements and processes. The Manager is primarily concerned with entering vehicle information, defining the available vehicle accessories, vehicle maintenance, determining pricing, and analyzing business performance measures such as revenue, upsell success, and so on. The application's business logic revolves around the following three primary entities and the relationships between them.

Customers

Customers contact the Fleet Clerk to make vehicle reservations, choose vehicle accessories, check out and return vehicles, and pay for vehicle rentals. Customer-related information is stored in the table named **FMCustomer**.

Vehicles

Vehicles vary primarily in their price, which is proportional to the vehicle *class*. The names of tables that store information about vehicles begin with **FMVehicle**.

Reservations and rentals

Reservations handle the relationship between customers and vehicles. Reservation information includes reservation dates, customer information, vehicle selection and price, and additional charges such as accessories or fees. Reservation and rental information is stored in the **FMRental** and **FMRentalCharge** tables. A

calculation engine handles the transactional information related to the pricing of vehicle reservations. Using this data model the Fleet Management application provides a basic car rental experience.

Extending the Fleet Management model

The basic Fleet Management application has been customized with additional capabilities that enable a rental car company to provide pricing incentives to its customers through discounts. The additional business logic and data that enables these discount capabilities is stored in the Fleet Management Extension model. The discount capabilities add value to the Fleet management application through three primary customizations.

The Fleet Management Extension data model

Two new tables have been added that store discount-related information. **FEDiscounts** stores the list of all discounts and their rates. **FERentalDiscountRelationTable** keeps track of the reservations that the discounts are applied to. Existing tables have been extended to account for the addition of discounts to the pricing scheme. The table that keeps track of the vehicle rate for a particular reservation, named **FMRental**, has been extended to accommodate discounts to the vehicle rate. The table that keeps track of the accessories for a reservation, named **FMRentalCharge**, has been extended to accommodate discounts applied to accessories.

The Fleet Management Extension Calculation Engine

The basic calculation engine has been customized to add the various pricing schemes defined by the new discounts. A plug-in class has replaced the functionality of the base calculation engine. When a vehicle is reserved for more than 7 days, the vehicle Fleet Management model calculates savings based on the difference between a vehicle's daily rate and a lower weekly rate. The plug-in removes the weekly rate calculation because this same behavior can be accomplished by using discounts.

The Fleet Management User Interface Extensions

The Rental, which is contained by the form named **FMRental**, has been extended to enable the Clerk to apply discounts to a reservation. The on-screen price summary is updated in real time with savings information related to discounts that can be applied to vehicles and accessories related to the reservation. In the following steps, you'll explore the customizations that have been made in the Fleet management Extension model, as well as re-implement a portion of the customizations for yourself.

Setup

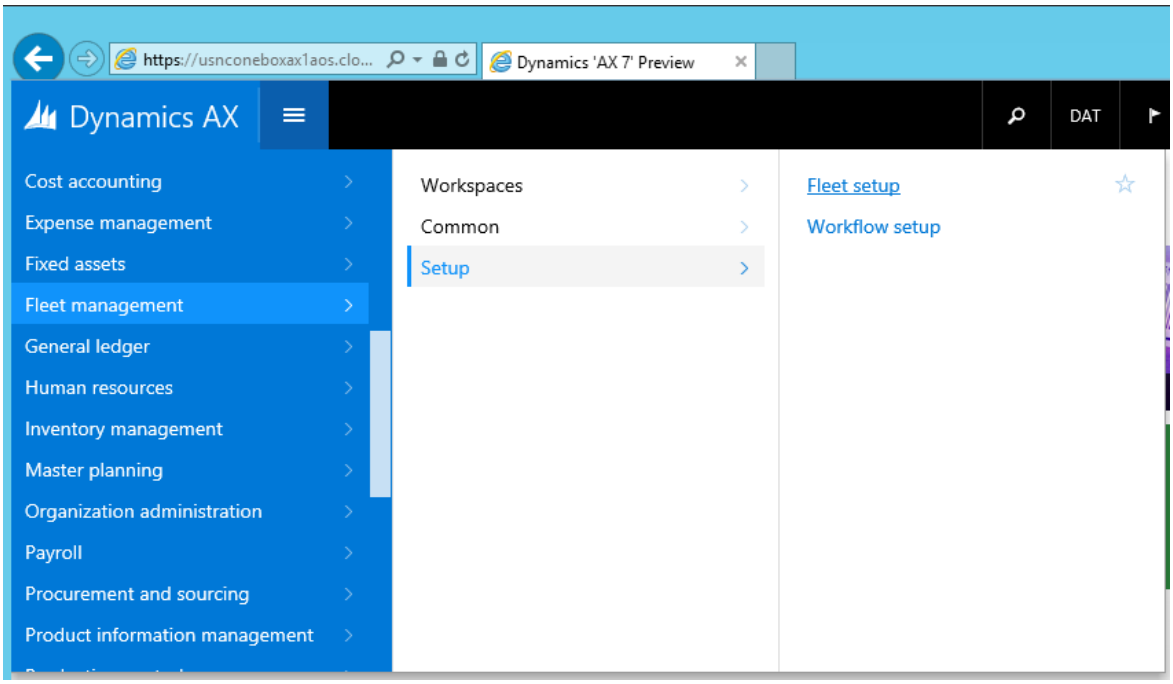
If you haven't opened the Fleet Management Solution in a previous tutorial, follow these steps. The fleet management solution file is available on the Dynamics AX downloadable VM.

1. On the **Desktop**, double-click the **Visual Studio** shortcut to open the development environment.
2. Open the **FleetManagement** solution. On the **File** menu, point to **Open**, and then select **Project/Solution**.
3. Browse to the desktop and open the **FleetManagement** folder. If the solution file is not on your computer, the steps to create it are listed in Tutorial: Create a Fleet Management solution file out of the Fleet Management models in the AOT.
4. Select the solution file named **FleetManagement**. The file type listed is Microsoft Visual Studio Solution.
5. Select **Open**. The solution may take some time to open.

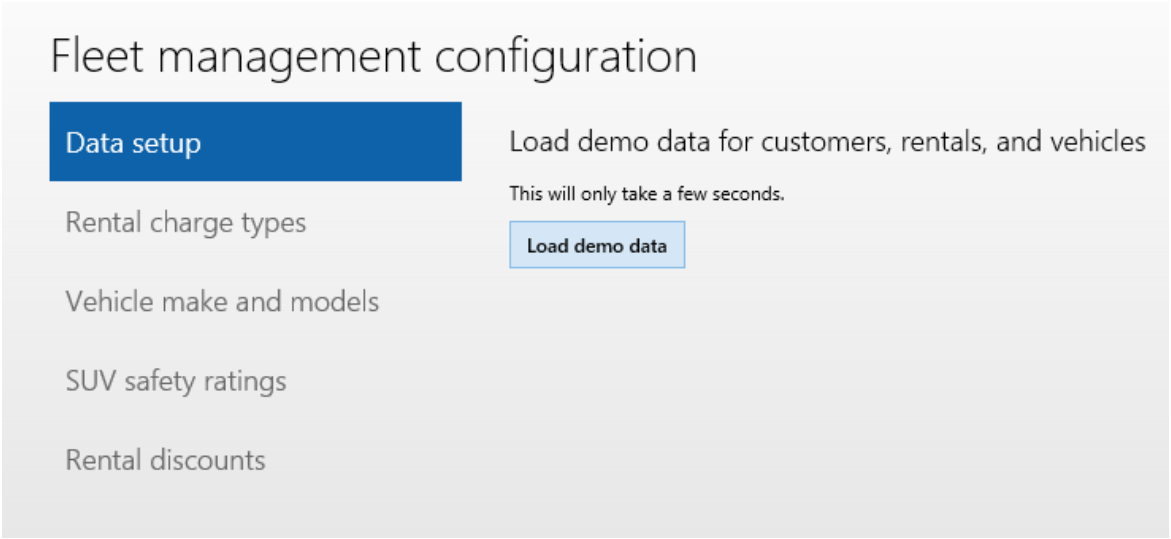
Installing the demo data

If you've already installed the demo data, you can skip to the next section.

1. In the VM, open Internet Explorer and navigate to the application's base URL.
2. Sign in.
3. On the dashboard, open the navigation pane and navigate to **Fleet Management > Setup > Fleet Setup**.



4. Click **Setup Demo Data**.

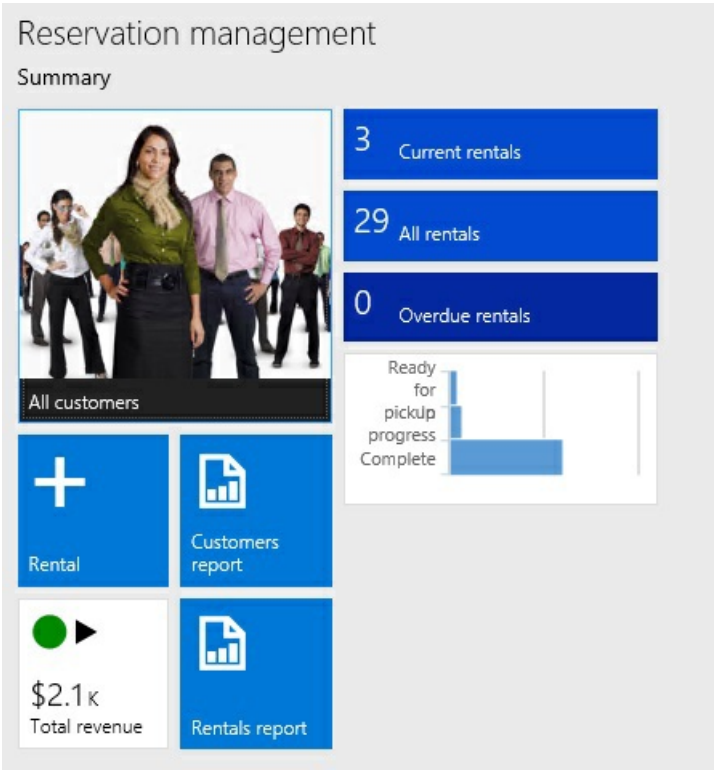


5. If you're prompted to reload the demo data, select **Yes**.
6. When the data is finished loading, select **Close**.
7. On the dashboard, open the navigation bar and navigate to **System Administration > Common > Maintain aggregate measurements**. (Steps 7 to 9 are not applicable on newer releases.)
8. Select **FMAggregateMeasurements**, and on the Action Pane, select **Refresh now**.
9. Wait until the processing completes. The ongoing processing is indicated at the top of the page by a series of moving dots. The processing is completed when the indicator disappears and the **Time Last Processed** field is updated.

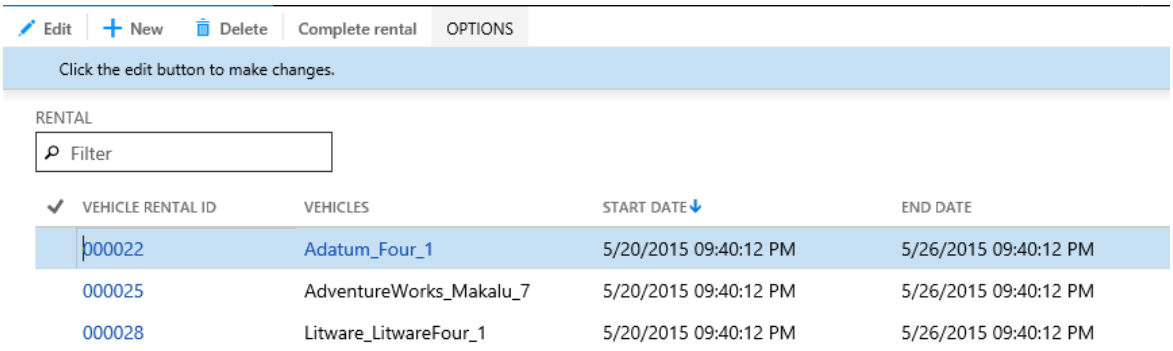
Open the FMRental form on the one-box environment

1. In the VM, open Internet Explorer and navigate to the base URL of your Dynamics AX application. For more information, see Deploy and access development environments.
2. Sign in, if prompted.
3. Find the **Reservation Management** tile and select it to open the Reservation Management workspace.

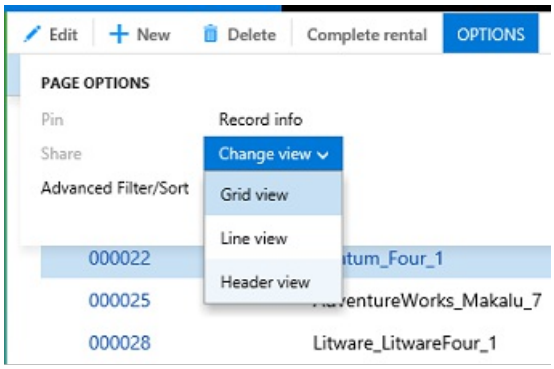
4. When the **Reservation Management** workspace opens, select **Current rentals**.



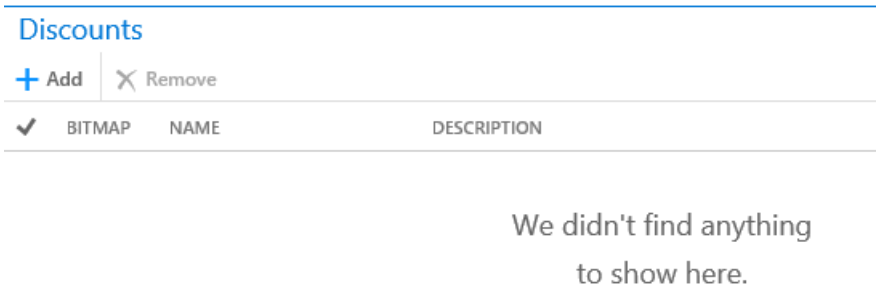
5. The **Rental** form opens in grid view.



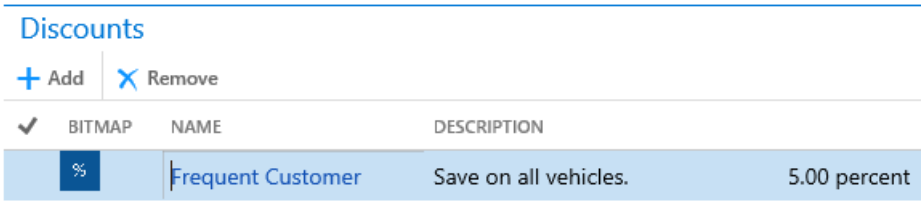
6. After the **Rental** form loads, select **Options** > **Change view** > **Header** to open the **Header view**.



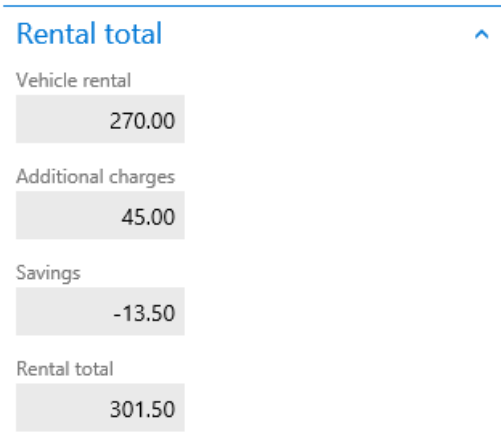
7. When the **Header view** form loads, scroll to the bottom and expand the **Discounts** tab. This tab isn't part of the Fleet Management model. It has been modeled in the Fleet Management Extension Model as an extension to the **FMRental** form.



8. Select **Add** to add a discount.
9. Select the **Frequent Customer** discount, and then select **OK**. The selected discount is added to the **Discounts** grid.



10. Use the shortcut key, **Alt+F2** to open the FactBox.
11. Expand the **Rental total** FactBox on the right and view the discount savings that are applied.

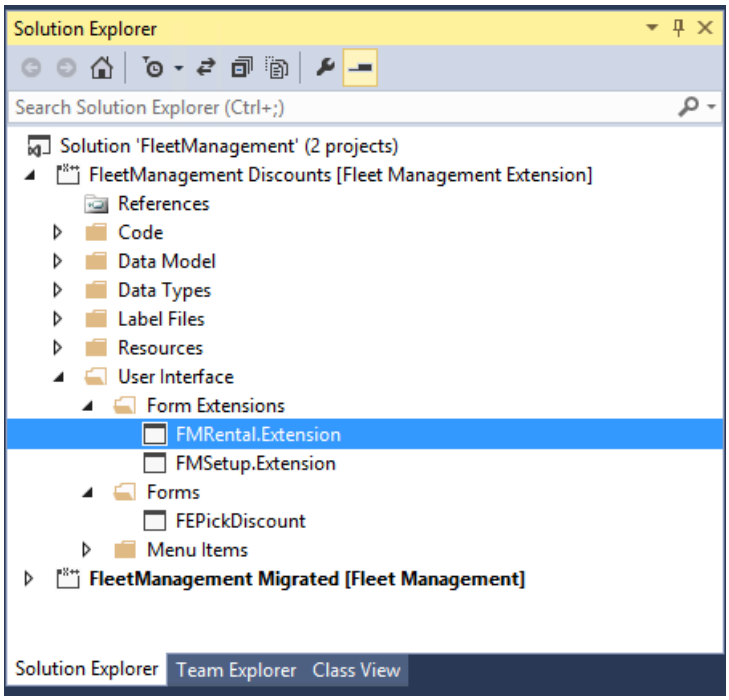


Overview of the Fleet management discount extension project

In this tutorial, the **FleetManagementDiscounts** Project contains the model elements that belong to the model named **Fleet Management Extension**. Here, you'll explore and learn about the project elements.

Navigate to FMRental.Extension in the Tree Designer

1. In the Visual Studio, in **Solution Explorer**, in the **FleetManagement Discounts** project, expand **User Interface > Form Extensions**.

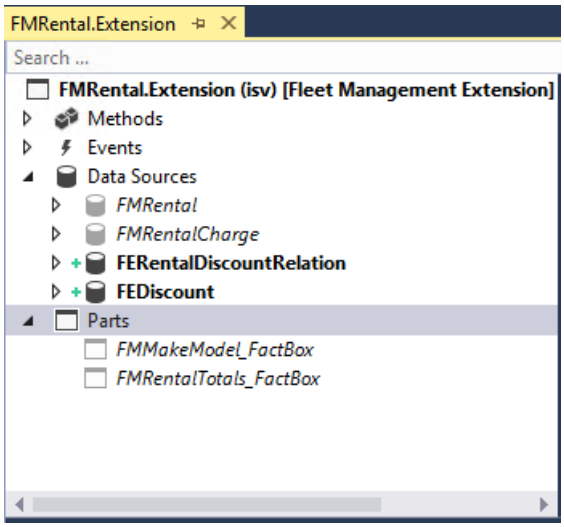


   The **FMRental.Extension** element is an extension element that extends the functionality of the **FMRental** form by adding two new data sources and a new tab control.

2. In **Solution Explorer**, double-click **FMRental.Extension** to open the designer. As the following image shows:

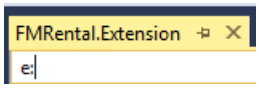
   - The data sources shown in *italic* text are data sources defined in the baseline form.
   - The data sources shown in **bold** are the ones defined in the current extension.

   The designer presents an integrated view of the model element, including its extensions. Read-only nodes are shown in italic text, while nodes that belong to the current extension are shown in bold, with other visual cues that indicate the type of customization.

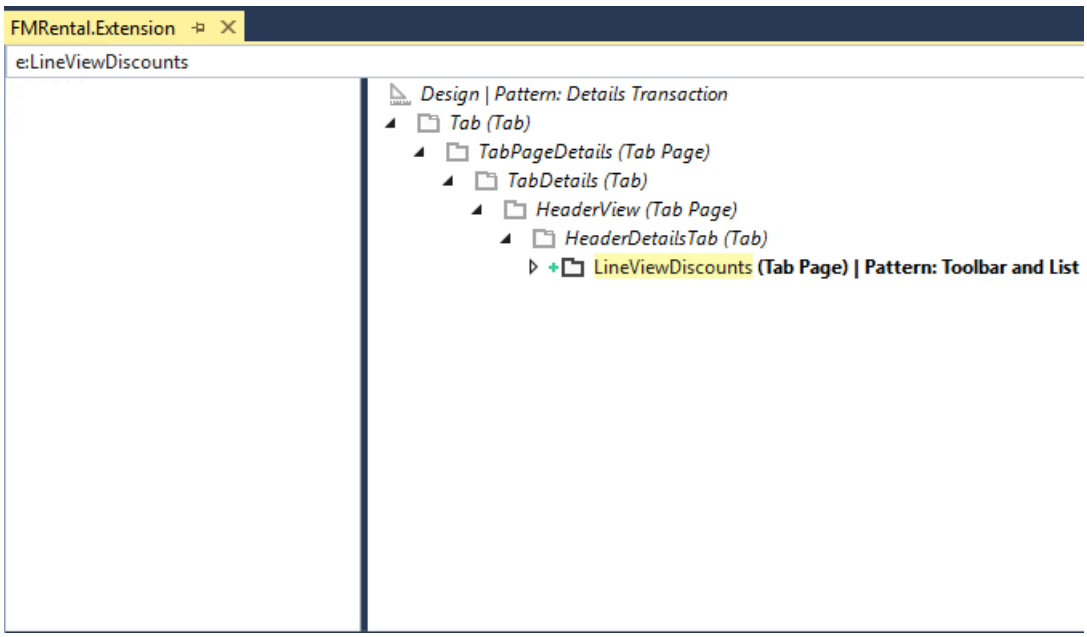


3. In the designer's search box, type 'e:' as shown in the image below. This filters the current designer to

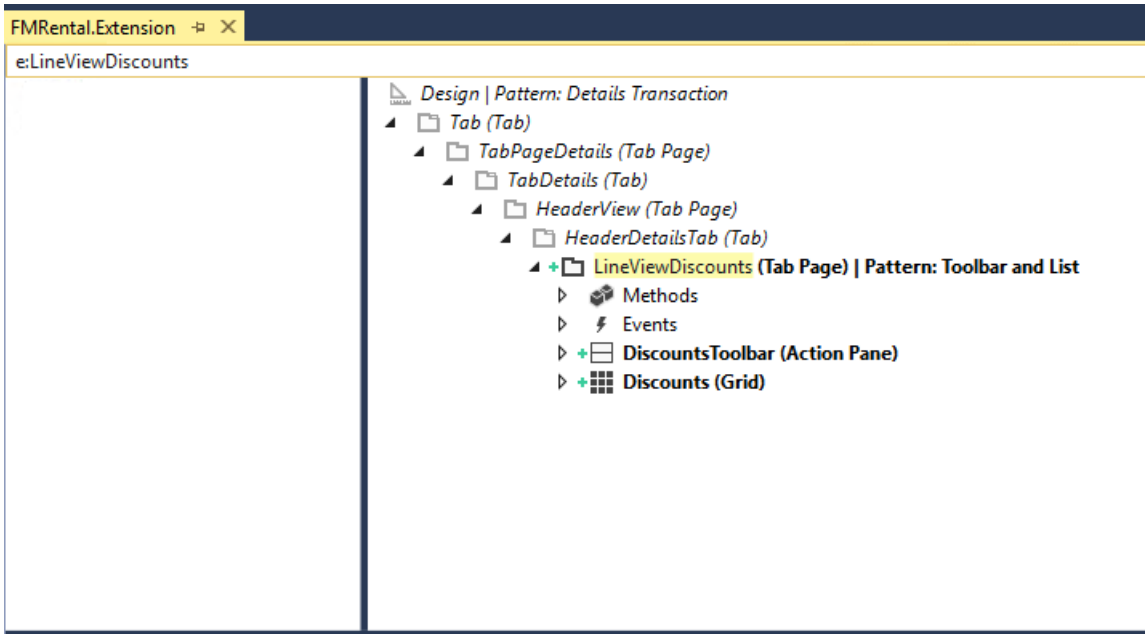
only show nodes that belong to the current extension.



4. You can also type 'e:LineViewDiscounts' to filter the designer to show nodes that match the name **LineViewDiscounts** and that belong to the current extension.

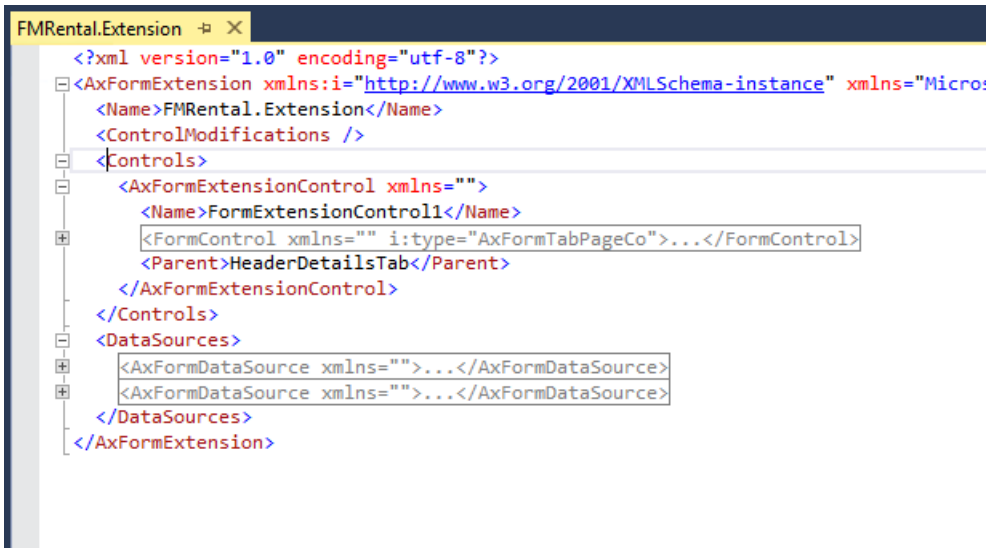


5. Expand the **LineViewDiscounts** node to see its contents.



Open the FMRental.Extension XML file to view the metadata

1. In the **Solution Explorer**, right-click FMRental.Extension form extension, and then click **Open with**.
2. In the **Open with** dialog box, select **XML (Text) Editor**, and then click **OK**.
3. When prompted to close the designer, click **Yes**.
4. Click the corresponding minus signs to collapse the child nodes of the **Controls** and **DataSources** nodes. Refer to the following image for the correct result.



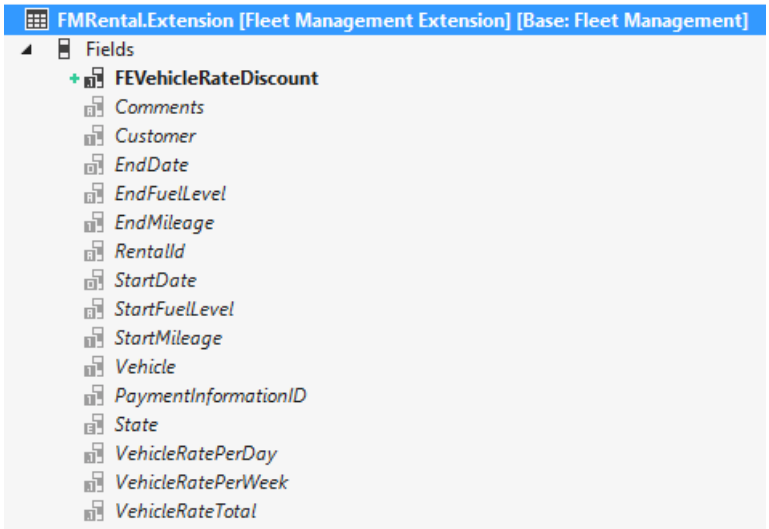
The XML file contains the metadata associated with the **FMRental.Extension** element. You can see that this file contains metadata that describes only one tab page control and two data sources that are part of the extension. You can also see that it doesn't contain any metadata from the base form.

View other elements in the Fleet Management discount extension project

The **FleetManagement Discounts** project contains two new tables, **FEDiscount** and **FERentalDiscountRelationTable**, and two extensions to existing Fleet Management tables, **FMRental** and **FMRentalCharge**.

1. In **Solution Explorer**, in FleetManagement Discounts, double-click **Data Model > Table Extensions > FMRental.Extension** to open the designer.

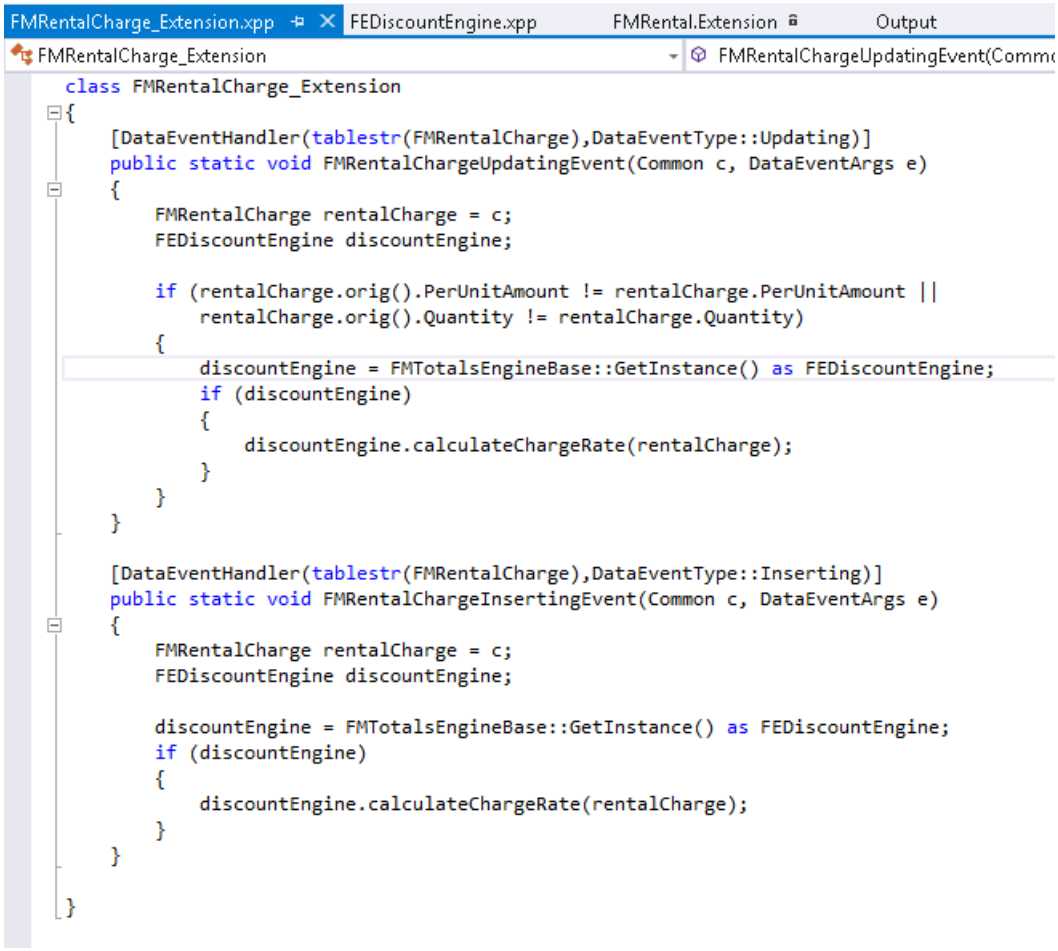
2. Expand the **Fields** node to see that this extension contains one added field, FEVehicleRateDiscount, to the base FMRental table.



3. Similarly, open the **FMRentalChange.Extension** element in the designer to explore its contents.

Inspect the data event handlers

In **Solution Explorer**, in the FleetManagement Discounts project, double-click **Code > Classes > FMRentalCharge_Extension** to open the code editor.



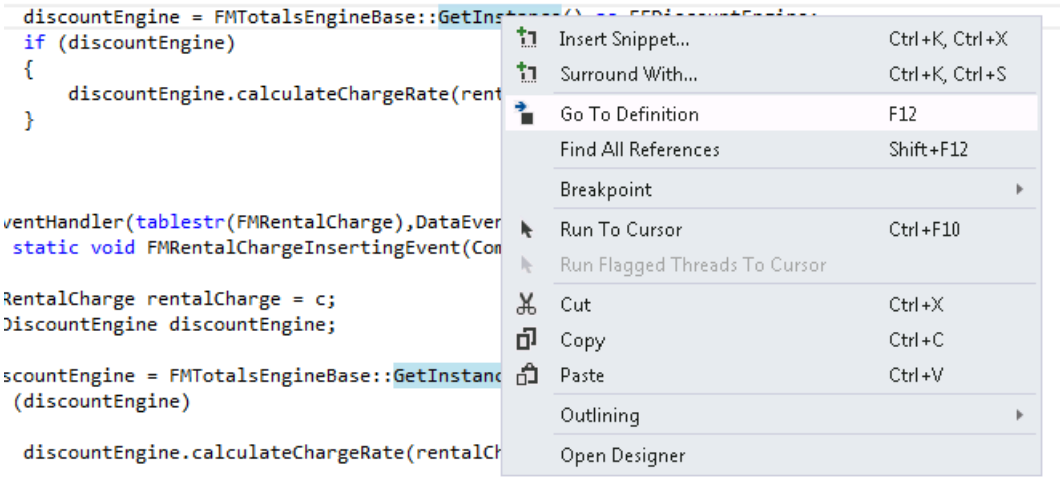
This class contains event handler implementations that subscribe to the **Updating** and **Inserting** events of the **FMRentalCharge** table. Microsoft Dynamics AX introduces data events that can occur on tables and other types. You can subscribe to data events of a table, enabling your application to extend business logic without overlayering base X++ code. Later in this tutorial, you'll see how easy it is to subscribe to table events.

> **NOTE**
>
> Notice that this class is an extension class (indicated by the _Extension suffix). You can author event handlers in any class, this class does not need to be an extension class. Extension classes are needed in order to create extension methods. For more details on extension methods, refer to the "Extension methods" section of the X++ debugger features article.

View the plug-in classes

In the event handler code of the **FMRentalCharge_Extension** class shown in the previous section, notice that both event handlers call **FMTotalsEngineBase::GetInstance** to retrieve the current instance of the Fleet Management calculation engine. The calculation engines are implemented by using plug-in classes. A class factory creates the appropriate instances of a plug-in class based on configuration or business data.

1. In the code editor window that displays FMRentalCharge_Extension.xpp, right-click **GetInstance**, and then select **Go To Definition**. The code editor opens with the abstract class **FMTotalsEngineBase**. This abstract class is called a **plugin point** and it's associated with the following attribute: [Microsoft.Dynamics.AX.Platform.ExportInterfaceAttribute()]



Plug-in classes represent extensions or implementations of abstract classes or interfaces. Plug-in classes are associated with attributes defining their metadata and the plug-in point. In this example, there are two plug-in classes associated with the **FMTotalsEngineBase** plug-in point. The base calculation engine is defined by the plug-in class **FMTotalsEngine**. You can find it in the project **FleetManagement Migrated > Code > Classes**.

```
[System.ComponentModel.Composition.ExportMetadataAttribute("TotalsEngine", "Default"),
System.ComponentModel.Composition.ExportAttribute("Dynamics.AX.Application.FMTotalsEngineBase")]
class FMTotalsEngine extends FMTotalsEngineBase
{
```

The discount calculation engine is defined by the plug-in class **FEDiscountEngine**. You can find it in the project **FleetManagement Discounts > Code > Classes**.

```
[System.ComponentModel.Composition.ExportMetadataAttribute("TotalsEngine", "Discount"),
System.ComponentModel.Composition.ExportAttribute("Dynamics.AX.Application.FMTotalsEngineBase")]
class FEDiscountEngine extends FMTotalsEngine
{
```

2. Look at the **GetInstance** method. It uses the plug-in factory **SysPluginFactory::Instance** to instantiate the current calculation engine based on current plug-in metadata. The plug-in metadata is specified in the global configuration table, **FMParameters**.

```
// the {key,value} pair from the SetManagedValue call should match the {key,value} pair from the attribute on the plugin class
meta.SetManagedValue("TotalsEngine", FMParam.TotalsEngine);

_instance = SysPluginFactory::Instance("Dynamics.AX.Application", classStr(FMTotalsEngineBase), meta) as FMTotalsEngineBase;
if (!_instance)
{
    warning('SysPluginFactory could not initialize totals engine - using default. Consider running fleet setup.');
    _instance = FMTotalsEngine::construct();
}
```

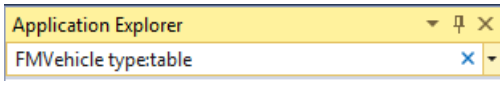
The Finance and Operations apps also support configurable plug-in classes where the plug-in metadata associate with the class isn't known at development time and is configurable at runtime by an administrator. This tutorial doesn't cover that feature.

Create additional Fleet Management extensions

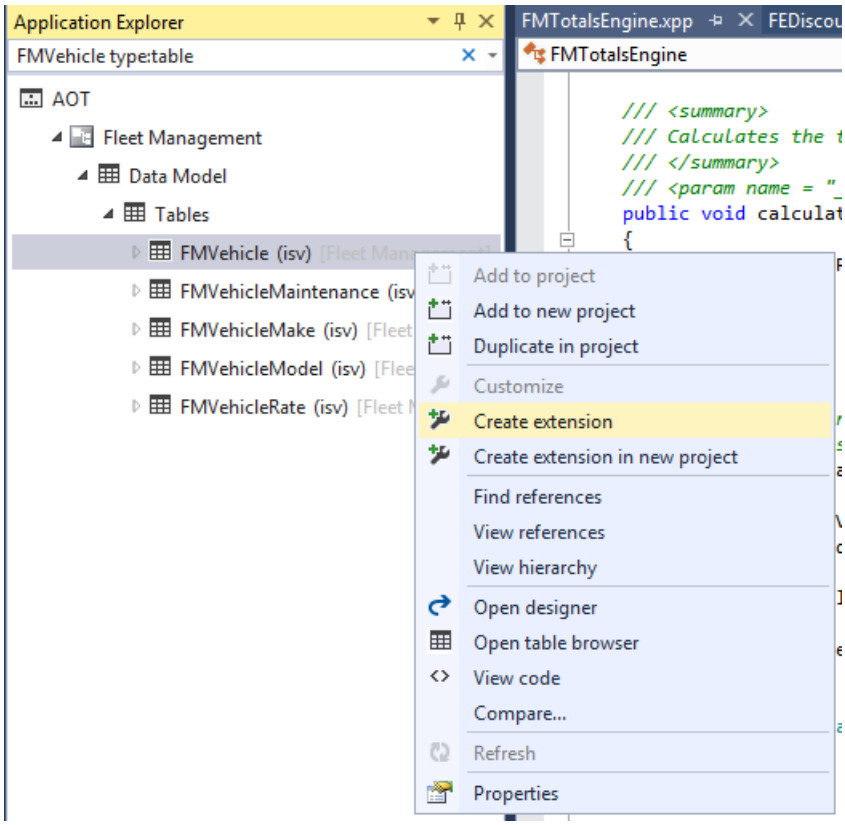
This section shows how you can use the Visual Studio tools to create and interact with extensions.

Extend the FMVehicle Table

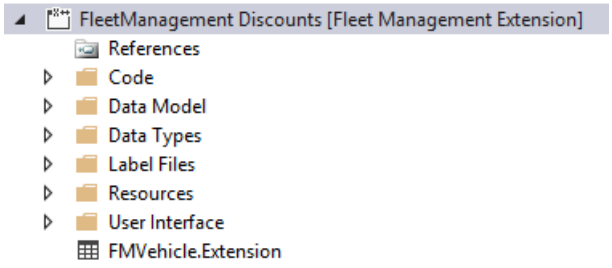
1. In **Solution Explorer**, select the **FleetManagement Discounts** project.
2. In Visual studio, in **Application Explorer,** select **View > Application Explorer**, and search for the table named FMVehicle. Type `FMVehicle type:Table` in the filter bar and press **Enter**.



3. Right-click **FMVehicle**, and then select **Create extension**.



An extension of the **FMVehicle** table is created in the **FleetManagement Discounts** project named **FMVehicle.Extension**.

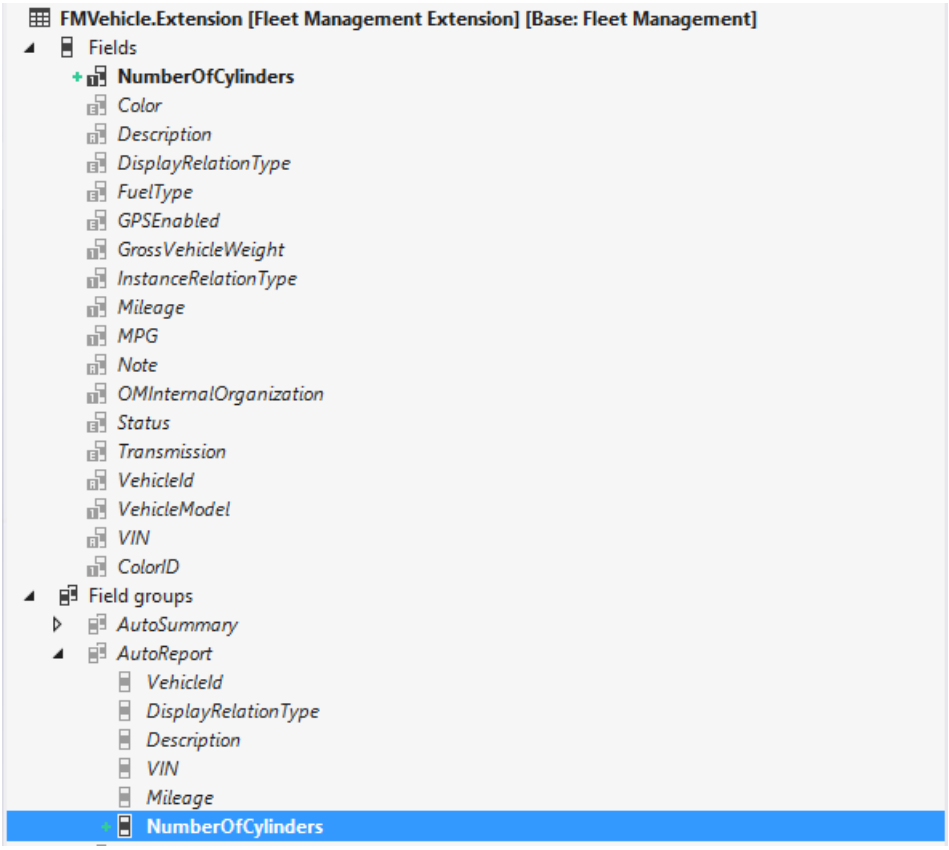


4. In **Solution Explorer**, right-click **FMVehicle.Extension**, and then select **Open with**. In the dialog box, select **XML (Text) Editor**, and then select **OK**. **Note**: This extension file is simply a template that doesn't contain metadata from the base **FMVehicle** table. An extension file will always contain only the metadata that defines the extension and nothing from the base model element.

```
<?xml version="1.0" encoding="utf-8"?>
<AxTableExtension xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
  <Name>FMVehicle.Extension</Name>
  <FieldGroupExtensions />
  <FieldGroups />
  <Fields />
  <Indexes />
  <Relations />
</AxTableExtension>
```

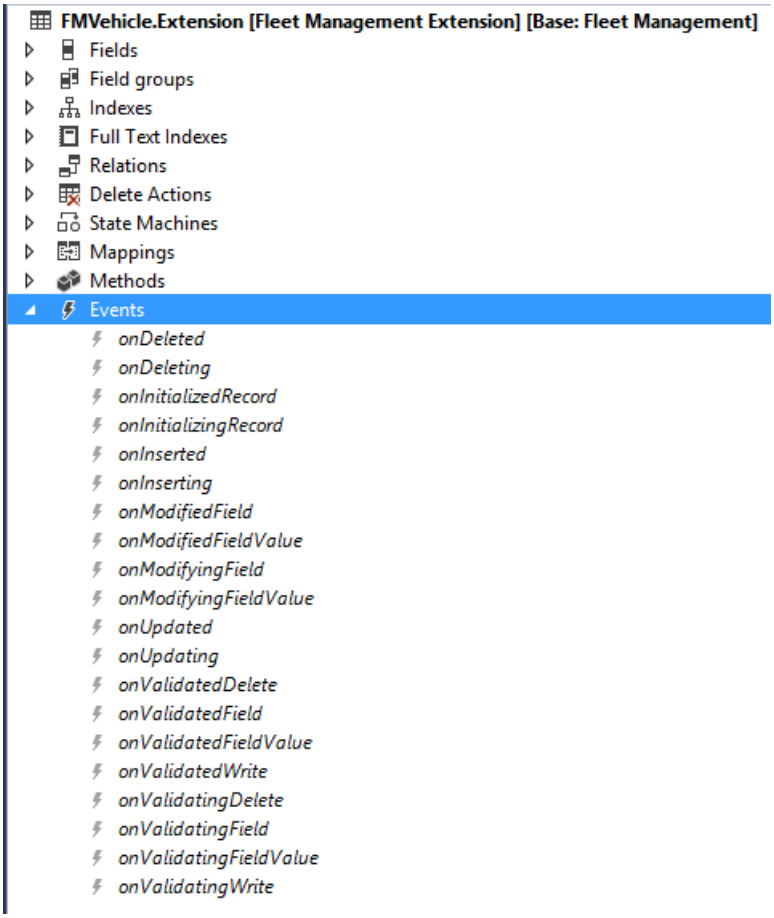
5. Close the XML editor.
6. In **Solution Explorer**, double-click **FMVehicle.Extension** to open the designer.
7. Right-click **Fields** and add a new integer field. Change the name of the field to **NumberOfCylinders**.
8. In the **Properties** window, set the **Label** property of the new field to **NumberofCylinders**.
9. Drag-and-drop the **NumberOfCylinders** field into the **AutoReport** field group to extend the field

group of the base table.



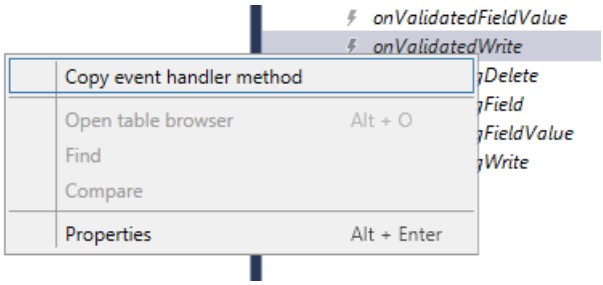
10. Save FMVehicle.Extension.

11. Expand the **Events** node. The **Events** node lists all events that the table exposes. This list includes events that are defined by the framework, and delegate methods that are defined by application developers.



> **NOTE**
>
> Different framework events are exposed on the designers of many types of element and sub-elements, like table events, form events, form data source events, and form control events.

12. Right-click onValidatedWrite, and then select **Copy event handler method**.



This step copies the event handler method signature to the clipboard.

13. Add a new class named **FMVehicleEventHandlers** to the **FleetManagement Discounts** project.
14. In **Solution Explorer**, double-click **FEVehicleEventHandlers** to open the code editor.
15. Right-click and paste the event handler method that you copied in step 12.

```
Class FMVehicleEventHandlers
{
    /// <summary>
    ///
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    [DataEventHandler(tableStr(FMVehicle), DataEventType::ValidatedWrite)]
    public static void FMVehicle_onValidatedWrite(Common sender, DataEventArgs e)
    {
    }
}
```

16. Insert the following code into the **FMVehicle_onValidatedWrite** event handler. This code validates that the number of cylinders can't be greater than 8.

```
[DataEventHandler(tableStr(FMVehicle), DataEventType::ValidatedWrite)]
public static void FMVehicle_onValidatedWrite(Common sender, DataEventArgs e)
    {
        ValidateEventArgs validateArgs = e as ValidateEventArgs;
        FMVehicle vehicle = sender as FMVehicle;
        boolean result = validateArgs.parmValidateResult();

        if (vehicle.NumberOfCylinders > 8)
        {
            result = checkFailed("Invalid number of cylinders.");
            validateArgs.parmValidateResult(result);
        }
    }
```

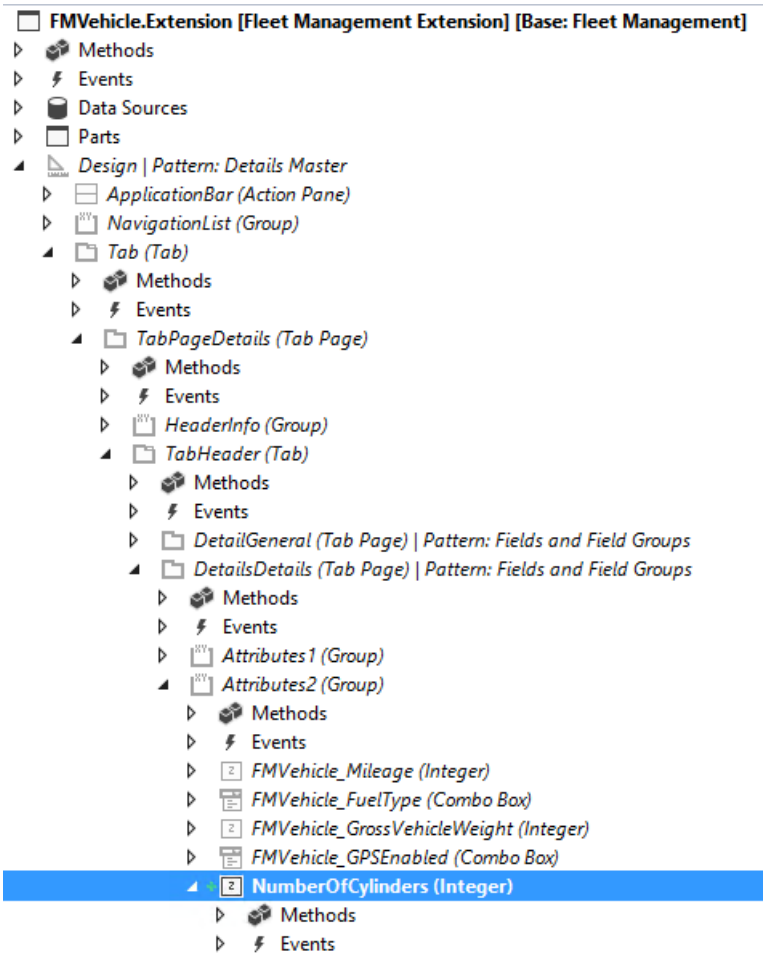
17. Save FMVehicleEventHandlers class

> **TIP**
>
> You can paste and define your event handlers in any class of your model. The class FMVehicleEventHandlers is used only as an example.

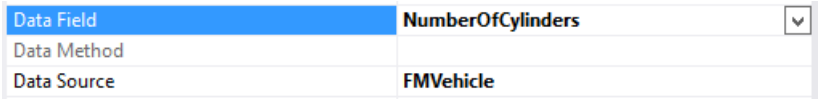
Extend the FMVehicle Form

Next, add an extension to the **FMVehicle** form in the **FleetManagement Discounts** project. First, be sure to select this project in **Solution Explorer**.

1. Use **Application Explorer** to find the form named **FMVehicle**, and in the **Application Explorer** filter bar, enter `FMVehicle type:form`.
2. Right-click the form, and then click **Create extension**.
3. Add a new integer control named **NumberOfCylinders** to the **Attributes2** group control as shown below. You can find this control by expanding **Design** > **Tab** > **TabPageDetails** > **TabHeader** > **DetailsDetails** > **Attributes2**.



4. Bind the new control to the **NumberOfCylinders** data field in the properties window as follows.

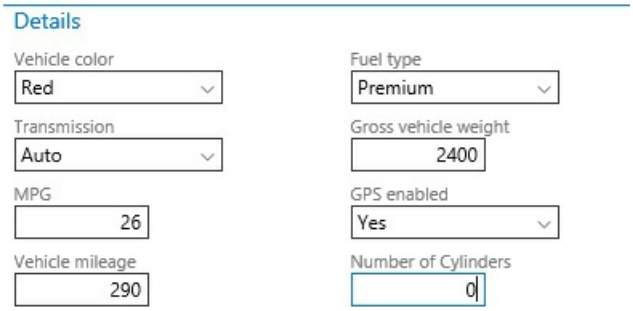


5. Save FMVehicle.Extension and build the project.

Test your extensions

1. In **Solution Explorer**, right-click **FleetManagement Discounts**, and then click **Set as StartUp project**.

2. Similarly, in FleetManagement Discounts, set the **FMVehicle.Extension** form as the startup object.
3. Press **Ctrl+F5** to start without debugging, or use the **Debug** menu.
4. After the **Vehicles** form opens, select a vehicle to view its details.
5. Expand the **Details** tab and notice the new **Number of Cylinders** field.

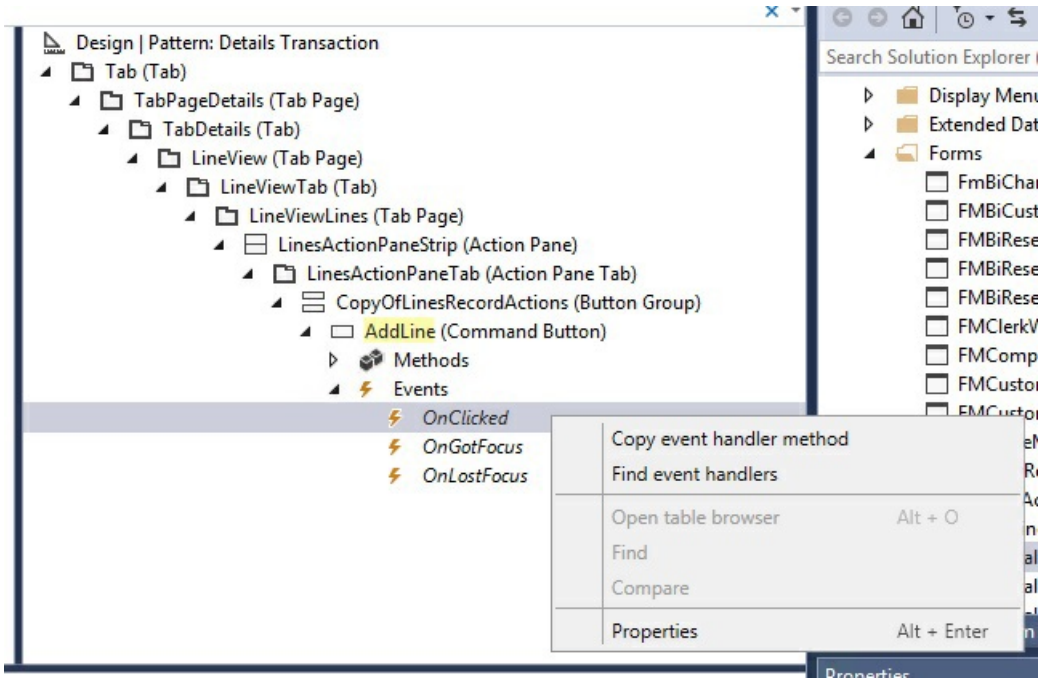


6. In the Action Pane, click **Edit**, and change the value in the **Number of cylinders** field to 12.
7. In the Action Pane, click **Save**.
8. Notice the validation error.
9. Enter a valid number of cylinders, less than 9, and then save the new value.

Experiment with event handlers on form controls

You can add event handler methods on existing controls.

1. Find the **AddLine** command button control in the **FMRental** form designer, right-click the **OnClicked** event, and select **Copy event handler method**.



2. Paste the event handler method in a class of the Fleet Management Extension model and add X++ code to implement it.

```
/// <summary>
///
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
[FormControlEventHandler(formControlStr(FMRental, AddLine), FormControlEventType::Clicked)]
public static void AddLine_OnClicked(FormControl sender, FormControlEventArgs e)
{
}
```

When implementing the AddLine_OnClicked event handler, you can access the button control instance using the **sender** parameter.

```
FormButtonControl button = sender as FormButtonControl;
```

If you need to access the parent form or any of its variables, this example shows how to access the **FormRun** instance and one of its data sources.

```
FormRun fr;
fr = sender.formRun();
var frDs = fr.dataSource("FMRental");
```

Experiment with event handlers on form data sources

Just like tables, form controls and other element types, form data sources and form data source fields provide framework-level events. The following example shows how you can use the ValidatingWrite event on a form data source or the Validating event on a form data source field to validate user input on the FMRental form. This functionality is available as of Platform Update 7.

```
/// <summary>
/// When saving a new rental, prevent setting the start mileage on the FMRental form to a value that is
equal to 1
/// </summary>
[FormDataSourceEventHandler(formDataSourceStr(FMRental, FMRental),
FormDataSourceEventType::ValidatingWrite)]
public static void FMRental_OnValidatingWrite(FormDataSource sender, FormDataSourceEventArgs e)
{
    var datasource = sender as FormDataSource;
    var args = e as FormDataSourceCancelEventArgs;
    if (args != null && datasource != null)
    {
        FMRental record = datasource.cursor() as FMRental;
        if (record.recId == 0)
        {
            if(record.startmileage == 1)
            {
                boolean doCancel = !checkFailed("Start Mileage = 1 is not allowed");
                args.cancel(doCancel);
            }
        }
    }
}
```

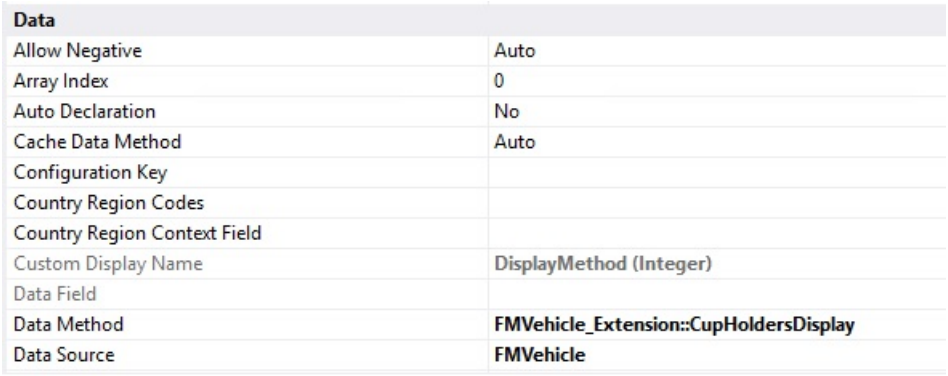
```
/// <summary>
/// Prevent changing the start mileage field on the FMRental form to a value that is equal to 1
/// </summary>
[FormDataFieldEventHandler(formDataFieldStr(FMRental, FMRental, StartMileage),
FormDataFieldEventType::Validating)]
public static void StartMileage_OnValidating(FormDataObject sender, FormDataFieldEventArgs e)
{
    var dataObject = sender as FormDataObject;
    var args = e as FormDataFieldCancelEventArgs;
    if (args != null && dataObject != null)
    {
        var datasource = dataObject.datasource() as FormDataSource;
        if (datasource != null)
        {
            FMRental record = datasource.cursor() as FMRental;
            if (record.RecId > 0)
            {
                if (record.StartMileage == 1 )
                {
                    boolean doCancel = !checkFailed("Start Mileage = 1 is not allowed");
                    args.cancel(doCancel);
                }
            }
        }
    }
}
```

Experiment with table extension display and edit methods

Extension methods enable you to extend tables by creating new display and edit methods on these tables without over-layering X++ code (Extension method must belong to a class named with an _Extension suffix). For example, this class shows how you can extend the FMVehicle table with an extension display method named CupHoldersDisplay.

```
public static class FMVehicle_Extension
{
    public static display int CupHoldersDisplay(FMVehicle vehicle)
    {
        return 7;
    }
}
```

On a form or form extension, you can bind a control to this display method by setting "Data Source = FMVehicle" and "Data method = "FMVehicle_Extension::CupHoldersDisplay" as the image below shows.



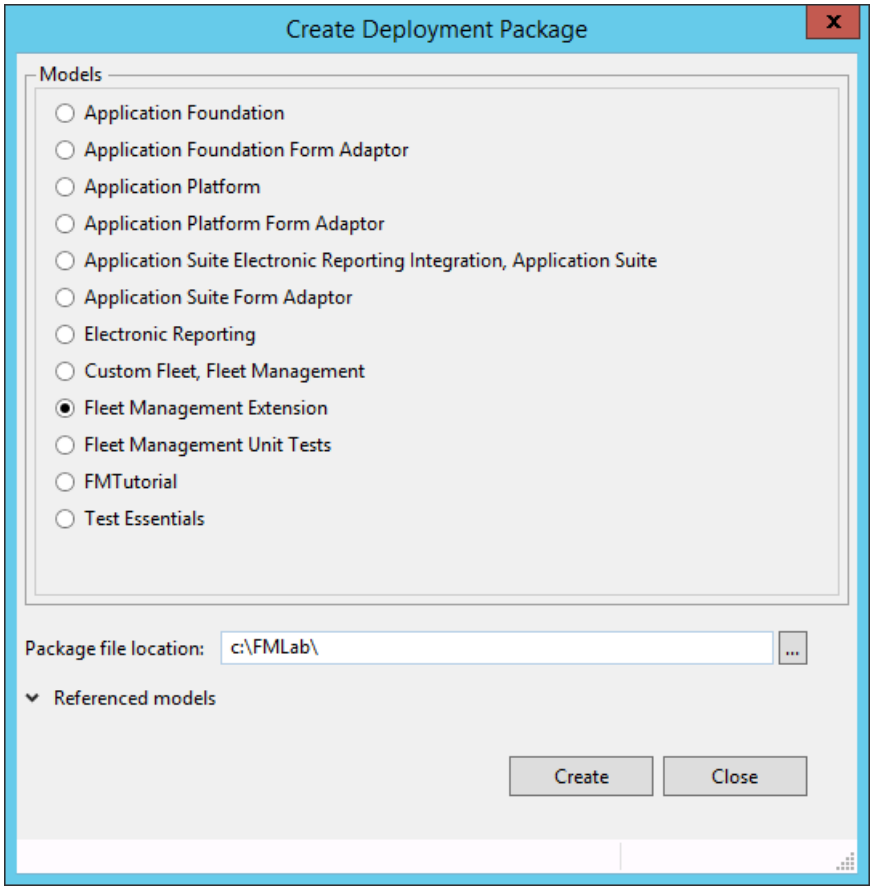
Data	
Allow Negative	Auto
Array Index	0
Auto Declaration	No
Cache Data Method	Auto
Configuration Key	
Country Region Codes	
Country Region Context Field	
Custom Display Name	DisplayMethod (Integer)
Data Field	
Data Method	**FMVehicle_Extension::CupHoldersDisplay**
Data Source	**FMVehicle**

Create a Fleet extension package for deployment

To deploy your extension to another environment, for example, a test, pre-production or production

environment, you must create a deployment package.

1. In Visual Studio, on the **Dynamics AX** menu, point to **Deploy**, and then select **Create Deployment Package**.



2. Select the **Fleet Management Extension** check box.
3. In the **Package file location** text box, enter "c:\FMLab".
4. Select **Create**. A deployment package that contains the Fleet management Extension package is created.

Additional resources

Download FMLab sample code

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Customize through extension and overlayering

2/18/2021 • 12 minutes to read • Edit Online

This topic discusses the two methods of customizing source code and metadata of model elements - overlayering and extensions and details supported extension capabilities.

Overlayering

You can customize source code and metadata of model elements that are shipped by Microsoft or third-party Microsoft partners. In order to customize metadata and source code of a model, the developer must create a new model that overlays the model they want to customize. For example, solution developers can provide code in the SLN layer, independent software vendors can use the ISV layer, and value-added resellers can use the VAR layer. Functionality defined in higher layers (VAR layer in this example) can override the functionality of lower layers. The overlaying model must belong to the same **Package** as the source model and belong to a layer that is higher than the source model. Overlayering is a powerful tool to perform advanced customizations of metadata and source code, but may increase the cost of upgrading a solution to a new version.

Extensions

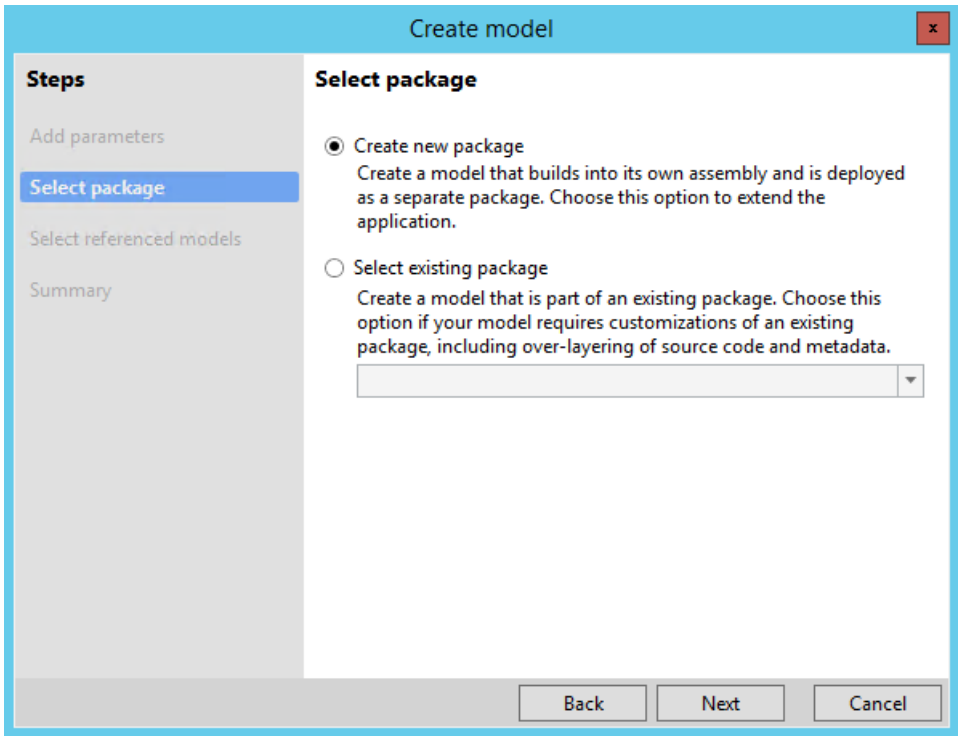
You can customize an application by using *extensions*. An extension enables you to add functionality to existing model elements and source code. Extensions provide the following capabilities:

- Creating new model elements.
- Extending existing model elements.
- Extending source code using class extensions.
- Customizing business logic. Ways to customize business logic include:
  - Creating event handlers to respond to framework events, such as data events.
  - Creating event handlers to respond to event delegates that are defined by the application.
  - Creating new plug-ins.

To get started, review or complete this tutorial: Customize model elements through extension.

Extension models and packages

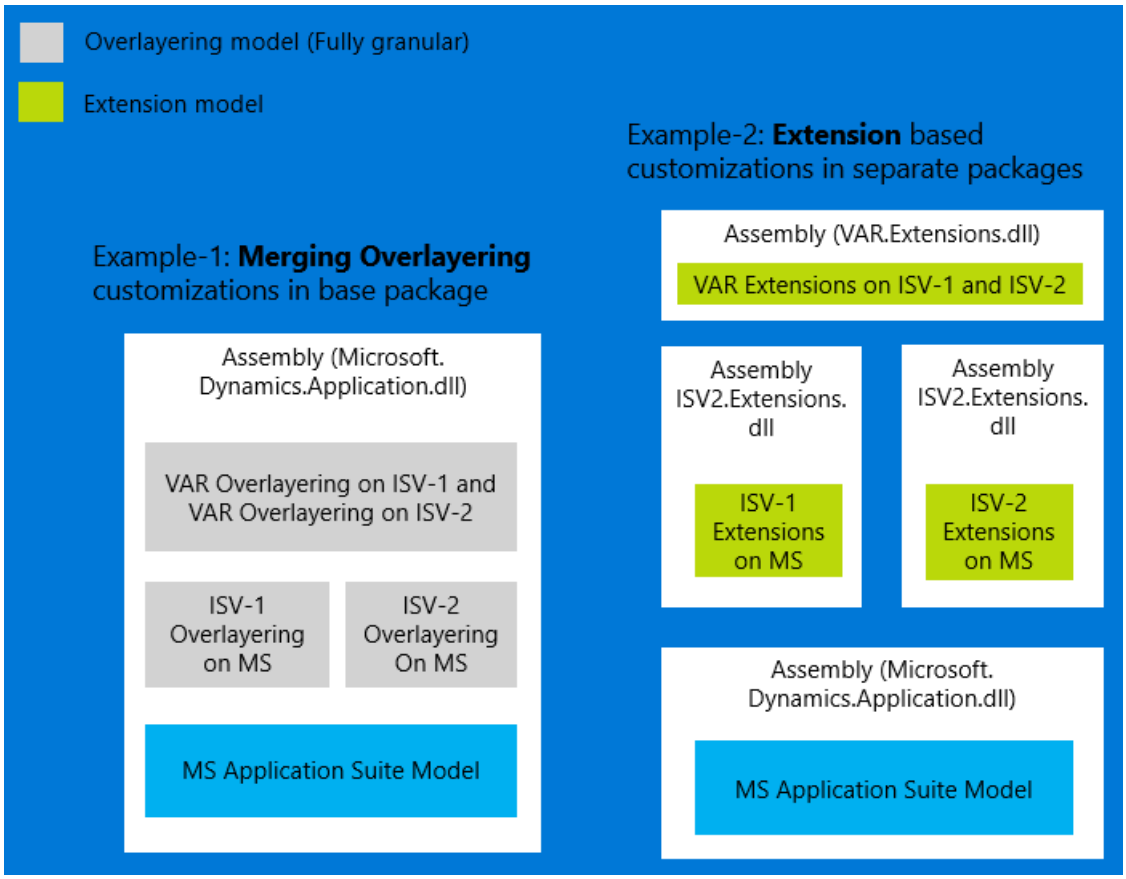
You can create a model that contains only new model elements, new code, or extensions. This model is compiled into its own separate assembly. These assemblies, along with related metadata and runtime artifacts can be packaged (as a deployable package file) and deployed on runtime sandbox or production environment. To create an extension model, go through the Create model wizard and select **Create new package** on the second step.



Extension models have several advantages, including:

- **Application lifecycle management (ALM):** Extension models simplify and improve the performance of deployments, builds, test automation and delivery to customers.
- **Design time performance:** Building your model or project doesn't require you to recompile the entire application.
- **Servicing:** In the cloud, Microsoft can install, patch, upgrade, and change internal APIs without affecting your customizations.
- **Upgrades:** Unlike overlayering, extensions reduce the cost of upgrading to a new version, as this approach eliminates costly code and metadata conflicts.

The following diagram illustrates how extensions get isolated in their assemblies.



Code extensions

You can extend source code in 3 ways:

- By subscribing to events (framework events and delegates)
- By writing plug-ins.
- By creating class extensions (aka class Augmentation), see section below.

You should understand the following characteristics of framework events:

- Events are implemented as multi-cast delegates, which means that more than one event handler can be subscribed to any particular event.
- Events are broadcast; there's no sequencing of calls to event handlers.
- Event handlers execute within the transaction scope of the base methods.

Events

Events are raised as preceding and succeeding operations around the base methods. This means that you have the opportunity to run code before a base method is called and after it has completed. Microsoft Dynamics AX 2012 introduced XPP events, which are also available in this release and can be subscribed to in your extensions.

Plug-ins

Plug-ins are extension points that are defined by the base application. By using a class-factory pattern, plug-ins enable you to replace the base functionality. You can see how to implement a plug-in in the tutorial, Customize model elements through extension.

Class Extensions

Class extensions enable you to augment a class by adding methods and variables to existing classes, tables and forms. For more details, refer to the topic Class extension model in X++.

Form extensions

You can extend the functionality of a form by extending its controls and data sources. For example, in a form extension, you can:

- Add a new control.
- Enable or disable a control.
- Change the text or label property of a control.
- Change a control's visibility.
- Change a form's help text.
- Change a form's caption.
- Add a new data source.
- Add a form part.

Other ways to customize a form, such as reordering controls in the form are planned to be included in a future release. In Microsoft Dynamics AX 2012, you could override form methods. In the current version, you use extensions to implement event handlers that are called from the base implementations of form methods. The following table lists each method and its associated events.

PUBLISHED FORM DATASOURCE METHOD	PRECEDING EVENT	SUCCEEDING EVENT
active	N/A	Activated
delete	Deleting	Deleted
validateWrite	ValidatingWriting	ValidatedWrite
write	Writing	Written
create	Creating	Created
executeQuery	N/A	QueryExecuted
linkActive	N/A	PostLinkActive
init	N/A	Initialized
validateDelete	ValidatingDelete	ValidatedDelete
reread	N/A	Reread
selectionChanged	N/A	SelectionChanged
markChanged	N/A	MarkChanged
leaveRecord	LeavingRecord	LeftRecord

PUBLISHED FORM OBJECT METHOD	PRECEDING EVENT	SUCCEEDING EVENT
init	Initializing	Initialized
close	Closing	N/A

PUBLISHED FORM OBJECT METHOD	PRECEDING EVENT	SUCCEEDING EVENT
run	N/A	PostRun
activate	N/A	Activated

PUBLISHED FORM CONTROL METHOD	PRECEDING EVENT	SUCCEEDING EVENT
modified	N/A	Modified
validate	Validating	Validated
leave	Leaving	LostFocus
enter	N/A	Enter
gotFocus	N/A	GotFocus
clicked	N/A	Clicked
selectionChange	SelectionChanging	N/A
pageActivated	N/A	PageActivated
allowPageDeactivate	AllowPageDeactivate	N/A
expand	Expanding	Expanded
tabChanged	N/A	TabChanged
dialogClosed	N/A	DialogClosed

Code behind extension forms

You can use class extensions to author X++Â logic associated with form extensions. This allows the definition of state variables accessible to form and control event handlers. It also allows overriding form methods without overlayering code. Refer to this blog article for an example.

Table extensions

You can create a table extension to extend a table's design and logic. You can add new fields, field groups, indexes, mappings and relations. You can also add new fields to existing field groups, change the label of a table field, change the Created By, Created Date Time, Modified By, Modified Date Time properties. Using table extensions, you can also change the Extended Data Type property on fields and set it to an EDT that is derived from the current EDT (*This is available as of platform update 8*).

In Microsoft Dynamics AX 2012, you could override the virtual methods of a table's base class to control the behavior that occurred during table operations, such as when creating, reading, updating, or deleting. In the current version, you instead use extensions to implement event handlers that are called from the base implementations of the table methods. The following table lists each table method and its events.

PUBLISHED TABLE METHOD	PRECEDING EVENT	SUCCEEDING EVENT
validateWrite	ValidatingWrite	ValidatedWrite
validateDelete	ValidatingDelete	ValidatedDelete
validateField	ValidatingField	ValidatedField
validateFieldValue	ValidatingFieldValue	ValidatedFieldValue
modifiedField	ModifyingField	ModifiedField
modifiedFieldValue	ModifyingFieldValue	ModifiedFieldValue
Insert	Inserting	Inserted
Update	Updating	Updated
Delete	Deleting	Deleted
Initvalue	InitializingRecord	InitializedRecord
FinalDeleteValidation	Executed when a delete operation is performed on a table object, before the operation is committed to the underlying database table	N/A
FinalInsertValidation	Executed when an insert operation is performed on a table object, before the operation is committed to the underlying database table	N/A
FinalReadValidation	Executed when a read operation is performed on a table object.	N/A
FinalUpdateValidation	Executed when an update operation is performed on a table object, before the operation is committed to the underlying database table.	N/A

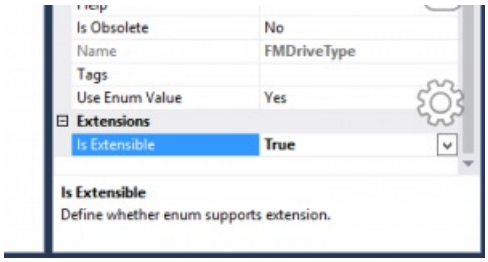
Validation events capture and return results by using the **DataEventArgs** parameter. The display and edit method modifiers are supported on table extensions.

View and Data entity extensions

You can extend a View or Data entity to achieve much of the functionality available with table extensions.

Enum extensions

You can extend any Enum that is marked extensible (IsExtensible=True).



By extending an Enum, you can add new Enum values to it. It is important to keep the following in mind when dealing with extensible Enums:

1. You cannot have X++ logic that depends on the integer value of Enum values (For example. *If (Enum1.v1 > Enum1.v2) ...* is not supported for extensible enums)
2. When Enum values of extensible Enums are synchronized into the database:
   - Integer values that belong to the baseline enum are deterministic, they come from the metadata.
   - Integer values that are an extension are generated during the synchronization process and are not deterministic.

EDT extensions

You can extend an EDT element in order to modify any of the following properties:

- Form help
- Label
- String size
- Help text

Query extensions

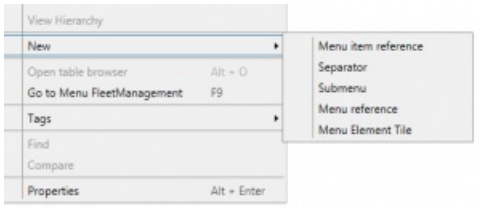
You can extend a Query element to achieve the following:

- Add ranges to an existing data source.
- Add new (embedded) data sources to an existing data source.
- Add new fields to an existing data source.

Menu extensions

You can extend a Menu element to achieve the following:

1. Add new menu items, submenus, menu references and tile references to an existing menu.
2. Hide an existing menu item, tile, or sub-menu in a menu by setting the **Visible** property to No.



Security role and duty extensions

You can extend a Security Role or a Security Duty to add new duties/privileges to these elements.

Report extensions

You can customize reports and business docs using extensions, below is a list of tutorials that help you learn more.

Customize App Suite reports by using extensions: Customizations to reporting solutions in the standard application are fully supported using a pure 'Extension' model. This article offers guidance on how to add the most common customizations to standard application reports without over-layering Application Suite artifacts. Here are some...

Create custom designs for business documents: This article focuses on the steps involved in crafting a custom report design for an existing application business document using a 'pure' extension model. Follow the steps below to associate a custom report design with an application document instance....

Expand Application Suite report data sets: This article focuses on the expansion of an existing report data set produced using X++ business logic in a Report Data Provider (RDP) class. Use custom delegate handlers and table extensions to include additional field data and/or calculations without...

Extend report menu items to redirect user navigation: This article focuses on the process of extending existing application menu items to redirect navigations with minimal code changes. Using this technique you will avoid the hassle of tracking down and replacing all references to an existing application...

Label extensions

You can create label extension files in order to modify the string value of a label, add new labels to the same label file or add new languages. To create a label extension file you must name it with a _extension suffix. For example, to extend the **FLM** labels of the Fleet Management model, do the following:

1. Create a project that belongs to a model that references Fleet Management (The model Fleet Management Extension is an example).
2. Add a new label file to the project and name it **FLM_Extension**.
3. Within the FLM_Extension label file, you can create new labels or modify the value of labels that are defined in the **FLM** label file of the Fleet Management model. Use the standard label editor to define new labels or redefine labels that already exist in the original FLM label file.
4. If your goal is to create translations of the FLM label, right-click on the FLM_Extension element in your project and select **Add new languages**. Follow the wizard to add translation files to the FLM labels.

> **NOTE**
> If the FLM_Extension file already exists in another model, you can name your file **FLM_ExtensionN** where N is any integer (For example FLM_Extension2, FLM_Extension3, ...etc)

Extension of Country/Region Codes

> **NOTE**
> This functionality is available as of Platform update 7.

The **Country Region Codes** property enables developers to restrict functionality to certain regions or countries based on the current legal entity's primary address. Developers can extend this functionality by setting the Country Region Codes property on the following extension element types: Menu extension, Menu Item extension, Table extension (and fields), Form extensions (form controls), EDT extensions, Enum extensions, and View extensions.

You can specify additional country/region codes in their extension. The effective country/regions (at runtime) associated with an element will be the union of all codes from the baseline element and all its extensions.

Event argument types

When an event takes place, the delegates described in the sections above get triggered. In this section, we provide the details of the types of the arguments that are passed as the event arguments. Some of the entries in the table below have a null in the column designating the event args; this means that no arguments are passed - the relevant information is in the first argument (typically called sender) in this case.

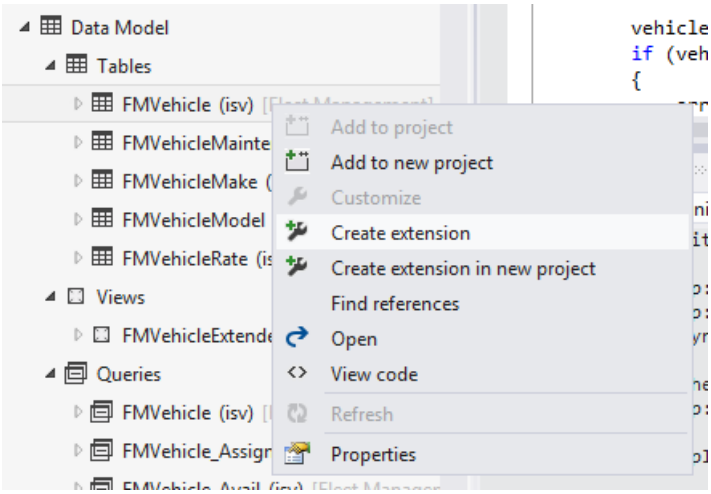
EVENT	ARGUMENT TYPE
onDefaultedField	DefaultFieldEventArgs
onDefaultedRow	null
onDefaultingField	DefaultFieldEventArgs
onDefaultingRow	null
onDeleted	null
onDeletedEntityDataSource	DataEntityContextResultEventArgs
onDeleting	null
onDeletingEntityDataSource	DataEntityContextResultEventArgs
onFindingEntityDataSource	DataValidationEventArgs
onFoundEntityDataSource	DataEntityContextRecordEventArgs
onGettingDefaultingDependencies	DefaultingDependenciesEventArgs
onGotDefaultingDependencies	DefaultingDependenciesEventArgs
onInitializedEntityDataSource	DataEntityContextEventArgs
onInitializedRecord	null
onInitializingEntityDataSource	DataEntityContextEventArgs
onInitializingRecord	null
onInserted	null
onInsertedEntityDataSource	DataEntityContextResultEventArgs
onInserting	null
onInsertingEntityDataSource	DataEntityContextResultEventArgs
onMappedDatasourceToEntity	DataEntityContextEventArgs
onMappedEntityToDataSource	DataEntityContextEventArgs

EVENT	ARGUMENT TYPE
onMappingDatasourceToEntity	DataEntityContextEventArgs
onMappingEntityToDataSource	DataEntityContextEventArgs
onModifiedField	ModifyFieldEventArgs
onModifiedFieldValue	ModifyFieldValueEventArgs
onModifyingField	ModifyFieldEventArgs
onModifyingFieldValue	ModifyFieldValueEventArgs
onPersistedEntity	DataEntityContextEventArgs
onPersistingEntity	DataEntityContextEventArgs
onPostedLoad	null
onPostingLoad	null
onUpdated	null
onUpdatedEntityDataSource	DataEntityContextResultEventArgs
onUpdating	null
onUpdatingEntityDataSource	DataEntityContextResultEventArgs
onValidatedDelete	ValidateEventArgs
onValidatedField	ValidateFieldEventArgs
onValidatedFieldValue	ValidateFieldValueEventArgs
onValidatedWrite	ValidateEventArgs
onValidatingDelete	ValidateEventArgs
onValidatingField	ValidateFieldEventArgs
onValidattingFieldValue	ValidateFieldValueEventArgs
onValidatingWrite	ValidateEventArgs

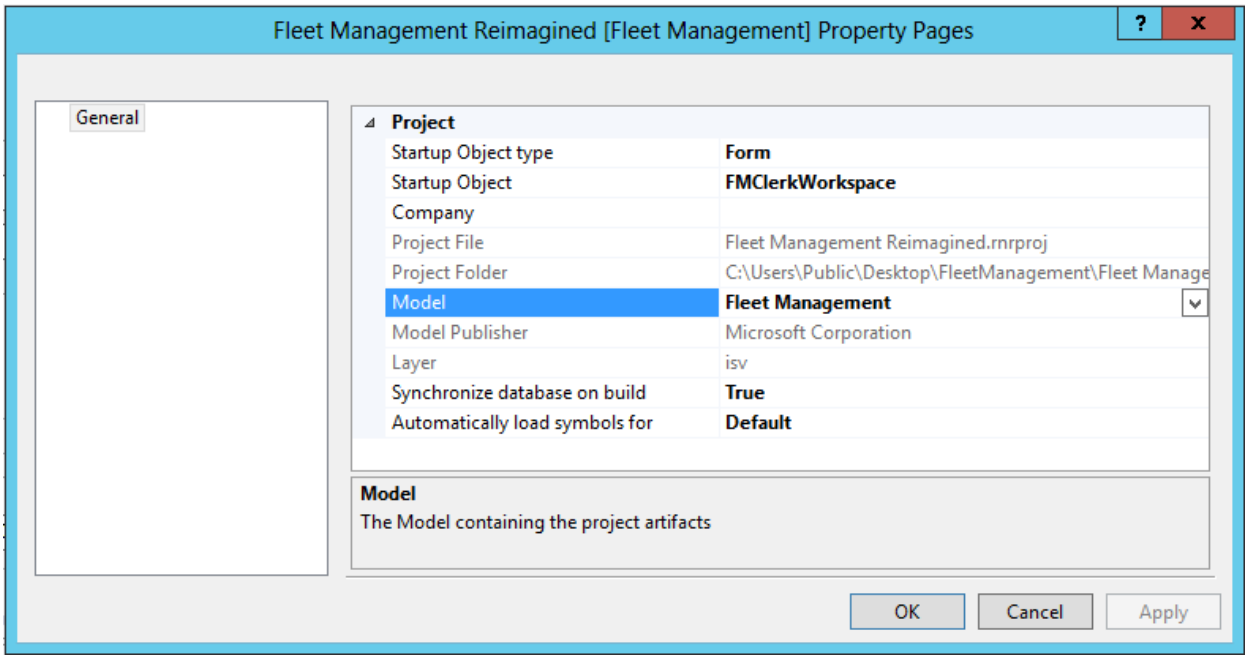
Development tools support

The development tools in Visual Studio provide integrated features to help you create and work with extensions. For example, when you right-click an element name in **Application Explorer**, you can create an extension for

that element.



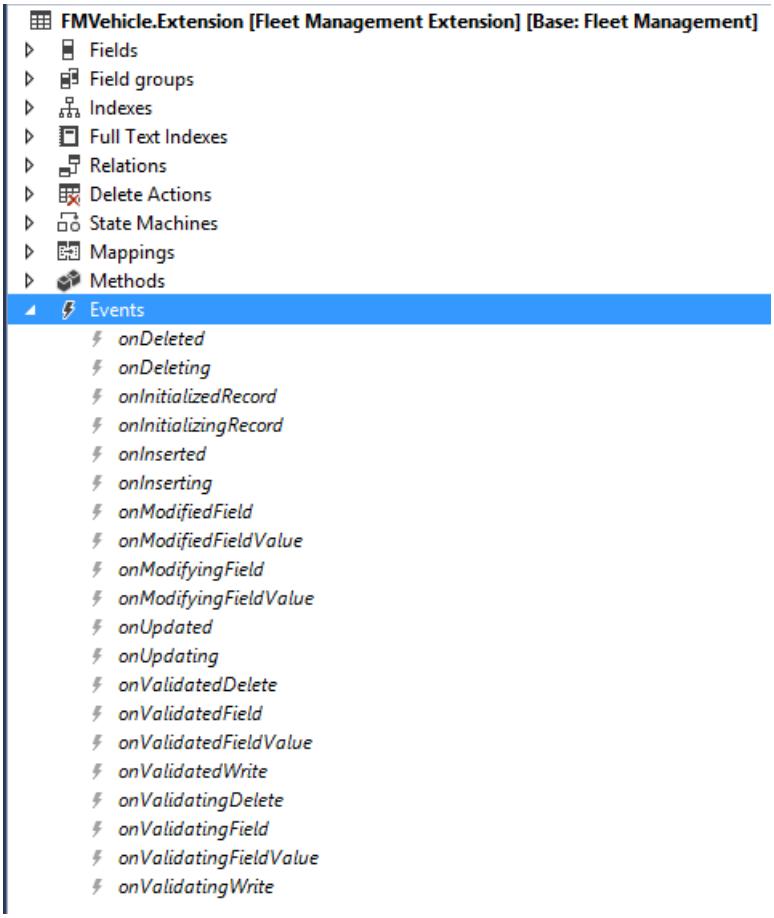
To create an extension, the current project in **Solution Explorer** must belong to a model that references the model of the selected element in **Application Explorer**. To view the model for a particular project, view the project properties.



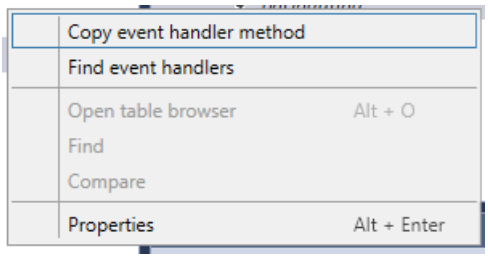
Visual Studio creates the extension file for you, either in the current project or in a new project. You can then work with the extension file either as source code or by using a designer. You package a code-extension model for deployment exactly like you would package any other model. On the **Dynamics 365** menu, point to **Deploy**, click **Create Deployment Package**, and then select the check box for the package name.

Framework events

Tables, form data sources, form controls, and other element types that support extension events list the available events (and delegates) under an **Events** collection node. For example, viewing the **Events** node of a table extension shows events that are defined by the framework, and delegate methods that are defined by application developers.



Note: Events are exposed on the designer on different element and sub-element types, like table events, form events, form data source events, form control events, and others. Open the context menu of an event node to interact with events:



- **Copy event handler method**: This option copies a method signature to the clipboard. You can paste it in any X++ code editor to define a method that subscribes to the selected event.
- **Find event handlers**: Searches and lists all methods subscribed to the selected event.

Additional resources

Customize model elements through extension

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

What's new or changed for extensibility

2/18/2021 • 2 minutes to read • Edit Online

This topic provides links to extensibility updates.

- Extensibility changes version 10.0.3
- Extensibility changes version 10.0.2
- Extensibility changes version 10.0.1
- Extensibility changes version 10.0
- Extensibility changes version 8.1.3
- Extensibility changes version 8.1.2
- Extensibility changes version 8.1.1
- Extensibility changes version 8.1
- Extensibility changes version 8.0.4
- Extensibility changes version 8.0.3
- Extensibility changes version 8.0.2
- Extensibility changes version 8.0.1
- Extensibility changes in version 8.0
- Extensibility changes in version 7.3
- Extensibility changes July 2017

Extensibility changes in Dynamics 365 for Finance and Operations version 10.0.3

2/18/2021 • 2 minutes to read • Edit Online

This topic lists the extensibility features that were implemented in Microsoft Dynamics 365 for Finance and Operations version 10.0.3. For more information about the schedule of changes that support extensibility, see Application extensibility plans.

Enumerations made extensible

The following enumerations have been made extensible in this update:

- LedgerJournalWFApprovalModule
- RetailReceiptTransaction

SQL operations made extensible

The following SQL operations have been made extensible in this update:

- CustVendTrans support for an extensible map pattern

Metadata changes

The following metadata changes have been made in this update:

- CostSheetAmount.NoOfDecimalsIsExtensible
- SalesLinePercent.NoOfDecimalsIsExtensible

Refactored methods

The following methods have been refactored to support extensibility:

- BankReconciliationDataInitializer.initDocumentOpenTmp
- BankReconciliationDataInitializer.initStatementOpenTmp
- Class\BomCalcJob_All.processSingleTask
- Class\KanbanEventQuantityMap.newStandard
- Class\MCRFullTextSearchRefresh.run
- Class\PdsRebateAgreementValidate.validate
- Class\PurchRFQFormLetter.main
- Class\ReqTransPoMarkFirm.getPurchIdSingleThread
- Class\TAMVendRebateCorrectClaims.correctClaims
- Class\TAMVendRebateCorrectClaims.createClaimCorrection
- Class\TAMVendRebateCorrectClaims.rebateAmountPerUnit
- Class\WhsReleaseToWarehouseForm.buttonRelease_clicked
- Classes\LedgerAllocationRules.ValidateDimension
- Classes\ProjJournalCheckPost.checkFeeJournalDimensions
- Commission_Sales.run
- CreditcardPaymentCardTokenize.getFromDialog

- CustInPaymDialog.openDialog
- CustVendDisputeHelper.canDeleteDispute
- EcoResCategoryTreeDatasource.new
- EcoResProductRelationtable.validateWrite
- Form\EcoResProductCreate.updateCallers
- Form\InventOnhandReserve\DataSource\InventSum.reserveNow
- Form\PdsRebateAgreement\DataSource\PdsRebateAgreement.executeQuery
- Form\ProcCategoryHierarchyManagement\FormDesign\CategoryTreeGroup\CategoryTreeCtrl.selection
- Form\ReqSupplyDemandSchedule.updateDesign
- Form\SalesTable\DataSource\MCRSalesLineDropShipment\field\DropShipment.modified
- Form\SalesTable\DataSource\SalesTable.create
- FormletterService.removeProforma
- InventJournalTrans.validateWrite
- InventJournalTrans_Tag.validateWrite
- InventMovement.addLedgerPhysicalAmount
- InventMovement.canAutoReserveQuantity
- InventTransSerialNumberCreate.checkFormat
- InventUpd_Reservation.updateReserveLess
- LedgerJournalEngine.onSegmentChangedForPrimaryAccount
- LedgerJournalTransCustPaym.enableDisableMandate
- LedgerTransStatementDP.processOffsetAccountInStaging
- PdsBatchAttribReserveForm.checkReserveLine
- PriceDisc.findDiscAgreement
- PriceDiscAdmCheckPost.postJournal
- PriceDiscHeading.updateDiscQty
- PriceDiscHeading.updateMultiLineDiscTmp
- PriceDiscPolicyFindOrCreate.run
- ProjInvoiceControl.projInvoiceControl
- ProjPostCostJournal.new
- PurchLineType.validateWrite
- ReqTransFormExplosion.tmpReqExplosionOnhandBuildServer
- ReqTransPoMarkFirm.createPurchTable
- ReqTransPoMarkFirm.updatePurchBuyerGroup
- RequisitionPurchaseOrderGeneration.createPurchaseOrder
- RetailBarCodeManagement.CreateBarCodeNoDim
- RetailTransactionServiceOrders.createCustomerOrder
- RetailTransactionTransformer.readTransactionSalesTrans
- SalesLine.createLine
- SalesLine.initFromPriceDisc
- SalesLine.insert
- SalesLine.update
- SalesLine.validateDelete
- SalesLine.writeRetailSalesLine
- SalesTable.updateMultiLineDisc
- SmaServiceFunctionLine_transfer.Run
- SmmOpportunityStatusUpdate.updateFromQuote

- Table\MCRCustpaymTable.salesTableByPassCreditLimit, displayOrderID, getCurrency, and mcrCustPaym\getCustomerPostingProfile
- Table\ReqPO.findAnySalesLineForReqPO
- TaxUncommitted.createTaxUncommitted, added local method createTaxUncommittedFromTmpTaxWorkTrans
- TmpTaxReport_IT.create
- TrvExpTrans.defaultTaxGroupFromWorker
- WHSBillOfLadingDP.insertWHSBillOfLadingTmp
- WHSControlItemId.populate
- WhsWarehouseRelease.creditLimitCheck
- WhsWorkCreate.addRangesToWorkTemplateQuery
- WHSWorkExecute.CreateTransferJournalLine
- WhsWorkTypePrintHandler.buildLabelAndConfirm

Other extensibility enhancements

- The PriceDiscHeading map was made extensible.
- **Retail channel:** Pre-triggers were added for Shipped, PackingSlip, and MarkAsPacked.
- **Retail channel:** The Cancellation charge dialog box can be overridden.
- **Retail channel:** Recall order default parameter value extension for the search order dialog box.

Extensibility changes in Dynamics 365 for Finance and Operations version 10.0.2

2/18/2021 • 2 minutes to read • Edit Online

This topic lists the extensibility features that were implemented in Microsoft Dynamics 365 for Finance and Operations version 10.0.2. For more information about the schedule of changes that support extensibility, see Application extensibility plans.

Enumerations made extensible

The following enumerations have been made extensible in this update:

- MarkupModuleType
- MCRCustPaymType
- PaymSchedBy

SQL operations made extensible

The following SQL operations have been made extensible in this update:

- JmgPayAdjustment.payAdjustLoop
- ProjPosting.ExtensionHash.New field
- WmsArrivalOverviewGeneration.buildPurch
- WmsArrivalOverviewGeneration.buildTransferOrder

Metadata changes

The following metadata changes have been made in this update:

- CostSheetPercent.NoOfDecimalsIsExtensible
- WHSCycleCountingWarehouseWorkLineEntity.IsPublic

Refactored methods

The following methods have been refactored to support extensibility:

- /Forms/ProjJournalTable/datasource/ProjJournalTable.initValue
- /Forms/PurchReqTable.instantiatePurchReqTableForm
- /Forms/PurchReqTable/DataSource/PurchReqTable.init
- /Forms/SalesQuotationProjLinkWizard/Controls/ProjInvoiceId.lookup
- /Tables/SalesTable.lastQuotation
- AccPolicyProductReceipt.isAccountingRequiredForSourceDocLine
- AssetFixedAssetEntity.overrideDataSource
- AssetProposalDepreciation.run
- AssetTableMethod.init
- BankAccountReconcile.validate
- Class\BomCalcCost.calcCostModel
- Class\MCRLoadContinuityCustInfo.insertLineData

- Class\McrPriceHistoryUpdate.insertNewlyFoundReferences
- Class\McrPriceHistoryUpdate.update
- Class\McrPriceHistoryUpdate.updatePriceHistoryLineReferences
- Class\ProjCopyItemEstimates.copyToItemRequirment
- Class\PurchAutoCreate_RFQ.createPurchOrderRFQLineReference
- Class\ReqEventProcessDeleteUnusedKanban.deleteUnusedKanban
- Class\ReqEventProcessDeleteUnusedKanban.run
- Class\ReqTransUpdate.updateLogAddQty
- Class\SalesCancelOrder.run
- Class\SalesCreateOrderFromCustomer.main
- Class\TAMVendRebateCorrectClaims.rebateAlreadyGiven
- Class\tamVendRebateTableStatusType_Approved.runPayment
- Class\TamVendRebateTableStatusType_Calculated.inserted
- Class\TamVendRebateTableStatusType_Calculated.runPayment
- Classes\TaxWithhold.createAllTaxWithholdTrans
- Classes\TaxWithhold.isCalculateTaxWithholdingNeeded_TH
- Classes\TaxWithhold.postTaxWithhold
- Classes\TaxWithhold.totalInvoiceLineAmountSettled_TH
- CustDirectDebitMandate.setDefaultMandate
- CustDueReportDetailDP.class declaration
- CustDueReportDetailDP.insertCustDueReportDetailTmp
- CustQuotationConfirmJour.printJournal
- CustVendCreatePaymJournal.pack
- CustVendCreatePaymJournal.parmHasBatchBeenSplit
- CustVendEditTaxBranch_TH.init
- CustVendSumForPaym.run
- CustVendTransSettlement.post
- DimDerDistRuleSalesComplInvoice_BR.createDimAllocForProjRevenue
- EcoResProductCreateExtended.SetAllowEditField
- EcoResProductVariantEntity.findDataSource
- FBSpedFileCreator_Fiscal_BR.createRecordC195
- Form\ProdTableCreate.canContinueWithEmptyDim
- Form\PurchCreateFromSalesOrder\DataSource\SalesLine.included
- Form\PurchCreateFromSalesOrder\DataSource\SalesLine.specifyVendAccount
- Forms\TaxWithholdTable.init
- FreeTextInvoiceDP.setSysDocuBrandDetails
- InventItemOrderSetupMap.checkNotStopped
- InventNonConformanceTable.InventNonConformanceTable.Create
- InventUpd_Estimated.updateAutoDimBatchId
- InventUpdateReserveMore.InventUpdateReserveMore
- InventValueReportPopulateItem.updateReportLinePL
- JmgCalcApproveDateView.viewDate member
- JmgCalcApproveWeekView.viewDate member
- JmgPayAdjustment.payAdjustLoop
- LedgerJournalPeriodicCopy.journalTransCopy
- LedgerTransStatementDP.populateTempTableLedgerInStaging

- MCRCustpaym.validateWrite
- MCRFullTextSearch.buildSearchText
- MCRFullTextSearch.truncate
- MCRHoldCodeTrans.insert
- MCRHoldCodeTrans.setOrderStoppedFlag
- MCRHoldCodeTrans.unreserve
- McrPriceHistoryForm.calcPotential
- McrPriceHistoryForm.insertPotentialTradeAgreements
- PaymTerm.validateWrite
- PdsRebateAgreement.checkLineBreaks
- PdsRebateAgreement.groupChangeCheckValid
- PdsRebateAgreement.lineAmountHasGapOrOverlap
- PdsRebateAgreement.lineQuantityHasGapOrOverlap
- PdsRebateAgreementLine.selectRebateAgreementLineMax
- PriceDisc.mcrCalcPostageDisc
- PriceDiscLinePolicyRule.retrieveSystemPolicyFieldList
- ProdUpdCostEstimation.updateSubPurchLine
- ProjBudgetManager.createBudgetLineDetail
- ProjBudgetManager.getQuery
- ProjForecastCost.validateWrite
- ProjForecastEmpl.validateWrite
- ProjForecastRevenue.validateWrite
- ProjLedgerUpdate.insert
- ProjPlanVersionsManager.importHierarchy
- ProjPlanVersionsManager.importProjPlanVersionRecords
- ProjPost.PostCost
- ProjPost.PostCost
- ProjWorkBreakdownStructureHelper.addQuotationRelatedRecordsForTask
- ProjWorkBreakdownStructureHelper.Addtask
- ProjWorkBreakdownStructureHelper.Addtask
- ProjWorkBreakdownStructureHelper.Addtask
- ProjWorkBreakdownStructureV2FormHelper.IndentTaskV2
- ProjWorkBreakdownStructureV2FormHelper.MoveTasks
- PurchFormletterParmDataInvoice.createParmLinesAndTable
- PurchLine.delete
- PurchLine.distributionUpdateNeeded
- PurchLine.initFromPriceDisc
- PurchLine.insert
- PurchLine.update
- PurchLineType.statusChangeAllowed
- ReqEventProcessKanban.newStandard
- ReqTransNeutralTracker.trackReqTrans
- ReqTransPoMarkFirm.create
- Retail channel: CartWorkflowHelper.AllowAggregation
- RetailEcoResProductReleaseManager_Extension.setAndSaveRetailProductProperties
- RetailMassUpdateUploadDBManager.insertIntoProductProperty

- RetailPeriodicDiscount.validatePriceGroup
- RetailTransactionServiceCustomer.newCustomer
- RetailTransactionTransformer.ReadDiscountLines
- SalesInvoiceDP.setSysDocuBrandDetails
- SalesInvoiceJournalPost.run
- SalesInvoiceJournalPostBase.run
- SalesLine.CheckItemId
- Table\InventTable.purchPriceAgreement
- Tables\TaxWithholdTrans.copyTaxWithholdTrans, initFromTaxWithholdTable, insert, validateWrite, amountTotalWHT, existPeriod_TH
- TAMVendRebatePaymentPost.main
- TAMVendRebateTableProcess.runProcess
- TrvPostExpenseHeader.postCustVendTransactions
- TrvPostExpenseHeader.postCustVendTransactions
- WhsControlLicensePlateId.process
- WhsLicensePlateLabelBuild.insertSingleLabelMenuItem
- WhsLicensePlateLabelBuild.insertSingleLabelPrintLine
- WhsrfControlData.processLegacyControl
- WhsWorkCreateProdPut.createReportFinishedParameters
- WhsWorkCreateProdPut.insertProdParmforCoByProduct
- WhsWorkCreateProdPut.insertProdParmForProdItem
- WhsWorkCreateProdPut.setAcceptError
- WHSWorkCreateReplenishment.checkExistingReplenWork
- WhsWorkExecuteDisplay.buildPick
- whsWorkExecuteDisplayInquiryLocation.buildLocationInquiry
- WmsArrivalOverviewGeneration.buildInventTransId
- WmsOrderTransType_OutputDontPostTransfer.decreaseQty
- WmsOrderTransType_OutputDontPostTransfer.increaseQtyOverdelivery

Other extensibility enhancements

- The accessModifier of Classes\BankPositivePayExport.Class changed from private to protected.
- The InventItemOrderSetupMap map was made extensible.
- **Retail channel:** Custom columns in RetailTransactionView.
- **Retail channel:** The sign-in request can be overridden.
- **Retail channel:** Shipping view extension controller class.
- **Retail channel:** Support for the AppBar button in AddressAddEditView.
- **Retail channel:** Support for overriding the Bank deposit amount key in the dialog box.

Extensibility changes in Dynamics 365 for Finance and Operations version 10.0.1

2/18/2021 • 2 minutes to read • Edit Online

This topic lists the extensibility features that were implemented in Microsoft Dynamics 365 for Finance and Operations version 10.0.1. For more information about the schedule of changes that support extensibility, see Application extensibility plans.

Enumerations made extensible

The following enumerations have been made extensible in this update:

- ACOCostStatus_BR
- ACOCostType_BR
- ACOJournalType_BR
- BankModuloCheck_NO
- InventTransferOrderType_BR
- ProdJourType
- ProjTransStatus
- RetailLabelTypeBase
- RetailLedgerBank
- RetailTenderFunction
- SalesPurchTrntype_BR
- SMAGetPriceFrom
- SMASubscriptionIndexChange

SQL operations made extensible

The following SQL operations have been made extensible in this update:

- InventSumDelta.findInventSumDeltaInventSumFieldsAll.
- LedgerFiscalJournal was changed so that it uses QueryObject.
- TaxTransDP.

Metadata changes

The following metadata changes have been made in this update:

- Data Entities/WMSItemArrivalJournalLineEntity.IsPublic, PublicCollectionName, PublicEntityName.
- DataEntities/LedgerJournalNameEntity/Fields/voucherSeriesCode.Allow Edit, Allow Edit on Create.
- DataEntities/LedgerJournalNameEntity/Fields/VoucherSeriesCompanyId.AllowEdit.
- Enums\SalesStatus::Backorder, Delivered.Label.
- Extended Data Types/WeightBase.Scale.
- InventTransferOrders added a form control group in the grid.

Refactored methods

The following methods have been refactored to support extensibility:

- AssetProposalDepreciation.run
- BankStatementValidate.validateDate
- Class\BankDocumentBankAccountTrans.loadSourceBuffer
- Class\BankReconMatchingRuleAutoProcessor.doProcessMatchRule
- Class\ProjJournalTransMapForm.initFromProjTable
- Class\RetailEodStatementPaymentJournal.createPaymentJournalLine
- Class\RetailKitAssemblyOrder.CreateOrUpdateBOMJournal
- Class\RetailTransactionServiceOrders.createOrUpdateRetailOrderLines
- Class\SalesInvoiceController.initReportName_IN
- Class\SalesInvoiceJournalPost.endUpdate
- Class\WrkCtrScheduler.loadRoute
- CreditCard.mcrInitFromCustPaymTable
- CreditcardProcess.mcrDoCapture
- CreditCardProcess.mcrDoRefund
- CreditCardProviderProcess.Submit
- CustCollectionLetterCreate.skipCustomer
- CustVendCheque.output
- CustVendSumForPaym.run
- EFDOCDanfe_BR.additionalInformationPageBreak
- EfDocDANFEDP_BR.additionalInformationBox
- ERDocuManagement.insertFile
- ERFileDestinationAttachment.saveFile
- ERFileDestinationBrowser.saveFile
- Form\BankReconciliationWorksheet.Init
- FreeTextInvoiceController.initReportName_IN
- HcmWorkerTransition.createHcmWorker
- InventCostClosingCancel_Init.createTasks
- InventDimCtrl_Frm_OnHand.initFromCaller
- InventMov_Jour_BOM.journalPostTrans
- InventProcessGuideDisplayLicensePlateDetailsPageBuilder.generateItemInfoForLicensePlate
- InventProcessGuideDisplayLocationDetailsPageBuilder.generateItemInfoForLocation
- InventStockCardDP.createInventStockCardTmpLineDetail
- InventTable.checkProjCategoryId
- InventUpd_WHSReservation.updateReserveMore
- JmgCalcApproveWeekView.initializeData
- LedgerConsolidate.Run and getSelectedDimensionAttributes
- LedgerFiscalJournalDP_IT.addStarsToTmpTable
- LedgerFiscalJournalDP_IT.insertLedgerFiscalJournalTmp_IT
- LedgerFiscalJournalDP_IT.insertLedgerFiscalJournalTmp_IT
- LedgerFiscalJournalDP_IT.processReport
- LedgerJournalTrans.checkAllowEditWhenCheckPrinted
- LedgerJournalTransUpdateVend.pdateNow
- LedgerVoucherTransList.First
- LedgerVoucherTransList.next
- MCRFullTextIndexField.tableIdFromEnum

- MCRFullTextIndexField.viewFromTable
- MCRInventSearch.searchProduct
- McrPriceHistoryLine_Purch.initAndInsertRebate
- PartyProvider.operatingUnitTypeToName
- PdsRebateAgreementValidate.construct
- POS_IssueLoyaltyCardView.NA
- PriceDiscAdmCheckPostPriceDiscTableUpdater.formattedQueryValue
- PriceDiscAdmTrans.checkItemRelation
- ProjBudgetManager.deleteProjBudgetLinesWhenZeroAmount
- ProjBudgetManager.updateProjBudgetLinesWithAmt
- ProjHourCostPrice.psaFindCostPrice
- ProjInvoiceProposalListPageInteraction.initializeQuery
- ProjPostEmplProposalSale.new
- ProjPostRevenueProposalSale.new
- ProjTable.initProjectFromCustomerAndInvoice
- ProjTransferPrice.findByContractResourceCategory, findTransferPrice, find
- ProjValElementServer.addProjToResource
- ProjValElementServer.deleteProjFromResource
- PurchPackingSlipJournalPost.postMarkupOnTrans
- PurchReqLine.setProjSalesPrice
- PurchRFQSendJournalCreate.createOrUpdateRFQLine
- ReqCalc.covCalcDim
- ReqCalc.covCalcDim
- ReqCalc.covCalcDim
- RequisitionPurchaseOrderGeneration.create
- RequisitionPurchaseOrderGeneration.create
- Retail extension point in the Commerce runtime (CRT) to override the ValidateCartLineQuantityAndPriceSymbol method
- RetailCatalogProductAttributeFormHelper.addProductAttributeControls
- RetailCreateSpecificLabel.makeLabel
- RetailEodStatementCustomerOrderInvoiceController.run
- RetailEodStatementPaymentJournal.ledgerBank2LedgerJournalACType
- RetailEodStatementPaymentJournal.postPaymentJournalForOthers, postPaymentJournalForSales, createTenderedPaymentLines, createPaymentJournalLine
- RetailEodTransactionTransformer.ReadTransactionHeader
- RetailEodTransactionTransformer.setExtensionProperty
- RetailEventNotificationAction.packingSlipCompletion
- RetailMediaAssociationHelper.associateProduct
- RetailProductPropertyManager.validateWriteOnInventModelGroupItem
- RetailSMBSeedGenerator.AccountReceivable
- RetailStatementPost.createPaymentLedgerTrans
- RetailTransactionSalesTransMark.MarkTransactions
- RetailTransactionServiceOrder.settleCustomerOrder
- RetailTransactionServiceOrders.cancelCustomerOrder
- RetailTransactionServiceOrders.createCustomerOrder
- RetailTransactionServiceOrders.createLedgerJournalForStore
- RetailTransactionServiceOrders.createOrUpdateRetailOrderHeader

- RetailTransactionServiceOrders.createOrUpdateRetailOrderLines
- RetailTransactionServiceTransactions.fillPaymentTransDetails
- RetailTransactionServiceTransactions.fillRetailTransactionDetails
- RetailTransactionServiceTransactions.fillSalesTransDetails
- RetailTransactionServiceTransactions.getJournalListQuery
- SalesCreateOrder.updateDeliveryAddress
- SubledgerJournalizer.loadAccountingdistributionTmp
- SubledgerJournalizer.recordSubledgerJourAccEntriesForRounding
- SubledgerJournalizer.recordSubledgerJournalAccountEntries
- Table\MCRContinuityScheduleLine.UpdateOtherLines
- Tax1099SummaryHelper.populateTaxSummaryFromVendSettlementTax
- TrvCreditCardTransactionEntity.validateWrite
- TrvExpenses.initializePersonalAmount
- TrvExpenses.updateFormVisibilityOnCategoryChange
- TrvExpenses.updateItemizationControls
- TrvExpTrans.copyValueToChildLines
- TrvExpTrans.defaultTaxGroupFromWorker
- TrvExpTrans.modifiedField
- TsTimesheetSignOffDP.insertTmpTSTimesheetSignOff
- WHSBillOfLadingDataUtil.populateCarrierInformation
- WHSBillOfLadingDataUtil.populateCustomerOrderInfo
- WHSLoadPlanningWorkbenchServerForm.addLoadLinesToLoad
- WHSLocationDirective.getValidSellableDaysQty
- WHSMobileAppAttachedImageDetails.getImageTypeFromSymbol
- WhsPostPackingSlip.updatePurchaseLoadLines
- WhsPostPackingSlip.updatePurchParmLineQuantityData
- WhsWarehouseRelease.main
- WhsWorkManualComplete.executeWorkLines
- WhsWorkManualComplete.performValidation
- WorkTimeCheckClassWorkCalendarDateLineTableWorkTimeLineTable class, validateWrite()
- WorkTimeLine.createWorkTimeCheck

Other extensibility enhancements

- **Retail channel:** Allow for extensions to support Select all and Clear all in OrderFulfillmentView.
- **Retail channel:** Expose Add return line to cart from the transaction application programming interface (API) by line ID.
- **Retail channel:** Line item locations can be viewed in OrderFulfillmentView.
- **Retail channel:** OrderFulfillmentView adds ICustomListColumn to allow for more information.
- Retail statement posting method adds another aggregation view by using the new RetailTransactionAggregationFieldList table that adds additional fields.

Extensibility changes in Dynamics 365 for Finance and Operations version 10.0

2/18/2021 • 2 minutes to read • Edit Online

This is a list of extensibility features that were implemented in Dynamics 365 for Finance and Operations version 10.0. For more information about the schedule of changes that support extensibility, see Application extensibility plans.

Enumerations made extensible

These enumerations have been made extensible in this update.

ENUMERATION
AssetAccrualCalendar
AssetYear
BankReconciliationReportType
BudgetPlanColumnPeriodLength
BudgetPlanHCMReportGroupOption
CurrencyTypeBrief_RU
EInvoiceStatus_IT
EInvoiceStatus_IT
HRPAuthorityBasis
HuExchOutflowType
InventJournalTagStatus
InvoiceAssociationType
MCRClaimType
MCRMerchandisingEventCategory
PaymAttribute
PaymProposalReportedBy
PayrollCategory

ENUMERATION
projActualVsBudget
ProjListStateId
ProjTransLayout
ProjType
RCashCheckContract
RCashDocRepresType
RCashDocType
RCashRemainLimitType
RCashTableAll
RCashTransStatus
SMARelationType
smmActivityTaskTimeType
TaxIdType
TrvAirlineServiceClassEnum
TrvFieldVisibility
TSPeriodFrequency
TSPerWeekMth
VendPaymentValidate

SQL operations made extensible

These SQL operations have been made extensible in this update.

OPERATION
JmgStampJournalTable.makeLines
MCRDropShipStatusUpdate_PurchLine.updatePurchDropShipStatusOnRecord
MCRDropShipStatusUpdate_PurchTable.updatePurchDropShipStatusOnRecord
SalesInvoiceJournalCreate.checkDocumentData_PL

OPERATION
SalesInvoiceJournalPost.postFailed

Metadata changes

These metadata changes have been made in this update.

OPERATION
/Data Model/Data Entities/BOMBillOfMaterialsVersionV2Entity.IsPublic
/Data Model/Data Entities/InventItemBatchEntity.IsPublic
/Data Model/Data Entities/InventProductSpecificOrderSettingsV2Entity.IsPublic
/Data Model/Data Entities/InventQualityGroupItemAssignmentEntity.IsPublic
/Data Model/Data Entities/InventQualityTestGroupEntity.IsPublic
/Data Model/Data Entities/ProductionPoolEntity.IsPublic
/Data Model/Data Entities/WMSItemArrivalJournalHeaderEntity.IsPublic, PublicCollectionName, PublicEntityName
/DataModel/Tables/WMSStorageLoadUnitReqTrans.WMSStorageLoadUnitReqTran
DimensionHierarchyType/EnumValue/RDeferrals
AOT/Data Model/Tables/CategoryTable.Create RecId Index
EcoResProductCategoryHierarchyEntity.Property.IsPublic
EcoResProductSpecificUnitOfMeasureConversionEntity.Property.IsPublic
EcoResReleasedProductVariantExternalCodeEntity.Property.IsPublic
InventProductSpecificOrderSettingsV2Entity.Property.IsPublic
"No of Decimals is Extensible" property on several EDTs
RetailLoyaltyRewardPoint.Replacement Key
Tables/CustTrans/Relations/ThirdPartyBankAccountId.Validate
Tables/EInvoicePropertyTable/Relations/EInvoicePropertyTypeTable.RelationshipType
Tables/ResourceSetup.FormRef
Tables/WHSTmpWorkExecuteListBoxItems/Fields/Elements.EDT

Refactored methods

These methods have been refactored to support extensibility.

REFACTORED METHODS
AdvancedLedgerEntryLine.setProjInvoiceLineLedgerDimension
AgreementConfirmationDP.getSalesAgreementHeader
AgreementConfirmationDP.getSalesAgreementHeaderHistory
PurchAutoCreate_Sales.createPurchLine
PurchCreateFromSalesOrder.run
InventItemPrice.insert
InventJournalTrans.setCostPrice
PurchLine.initFromReqPO
AssetBook.initDepreciationProfile
AssetDepreciationProfile.validateStraightLine
AssetProposalDepreciation.run
BankPaymAdvicePrint.BankPaymAdvicePrint (variable)
BankReconciliationMatchingMatchProcessor.constructMatch
BankReconMatchingMatchStmtReversalDoc.Multiple
BankStatementDocumentEntity.postGetStagingData
BankVoucher.post
BomCalcItemLine.mustExplodePrice
BOMCopyToProd.delete
BOMCreateDialog.promptCreateBOMDialog
BomRouteCopyJob.initFromItemId
BudgetPlanningConfiguration.displayYearOffset
BudgetPlanningConfiguration.updateColumnPeriodLengthValueLabel
CatVendorCatalogProductApproval.getApprovedProductForRetail
LedgerJournalTransType.validateAccountType
LedgerTransferOpening.processQuery

REFACTORED METHODS
BankPositivePayExport.generatePositivePayFile
BankPositivePayExport.updateBankPositivePay
CaseSendEmail.getEmailMessage
ContactPerson.insert
CostSheetModeStrategyStaging.createCostSheetNodes
CreditCard.recordAuthorization
CreditCard.recordCapture
CreditCardPaymentJournal.createJournal
CreditCardPaymentJournal.Init
CreditCardPaymentJournal.run
CustAgingReportContract.Validate
CustAgingReportDP.CustAgingReportTmp
CustAgingReportDPclass.insertCustAgingReportTmp
CustAgingReportDPclass.setCustAgingReportTmpInReverse
CustBalanceList.insertIntoTmpAccountSumV2
CustBillOfExchangePostRemit.postSettlingStep
CustCollectionsSetTransactionStatusHelper.createActions
CustCustomerBaseEntity/CustCustomerEntity/CustCustomerV2Entity/CustCustomerV3Entity.processChangesForApproval
CustCustomerDetailEntity/CustCustomerDetailV2Entity.processChangesForApproval
CustInvoiceJour.setInvoiceAddress
CustInvoiceLine.getCustBillingCodeLedgerAccount
CustInvoiceLine.setProjInvoiceLineLedgerDimension
CustInvoiceLine.setProjInvoiceLineLedgerDimensionBase
CustInvoiceLine.shouldDefaultLedgerDimensionFromProject

REFACTORED METHODS
CustOutPaymRecord_Cheque.checkValues
CustPostInvoiceJob.custPostInvoiceUpdate
CustVendAgingCalculation.process
CustVendChequeSlipTextCalculator.getChequeDocLength
CustVendChequeSlipTextCalculator.getMinimumSlipLines
CustVendChequeSlipTextCalculator.fillSlipText
CustVendChequeSlipTextCalculator.Property
CustVendEditTaxBranch_TH.init
CustVendOutPaym.getSumByCurrency
CustVendPaymInvoiceWithJournal.createJournal
CustVendPaymInvoiceWithJournal.createPayment
CustVendPaymProposal.resolvePaymAccountAndType
CustVendPaymProposalLine.paymTransactionAmountMST
CustVendPaymProposalTransferToJournal.getVoucherNum
CustVendReversePosting.restoreCustVendTransOpen
CustVendSettle.postDueToAndFromCreateTrans
CustVendSettle.postExchRateLedgerTrans
CustVendSettle.settleNow
CustVendSettle.updateCustTaxInvoice_TH
CustVendSumUpJournal.createTrans
CustVendSumUpJournal.createVoucher
CustVendTransreorg.end
CustVoucher.updateProjTransPosting
DimDerDistRuleProjectRevenueExt.processRegularTransactions
DimDerDistRuleProjectRevenueExt.processIntercompanyTransCustInvoice

REFACTORED METHODS
DimDerDistRuleProjectRevenueExt.processIntercompanyTransExpense
DimDerDistRuleProjectRevenueExt.processIntercompanyTransTimesheet
DimDerJourRuleProjectTimesheetsExt.getDefaultDimensionAllocation
EcoResEnumerationAttributeTypeValue.createAttributeValuesFromEnum
EcoResProductReleaseForm.addProductsToRelease
EInvoice_IT.newCustInvoice
EInvoice_IT.newProjInvoice
EUSalesListReportingEngine.Construct
FiscalDocument_BR.lastIssueDateForSeries
FormletterJournalPost.docuRefCopyByRecId
ForecastSales.Update
HcmActionState.lookupReferenceActionTypeSetup
HcmWorker.init
HcmWorker.updateEmploymentControls
HcmWorkerActionHireCompletion.getHrmApplication
HcmWorkerTransition.createHcmEmployment
HRCCompGridView.initCompRecord
HRMCompFixedEmpl.enforcePayRateTolerance
HRPDefaultSigningLimitRule.insertFormDataSourceJobDetail
HRPDefaultSigningLimitRule.populateDetailGrid
HRPDefaultSigningLimitRule.SaveValidation
HRPDefaultSigningLimitRule.insertOrUpdateFormDataSource
HRPDefaultSigningLimitRuleCompensation.getSelectedCompensation
HRPDefaultSigningLimitRuleCompensation.getAvailableCompensation
HRPDefaultSigningLimitRuleCompensation.selectRecords

REFACTORED METHODS
HRPDefaultSigningLimitRuleCompensation.unselectRecords
HrpWorkerLimit.getActiveDefaultSLRule,
HrpWorkerLimit.getDefaultSigningLimits
HrpWorkerLimit.getWorkerSigningLimit
HrpWorkerLimitr.getSigningLimitsIfRequestNotRequired
InterCompanyTransferInventDim.Entire class
InterCompanyTransferInventDim.transfer
InventBatch.update
InventCountCreate_Base.createInventJournalTrans
InventInventoryDimensionEntityFieldsMapping.resolveInventDim
InventMov_Jour_BOM.journalCheckTrans
InventMov_Jour_Loss_Project.checkAccountOperations
InventMov_Journal.journalSetItemId
InventMov_Statement.pdsCWRemainPhysical
InventMovement.performFinancialLedgerUpdate
InventProcessGuideAdjustInController.initialStepName
InventQualityManagementBlock.run
InventQualityManagementCreateHandler.purchFormLetterBeforeHelper
InventQualityOrderTableValidator.checkQty
InventSum.retrieveMatchingInventSumDeltaForTTSId()
InventTrackingRegisterTransForm.construct
InventTransAdjust.updateNow
InventTransferUpdReceive.updateInventTransferLine
InventTransWms_Register.updateInventFromMovementServer
InventUpd_ChildReference updateLess* methods

REFACTORED METHODS
InventUpd_ChildReference.updateMoreIssue
InventUpd_Estimated.updateAutoDimMovement
InventUpd_Physical.UpdatePhysicalReturnedIssue
InventUpd_Physical.updatePhysicalReturnedReceipt
InventUpd_WHSReservation.continueInventTransUpdateReserveMoveLoop
InventUpdateOnhand.checkOnhand()
InventUpdateReserveMore.buildQueries
JmgMESDocuHandling.openFile
JmgProfiles.insertTimeGapsPlannedAbs
JmgStampJournalCalculate.run
JmgStampJournalTransfer.cancelExecute
JmgStampJournalTransfer.cancelExecute
LeanCost_Init.execute
LedgerAllocationController.allocateAmounts
LedgerAllocationProcessRequest.createVoucherDestinations
LedgerJournalCheckPost.replaceTmpVoucher
LedgerJournalDeleteTransaction.deleteLedgerJournalTransRelated
LedgerJournalEngine.currencyModified
LedgerJournalPeriodicCopy.journalVoucherCopy
LedgerJournalTrans.initForCurrency
LedgerJournalTrans.validateWrite_Server
LedgerTransModule.insertTransactionList
LedgerTrialBalanceContract.DataMemberAttribute
LedgerVoucherObject.allocateTransaction
LedgerAllocationController.allocateRecursive

REFACTORED METHODS
MCRCheckHoldWB\Release.clicked
MCROrderEventSetup.find
MCRSalesOrderRecap.Control:SubmitButton.clicked
MCRSalesQuickQuote.Modified
MCRTmpPickingWorkbenchTrans.initFromSessionCriteria
MultilineString.POSDeveloperSupport
OriginalDocuments.insertDocument
PaymSchedCalc_Amount.createTransaction
PdsRebateFindAndCreate.findPdsRebateAgreementAndCreateClaim
PdsRebateFindAndCreate.findPdsRebateAgreementAndCreateClaim()
PdsRebatePaymentPost.insertRebateEntryForGrouping
PmfFormCtrl_BOM_BOMVersion.modifiedFormulaSize
PriceDiscAdmCheckPost.postJournal
ProdJournalCheckPostProd.postTransLedger
ProdJournalTransBOM.inventBatchId.validate
ProdMultiReportFinished.insert
ProdUPDCostEstimation.CreateProdBOM
ProdUpdReportFinished.updateBOMConsumption
ProdUpdStartUp.createJournals
ProjBegBalJournalTrans_CostSales.validateField, validateWrite
ProjBudgetManager.deleteBudgetLinesBeforeImportForRevs
ProjBudgetManager.getQuery
ProjBudgetRevisionManager.createBudgetLines
ProjBudgetTransactionManager.isOverrunAllowed
ProjectCommitmentFacade.updateProjectCommitmentsMap

REFACTORED METHODS
ProjectMainAccDimensionListProvider.populateMainAccountDimensionList
ProjForecastBudgetCopy.do_Cost
ProjForecastBudgetCopy.do_empl
ProjForecastBudgetCopy.do_onAcc
ProjForecastBudgetCopy.do_sales
ProjGroupChange.checkPostedTrxAccounts
ProjIntercompanyCustomerInvoiceCreator.createInvoiceLine
ProjInvoiceJournalPost.postCustVend
ProjInvoiceJournalPost.validateNoTax
ProjInvoiceProposalInsertLines.run
ProjJournalTrans.validateWrite
ProjPlanVersionCopyHierarchy.addProjPlanVersionFields
ProjPlanVersionCopyHierarchy.insertProjPlanVersionRecords
ProjPlanVersionCopyHierarchy.ProjPlanVersionCopyHierarchy
ProjPlanVersionsManager.importProjPlanVersionRecords
ProjPost.PostNeverLedger
ProjPost.PostTurnover
ProjPosting.updateDatasourceRanges
ProjTable.validateWrite
ProjTable.validateWriteServer
ProjTask.addTask
ProjValSetupEmplProj.ProjValSetupEmplProj.AddResourceButton.Click
ProjWBSDataEntityHelper.postInsertOperation
PurchAutoCreate_ReleaseFromAgreement.createLines
PurchInvoiceJournalPost.calcLastPurchPrice

REFACTORED METHODS
PurchPackingSlipJournalPost.updateSourceLineBeforePosting
PurchReqLine.defaultBuyingLegalEntity
PurchRFQCaseAutoCreate_PurchReq.calcRFQHeaderValues
PurchTable/InventDim/InventBatchId.modified
ReqTransPoMarkFirm.executeAction
ReqTransPOMarkFirm.CreateProdBOM
ReqTransPoMarkFirm.purchTablePostProcessing
ReqTransPoMarkFirm.setDeliveryDateAndPriceDisc
ReqTransPoMarkFirm.setGroupingIndicators
RequisitionPurchaseOrderGeneration.Create
RequisitionPurchaseOrderGeneration.createPurch
RequisitionPurchaseOrderGeneration.getVendors
ResReserveCapacity.getCapacityPercentage
RetailCreateLinesFromProductsToAdd.loadDiscountLines
RetailMassUpdateValidator.validateWriteOnInventModelGroupItem
RetailMediaAssociationHelper.populateMediaAssociationTable
RetailOENInfo.parseEmailTemplate
RetailPrintLabels.loadFromArgs
RetailPrintLabels.loadLines
RetailTransactionServiceOrders.createOrUpdateRetailOrderHeader
RetailTransactionServiceOrders.createOrUpdateRetailOrderLines
SalesConfirmJournalPost.createReportData
SalesFormLetter_Invoice.checkInvoicePrices
SalesInvoiceDP.setPackingSlipDetails
SalesInvoiceJournalPostBase.updateInventory

REFACTORED METHODS
SalesInvoiceJournalPostBase.updateInventoryFinancialForSalesInvoiceLine
SalesLine.CheckItemId
SalesLine.getInventQtyFromCWUnit
SalesLine.setInventDeliverNow
SalesLineType.validateWrite
SalesQuotationDP.createTaxLines
SalesQuotationDP.itemId
SalesQuotationLine.PriceDate
SalesQuotationTable.active
SalesTable-DataSource_mcrSalesTable-DataField_SourceId.modified
ShipOrderForm.POS.ChangeOriginOnShipOrders
SmabomDesignerCtrl.listInsertHistory
SmabomDesignerCtrl.treeSubstituteBOMonNode, treeDeleteNode, treeDeleteChildrenCollect
SMAServiceObjectrelation.jumpRefBOMTable
SubledgerJournalizerProjectExtension.createProjectActualCostDetail
SubledgerJournalizerProjectExtension.createProjectActualSalesDetail
SubledgerJournalTransferCommand.insertGeneralJournalAccountEntryRelated
SubledgerJournalTransferCommand.insertGeneralJournalAccountEntryRelatedDetail
SubledgerJournalTransferCommand.insertGeneralJournalAccountEntryRelatedDetail
SubledgerJournalTransferCommand.insertGeneralJournalAccountEntryRelatedSummarized
SubledgerJournalTransferCommand.insertGeneralJournalAccountEntryRelatedSummarized
SubledgerJournalTransferCommand.insertGeneralJournalEntryRelated
SuppItem.calcSuppItem
TaxProformaSpec.parmTaxSpec
TaxWithhold.postTaxWithhold

REFACTORED METHODS
TaxWithholdSlipDP_TH.createTaxWithholdSlipTmp
TaxWithholdSlipDP_TH.createTaxWithholdSlipTmp
TmsProcessXML_Base.readRateShipment
TMSRouteHelper.getShipDates
TradeLineNumberManager.checkLineNumber
TrvCreditCardReminder.mail
TrvCreditCardReminder.runQT
TrvExpenditureParticipantProvider.resolveFromDimensions
TrvExpenditureParticipantProvider.resolve
TrvExpenditureParticipantProvider.resolveProjectAuthorities
TrvExpenses.openSplitDetailsForm
TrvExpTable.validateSubmit
VendAgingReportController.getReportName
VendBalanceList.insertIntoTmpAccountSum
VendInvoiceInfoListPage.postInvoice
VendOutPaymRecord_Cheque.checkValues
VendVendorEntity/VendVendorV2Entity.processChangesForApproval
VestingID.Table: HRMCompVarAward
WhsContainerization.packTmpWorkLine
WhsControlBatchId.process
WHSPostPackingSlip.canShipConfirm
WHSPostPackingSlip.shipConfirmLoad
WHSProcessGuideStartChangeWarehouseStep.doExecute
WhsrfControlData.batchExistInLocation
WhsShipConfirm.tmsMultiLoadShipConfirm

REFACTORED METHODS
WHSSplitWork.handleOrignalWorkLine
WHSSplitWork.handleRemainingPickTrans
WHSSplitWork.processRemainingTransaction
WHSSplitWork.updateClosedPickTrans
WhsUnShip.cleanUpTOInventTransDims
WhsWarehouseRelease.createShipmentsForTransferOrders
WHSWorkCreate.createWorkInventTrans
WHSWorkCreate.createWorkTable
WhsWorkCreateProdPut.createReportFinished
WhsWorkCreateReceiving.createBatch
WHSWorkExecute.putAwayToLocation()
WHSWorkExecuteDisplay.buildInventoryStatus
WhsWorkExecuteDisplay.getNextFormState
WhsWorkExecuteDisplay.processTrackingDimDetails
WhsWorkExecuteDisplay.processVendorBatchDetails
WhsWorkExecuteDisplay.processWorkLine
WHSWorkExecuteDisplay.setBatchDetails
WhsWorkExecuteDisplayLoadItemReceiving.buildPOReceiving
WHSWorkExecuteDisplayLPReceiving.generateItemInfoForReceiving
WhsWorkExecuteDisplayPOLineReceiving.buildPOReceiving
WhsWorkExecuteDisplayPOLineReceiving.buildPOReceiving
WhsWorkExecuteDisplayReportAsFinished.displayForm
WHSWorkTable.satisfyDemandWorkLine
WmsArrivalCreateJournal.createWMSJournalTransFromArrivalDetails
WmsJournalCheckPostReception.returnOrderUpdate

REFACTORED METHODS
WmsOrderCreate.updateCreatewmsOrder
WorkflowHierarchyProviderHelperEventHandler.addDataSourceFieldsDelegate
WorkflowHierarchyProviderHelperEventHandler.loadLimits
WrkCtrScheduler_Prod.saveOperation

Other changes

The following additional changes have been made for extensibility.

- Convert queries where InventSumFields is used to SysDa.

Extensibility changes in Dynamics 365 for Finance and Operations version 8.1.3

2/18/2021 • 2 minutes to read • Edit Online

This is a list of extensibility features that were implemented in Dynamics 365 for Finance and Operations version 8.1.3. For more information about the schedule of changes that support extensibility, see Application extensibility plans.

Enumerations made extensible

These enumerations have been made extensible in this update.

ENUMERATION
AttributeDataType
BankDocumentBookType
BudgetPlanHCMReportGroupOption
CommitmentType
CommitmentType
CustVendNegInstStatus
CzAdvanceInvoiceStatus
DateTransactionDuedate
LedgerAllocationMethod
LedgerCovDocumentType
MCRFraudType
PercentHours
PerDayWeekMthQtYr
PrepaymentHandlingLayout_W
ProdStatusAll
TaxDirection
TaxType_IT

ENUMERATION
WHSWaveTemplateType

SQL operations made extensible

These SQL operations have been made extensible in this update.

OPERATION
InventTrans.deleteReturnTransOrigin
InventUpd_ChangeDimension.updateForceInventTrans
PriceDiscAdmCheckPost.updatePriceDiscTableRecords
ProdJournalCleanUp.deleteJournals
ProjBudgetManager.createBudgetCostForecast
ProjBudgetManager.createBudgetEmplForecast
ProjBudgetManager.createBudgetRevenueForecast
ProjBudgetManager.createBudgetSalesForecast
SubledgerJourFinalNetAmtEntryProvider.getFinalRelievingEntriesWithNetAmounts
SubledgerJourFinalRelieveEntryProvider.populateEntriesBySide
SubledgerJournalAccountEntryRelievingTmp.mergeJournalizationEntries
SubledgerJournalFinalReliever.findEntriesAlreadyRelieved
SubledgerJournalFinalReliever.findEntriesToRelieve
SubledgerJournalizer.loadFinalizeSubledgerJournalTmpDetail
SubledgerJournalizer.loadInterCompanyNonOffsetSubledgerJournalTmpDetail
SubledgerJournalizer.loadInterCompanyOffsetSubledgerJournalTmpDetail
SubledgerJournalizer.loadRelievingSubledgerJournalTmpDetail
SubledgerJournalizer.loadReversingSubledgerJournalTmpDetail
SubledgerJournalizer.loadYearEndSubledgerJournalTmpDetail
SubledgerJournalizer.summarizeJourAccountEntryDetailForRound

Metadata changes

These metadata changes have been made in this update.

OPERATION
Classes/CustVendSettle/classDeclaration.field modifier
Classes/LedgerPostingGeneralJournalController/transferLines.HookableAttribute
Classes/VendPaymentJournalDP.none
EcoResProductAttributeTranslationEntity.IsPublic
EcoResProductBarcodeEntity.Property.IsPublic
Enum/RDeferralsCalculatePeriod.Country region code
Enum/RDeferralsInitRetirementDate.Country region codes
Enum/RDeferralsInitWriteStartDate.Country region codes
Enum/RDeferralsInitWriteStartDate/EnumValue
Enum/RDeferralsInterval.Country region code
Enum/RDeferralsManualCalcType.Country region codes
Enum/RDeferralsMethod.Country Region Codes
Enum/RDeferralsPostValue.Country region codes
Enum/RDeferralsStatus.Country Region Code
Enum/RDeferralsTableGroupAllBook.Country Region Codes
Enum/RDeferralsTransType.Country Region Codes
Extended Data Types/AttributeValueFloat.No Of Decimals is Extensible
Increase Description in Project invoices/proposal lines for OnAccount and Expense line type
InventValue report now supports custom dimensions
Maps/AccountSumMap.Balance08,Balance08Cur,Balance09,Balance09Cur
Table/PurchTable.Visible = True
Table/VendUnrealizedRev/Field/ReversalDate.Allow edit.AllowEdit
Tables/DirPartyTable/Fields.AOS Authorization
Tables/VendSettlement/Relations/VendTrans.RelationshipType
Tables/WHSDocumentRoutingLine/Indexes/DocumentRoutingTablePrinterNameIdx

OPERATION

Refactored methods

These methods have been refactored to support extensibility.

REFACTORED METHODS
AssetJournal.populateLedgerJournalTrans
AssetPost.postToGeneralLedger
AssetProposalDepreciation.run
AssetTransfer.addLedgerVoucherTransObjects
AxSalesLine.setDefaultDimension
BankChequeCopy.fillTmpChequePrintout
BankChequeLayout.updateDesign
BankDepositSlip.modifyBankAccountTrans
BankJournalHeaderEntity.ValidateField
BankPositivePayExport.sendFileToUser
BankStatementValidate.validateDate
BankStmtISOAccountStatement.deleteStatement
BOM.defaultField
BOM.validateField
BOM.validateWrite
BomCalcItem.insertBOMCalcTable
BomCalcItem.insertBOMCalcTrans
BomCalcProd.calcCostSheet
BOMReportFinishMax.retrieveInventTable
BudgetCaculateBalance.getActualLedgerAmountsQuery
BudgetCaculateBalance.getOriginalBudgetQuery

REFACTORED METHODS
BudgetCaculateBalance.getRevisedBudgetQuery
BudgetSourceCollectionIntegrator.newBudgetSourceCollectionIntegrator
BudgetSourceIntegrator.newBudgetSourceIntegrator
ChequeController.ClassDeclaration
ChequeController.init
ContactPersonApplicationSuiteEventHandlers.initializedFromCommonEventHandler
CustBillOfExchangePostRemit.postSettlingStep
CustDirectDebitMandate.checkBankIBAN
CustDueReportDetailDP.updateCustDueReportDetailTmp
CustPostInvoiceJob.custPostInvoiceUpdate
CustSettleJournalizingEntries.createGeneratedEntries
CustSettleJournalizingEntries.getInterestOriginatingEntries
CustSettleJournalizingEntries.getOriginatingEntries
CustTransDetails.new
CustTransListDP.insertCustTransListTmp
CustTransOpenCashFlow.generateCashFlow
CustVendCheque.output
CustVendCreatePaymJournal_Vend.shouldAddCustVendTransOpen
CustVendEditTaxBranchHelper_TH.init
CustVendExchAdjTrans.initLedgerVoucher
CustVendSettle.reverseTax
CustVendTransReorg.paymentSchedSplit
CustVendTransReorg.post
CustVendTransReorg.reorganize
CustVendVoucher.setTransactionTxt

REFACTORED METHODS
CustWriteOff.createTaxJournalLines
DimDerJourRuleProjectTimesheetsExt.getDefaultDimensionAllocation
DirPartyRoleIsExternallyMaintained.setFieldRestrictions
EcoResDocumentAttachmentEntity.insertDatasourceDocuRef
EcoResProductNumberBuilderVariant.getEnumeratorForEnabledOrderedProductDimensions
EFDocMsgFormat_XmlSubmit_BR.createXmlDocumentFromEFDocument
EFDocumentXpath_BR.Not applied
ERclasses - class signature
FiscalDocParmDataCreatorInvTransfer_BR.initHeaderParmData
FiscalDocParmDataCreatorInvTransfer_BR.setinventTransferTableFiscalInfo
FiscalDocumentParmDataCreator_BR.initTaxTransParmDataFromTaxTrans
FiscalDocumentParmDataCreator_BR.setAccountingAmountOnFiscalDocumentLines
FiscalDocumentPost_BR.initFiscalDocument
FreeTextInvoiceDP.insertIntoFreeTextInvoiceHeaderFooterTmp
HcmWorkerTransition.createHcmEmployment
HrpExpireWorkerLimits.expireLimitRequest,expireApprovedLimit,getAuthorityBasis
InventMov_Sales.accountBalanceSheet
InventReleaseOrderPickingForm_Sales.bldInventReleaseOrderPickingTmp
InventTestAssociationTable.checkExecutionTime
InventTrans.insertReturnTransOrigin
InventTrans.updateMarkReqTransCov
InventTransferOrderCopying_BR.createTransferLines
InventTransferOrderCopying_BR.createTransferOrder
InventTransferUpdShip.updateInventTransferLine
InventUnusedDimCleanup.isCandidateInventDimIdTable

REFACTORED METHODS
InventUpd_ChangeDimension.updateTransSwitchDim
InventUpd_ChildReference.updateLessReceipt
InventUpd_ChildReference.UpdateMoreReceipt
InventUpd_Financial.updateFinancialIssue
InventUpd_Financial.updateStdCostPrice
InventUpd_Picked.updatePickMore
InventUpd_Registered.updateRegisterLess
InventUpd_Registered.updateRegisterMore
InventUpd_Reservation.updateReserveLess
InventUpdate.initializeInventTransToIssueListFromDatabase
InventUpdate.initInventTransToReceiveList
JmgJobBundleProdFeedbackForm.getTmpJobBundleProdFeedback
JmgPaySpecificationDP.Class declaration
JmgPostStandardSystem.getProjTransCostPrice
JmgPostStandardSystem.postIPCTime
JmgPostStandardSystem.postProjTime
JmgProfiles.bundleSlizeTime
JmgProfiles.insertTimeGapsPlannedAbs
JmgProfiles.sumPayEventsSec
JmgStampJournalTransfer.insertStampTrans
JmgTransferEvents.createPayEventsArray
JmgTransferEvents.insertEvents
JournalizingDefinitionManagerPurch.getDefaultJournalizingDefinition
LedgerAccrualTrans.post
LedgerAllocationBasisRules.getBasisAmount

REFACTORED METHODS
LedgerJournalCheckPost.checkJournal
LedgerJournalCheckPost.postJournal
LedgerJournalCheckPost.postTransV2
LedgerJournalCheckPost.updateInterCompanyJournal
LedgerJournalTrans.markedForSettlementError
LedgerJournalTrans.validateWrite_Server
LedgerJournalTransProject.checkProjId
LedgerJournalTransUpdate.updateInterCompany
LedgerJournalTransUpdateBank.updateNow
LedgerJournalTransUpdateCust.updateNow
LedgerJournalTransUpdateLedger.createTaxLinkForTaxTransfer
LedgerJournalTransUpdateLedger.updateNow
LedgerJournalTransUpdateVend.updateNow
LedgerTransPerJournalDP.insertForLedgerBase
LedgerTransPerJournalDP.insertVoucherDetails
LedgerTransPerJournalDP.processReport
LedgerVoucher.check
LedgerVoucherGroup.end
LedgerVoucherObject.addBalanceAdjustments
LedgerVoucherObject.allocateTransaction
LedgerVoucherObject.post
LedgerVoucherObject.updateBalances
LedgerVoucherTransObject.check
Markup.copy
McrPriceHistoryLine_Sales.initAndInsertRebate

REFACTORED METHODS
Method signature: Adding contextual information before doing Unit of Measure calculation
PdsRebateFindAndCreate.findPdsRebateAgreementLineAndCreate
PdsRebateFindAndCreate.tamFindBillBackAgreementAndCreateClaim
PriceDisc.findDisc
PriceDisc.findDiscAgreement
PriceDisc.findItemPrice
PriceDisc.findPrice
PriceDisc.findPriceAgreement
PriceDiscAdmCheckPost.runFromContract
ProdJournalCheckPost.postProdJournalTableBOM
ProdJournalCheckPostProd.postTransLedger
ProdJournalCreateBom.createSingleLineProdBOM
ProdTableCleanUp.deleteProductions
ProdUpdCostEstimation.costEstimateItems
ProdUpdCostestimation.updateSubProdTable
ProdUpdReportFinished.updateBomConsumption
ProdUpdStartup.UpdateBomConsumption
ProjAdjustment.getNewTotalCostAmount
ProjAdjustment.setHourCostPrice
ProjAdjustmentSplit.createNewTrans
ProjAdjustmentSplit.initializeTmpProjAdjustmentCreate
ProjAdjustmentUpdate_Post.post
ProjAdjustmentUpdate_Post.postCost
ProjAdjustmentUpdate_Post.postItem
ProjBudget.queryProjBudgetLineCost / ProjBudgetLineCost(DataSource)

REFACTORED METHODS
ProjBudget.queryProjBudgetLineCost / ProjBudgetLineRevenue(DataSource)
ProjBudgetManager.createBudgetFromForecastModel
ProjCategoryLookup.buildQueryTsTimesheetLine
ProjInvoiceChoose.main
ProjInvoiceJournalCreate.initJournalHeader
ProjInvoiceJournalPost.createProjInvoiceSalesLine
ProjInvoiceProposalInsertLines.doCost, doEmpl, doItem, doOnAccount, doRevenue, doSalesLine
ProjInvoiceProposalPeriodic.Validate
ProjJournalTrans.setHourCostPrice
ProjJournalTrans.setHourPrices
ProjJournalTrans.setHourSalesPrice
ProjPlanVersionsManager.CopyHierarchy
ProjPost.postCost
ProjPostCostProposalSale.projTransUpdate
ProjPostEmplProposalSale.projTransUpdate
ProjPostItemProposalSale.projTransUpdate
ProjPostRevenueProposalSale.projTransUpdate
ProjTable.createSalesTable_ItemReq
ProjTable.editSubProjects
psaContractLineInvoiceDP.insertTmpPSAContractLineInvoice
PsaGenerateQuotationLines.createSalesQuotationLines
PsaManageInvoiceDP.insertTmpPSAManageInvoice
PSAProjInvoiceDP.processLinesFromInvoiceJournal
PSAProjInvoiceTaxTmp.insertPSAProjInvoiceTmpForTax
PSAProjPostEmplIndirectProposal.indirectCreditAccountTurnover

REFACTORED METHODS
PurchCreateFromSalesOrder.autoCreatePurchOrder
PurchCreateFromSalesOrder.checkLine
PurchCreateFromSalesOrder.main
PurchCreateFromSalesOrder.querySalesLine
PurchCreateFromSalesOrder.run
PurchFormletterParmDataInvoice.copyMarkupFromPurchOrder
PurchFormletterParmDataInvoice.createInvoiceHeaderFromTempTable
PurchFormletterParmDataInvoice.createLineAsset
PurchFormletterParmDataInvoice.selectChooseLines
PurchInvoiceJournalPost.postInventory
PurchLineType.initReleasedProductSpecificDefaulting
PurchOrderLineSourceDocumentLineItem.initMonetaryAmountValue
PurchReApprovalPolicyRule.evaluatePolicy
PurchRFQAcceptJournalPost.updateSourceTable
PurchRFQSendJournalCreate.createOrUpdateRFQLine
PurchTableForm.main
rDeferralsJournal.createTrans
rDeferralsProposalReceipt.createJournalLines
rDeferralsProposalRetirement.createJournalLines
rDeferralsProposalWritingOff.createJournalLines
rDeferralsTableMethodIterator.new
ReqDemPlanForecastAggregator.deaggregate
ReqDemPlanImportForecastService.insertDemandForecast
ReqIntercompanyDemand.initReqTransFromIntercompanyReqPO
ReqPO.Update

REFACTORED METHODS
ReqTrans.updateBOMQty
ReqTransPoMarkSumUp.updateSumUp
RequisitionReleaseStrategy.runAutoPurchOrderGeneration
RetailTransactionSalesTransMark.findInventDimFromWorkingTable; and more please review attachment
RetailTransactionSalesTransMark.updateTransactionSalesLine_InventDimId
RetailTransactionServiceOrder.createOrUpdateRetailOrderLines
RetailTransactionServiceOrders.cancelCustomerOrder
RunBaseMultiParm.initFromForm
SalesCopying.callerSalesTable.returnItem
SalesFormletterParmDataInvoice.createBasedOnPackingSlip
SalesInvoiceController.initFormLetterReport
SalesInvoiceController.parmRunOnBlockMode_TH
SalesInvoiceController.PrintMgmtPrintSettingDetail
SalesInvoiceDP.addProductDimensionsToInventDim; tmpTaxWorkTrans;
SalesInvoiceDP.initInventDimData
SalesInvoiceDP.populateSalesInvoiceHeaderFooterTmp
SalesInvoiceDP.printBackorders
SalesInvoiceDPBase.createData
SalesInvoiceDPBase.init
SalesInvoiceJournalPost.postCustVend
SalesInvoiceJournalPost.postLine
SalesLineType.pmfValidateBatchId
SalesLineType_ReturnItem.validateField
SalesPackingSlipController.initFormLetterReport
SalesPackingSlipController.parmRunOnBlockMode_TH

REFACTORED METHODS
SalesPackingSlipDP.checkPrintLineHeader
SalesPackingSlipDP.initializeInventDimReportSetup
SalesPackingSlipJournalPost.updateInventory
SalesParmTable.createPaymentSched
SalesPurchOperationTypeController_BR.getReferenceLookup
SalesQuotationDP.initializeInventDimReport
SalesQuotationProjLinkWizard.endUpdate
SalesQuotationTableType.disableFieldsIfCustomerAccountNotSpecified
SingleReturn.POSDeveloperSupport
SubledgerJournalizer.addRelievingAccountingDistributions
SubledgerJournalizer.fillPreviewTmpSummaryWithRounding
SubledgerJournalizer.loadaccountingDistributionTmp
SubledgerJournalizer.loadFinalizeSubledgerJournalTmpDetail
SubledgerJournalizer.loadInterCompanyNonOffsetSubledgerJournalTmpDetail
SubledgerJournalizer.loadInterCompanyOffsetSubledgerJournalTmpDetail
SubledgerJournalizer.loadIntercompanySubledgerJournalTmpDetail
SubledgerJournalizer.loadRelievingSubledgerJournalTmpDetail
SubledgerJournalizer.loadReversingSubledgerJournalTmpDetail
SubledgerJournalizer.loadStandardSubledgerLedgerJournalTmpDetail
SubledgerJournalizer.loadYearEndSubledgerJournalTmpDetail
SubledgerJournalizer.previewSummarizeJournalAccEntryDetail
SubledgerJournalizer.recordSubledgerJourAccEntriesForRounding
SubledgerJournalizer.recordSubledgerJournalAccountEntries
SubledgerJournalizer.summarizeJournalAccountEntryDetail
TAMTradePromotion.ValidateFundCostLevel

REFACTORED METHODS
taxBooksection.checkNumberSequenceSetup
TaxCalculationAdjustment.adjustBaseForTaxIncluded
TaxJournalSpec.parmTaxSpec
TaxProjInvoice.new
TaxReport770TransHandler_IT.transferTaxWithholdTrans
TaxReport770Validate_IT.validateVendors
TaxSplitPaymentPost_IT.creareReverseTaxTrans
TaxWithhold.postTaxWithhold
TaxWithholdSpecialDP.createTaxWithholdSpecialTmp
TmpProjAdjustmentCreate.createFromAdjustment
TmpProjAdjustmentCreate.fieldModifiedProjId
TmpProjAdjustmentCreate.setDimension
TmpProjAdjustmentCreate.setHourCostPrice
TransactionReversal_Asset.reversalBook
TransactionReversal_Cust.createAuxiliaryCustTrans
TransactionReversal_Ledger.createGeneralJournal
TrvExpTrans.setTaxGroup
TSTimesheetEntry.setFocusOnDayIndex
TsTimesheetFavorites.createTimesheetLines
TsTimesheetFavorites.validateWrite
TSTimesheetTable.checkHours
VendAccountStatementIntDP.insertVendAccountStatementIntTmpRil
VendOutPaymControlController.Insert
VendPaymentJournalDP.insertDataFromSpecTrans
VendRequestAddVendor.init

REFACTORED METHODS
WhsContainerization.createNewContainer
WHSLoadLine.updateQtyLeftToLoad
WHSLoadTableAssignOriginInfo.assignFromFirstLoadLine
WHSLocationDirective.findPickPutLocation
WhsPostPackingSlip.alterLoadLine
WHSProdTable.pickBatchQtys
WhsWorkCreate.checkMaximums
WHSWorkCreateProdPut.insertProdParmforProdItem
WhsWorkExecuteDisplay.buildAboveLocationDimensions
WhsWorkExecuteDisplay.buildPick
WhsWorkExecuteDisplay.getNextFormState
WhsWorkExecuteDisplay.processWorkLine
WHSWorkExecuteDisplayCycleCount.buildCycleCount
WhsWorkExecuteDisplayCycleCount.buildFinish
WhsWorkExecuteDisplayPOItemReceiving.buildPOReceiving
WhsWorkExecuteDisplaySpotCycleCounting.displayForm
WhsWorkExecuteDisplaySystemGrouping.displayNextForm
WhsWorkExecuteDisplaySystemGrouping.getWorkIdFromFieldName
WhsWorkExecuteDisplayUserDirected.displayForm
WhsWorkExecuteDisplayUserGrouping.displayForm
WhsWorkExecuteDisplayValidateUserDirect.displayForm
WhsWorkExecuteDisplayValidateUserDirect.validateUserDirectWorkExists
WHSWorkTable.executeWorkLinesInit
WmsJournalCheckPostReception.returnOrderUpdate
WmsPickingList_OrderPickDP.initQueryWMSOrderTrans

REFACTORED METHODS
WmsPickingList_OrderPickDP.insertIntoTempTable
WmsPickingList_OrderPickDP.orderQty
WmsPickingList_OrderPickDP.orderUnit
WmsPickingList_OrderPickDP.printDocumentHeader
WmsPickingList_OrderPickDP.setWMSPickingList_OrderPickTmpTemplate

Other changes

The following additional changes have been made for extensibility.

- Updated aging and balance list classes and forms to support the ability for customizations to increase the number of calculated aging buckets.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Extensibility changes in Dynamics 365 for Finance and Operations version 8.1.2

2/18/2021 • 2 minutes to read • Edit Online

This is a list of extensibility features that were implemented in Dynamics 365 for Finance and Operations version 8.1.2. For more information about the schedule of changes that support extensibility, see Application extensibility plans.

Enumerations made extensible

These enumerations have been made extensible in this update.

ENUMERATION
DimensionHierarchyType
DirPartyType
DirPersonMaritalStatus
PrintPostCancel
INSAffiliate
LedgerJournalLinesDisplayOption
LedgerTransPerJournal
ProjDortValue
ProjPaymentStatus
RequisitionReleaseType
RetailPOSSeedDataType
SysDimension
TrvExpType
TSTimesheetEntryGridView
VendProspectiveVendorRegistrationWizardTab

Metadata changes

These metadata changes have been made in this update.

OPERATION
DataEntities/LedgerJournalNameEntity/Fields/DeleteLinesAfterPosting.Allow Edit
DataEntities/LedgerJournalNameEntity/Fields/DeleteLinesAfterPosting.AllowEditOnCreate
Forms/AssetProposalDepreciation/Design/Tab/ParametersTabPage/ParametersGroup/SummarizedDepreciationControl.Value
Data manipulation method not raising event: PriceDiscAdmDeleteTradeAgreements.run
Data Types/Base Enums/WHSReverseWorkMode.Label
DataEntity smmProspectEntity is not public
DataEntityView/GeneralJournalAccountEntryEntity.PublicCollectionName, PublicEntityName and IsPublic
Enum/HcmPersonGender/EnumValue/NonSpecific.Label
LedgerJournalEngine.shouldOverwriteAmountWithSettledAmount
Query/LedgerDerivedFinHierarchy/EcoResCategoryHierarchyRole_1/Ranges/NamedCategoryHierarchyRole.Range/Value
Table/TSTimesheetLine/TableFieldEnum
Tables/InventTransPosting.DateVoucherTransIdx
Update unique indexes in pricing tables for project

Refactored methods

These methods have been refactored to support extensibility.

REFACTORED METHODS
AgreementConfirmationDP.getAgreementLine
AgreementConfirmationDP.getAgreementLineHistory
AssetBook.initDepreciationProfile
AssetPost.createTrueUpDepreciation
AssetPost.reduceLastDepreciation
Bank_CA.checkBankAccount
Bank_CA.checkBankRegNum
BankReconMatchingRuleAutoProcessor.doProcessMatchRule
BankReconMatchingRuleAutoProcessor.performMatchAction

REFACTORED METHODS
BomCalcItem.calcCostSheet
ChequeCopy.printCheque
ChequeDP.fetch
Coupons.AddCouponTrigger
Cust.initLedgerVoucher
CustAgingReportDP.heading
CustBalanceList.constructAgingCalculation
CustCollectionLetterCreate.createJournal
CustCollectionLetterCreate.run
CustCollectionLetterPost.updateQuery
CustCollections.showAgingIndicator
CustCollectionsExcelStatement.setTransactionWorksheetHeader
CustDirectDebitMandate.lookupReference
CustDirectDebitMandate.validateMandate
CustDirectDebitMandate.validateMandate
CustFreeInvoiceCorrection.createAdjustingCorrectedInvoice
CustFreeInvoiceCorrection.createTaxes
CustFreeInvoiceCorrectionPost.postAdjustingInvoice
CustFreeInvoiceCorrectionPost.validate
CustinvoiceLine.insert
CustInvoicePrintJob.buildQueryForFreeText
CustInvoicePrintJob.processFreeText
CustOpenTrans.editMarkTrans
CustOpenTransReverse.markTrans
CustOverPaym.run

REFACTORED METHODS
CustPackingSlipJour.printJournal
CustPaymEntry.hasMultipleOpenTransReferences
CustPaymEntry.isInvalidOpenTransReference
CustPostInvoice.allocateNumAndVoucher
CustPostInvoice.createJournalHeader
CustRecurrenceInvoicePostService.postRecurrenceInvoice
CustSettlementPriorityProcessing.initCustTransOpen
CustStatistics.TmpStatPer.linkActive
CustTable.createRecord
CustTable.CustTable_DS/fields/CustGroup/modified
CustVendCheque.checkDataOk
CustVendCheque.output
CustVendChequeSlipTextCalculator.getMaxSlipLines
CustVendChequeSlipTextCalculator.getUnprintableReportArea
CustVendCreatePaymJournal.runPaymentProposalGenerationProcess
CustVendCreatePaymJournal.runPaymentProposalGenerationProcess
CustVendOpenTransManager.createTaxWithholding
CustVendPaymProposal.addCustVendTransOpen
CustVendReversePosting.restoreCustVendTransOpen
CustWriteOff.calcSalesTaxOnOpenTrans
CustWriteOff.generateSummarizedTmpTaxTrans
DataEntityView/ExpenseJournalLineEntity.DataEntityView/ExpenseJournalLineEntity
DirPartyPostalAddressFormHandlerExt.onUpdateTransactionCaller_delegate
Extensible class method: PriceDisc.mcrPriceDiscTableFound
FBSpedFileCreator_Contabil_BR.createRecordI052

REFACTORED METHODS
FiscalDocumentDate_BR.lastIssueDateForSeries
HrpSigningLimitPolicyUtil.createDefaultLimit
HrpSigningLimitPolicyUtil.insertJobOrCompensationRule
HrpSigningLimitPolicyUtil.private RefRecId checkLimitAgreementDetail(HRPTmpLimitAgreementRule _tmpLimitAgreementRule,HRPAuthorityBasis _authorityBasis)
HrpWorkerLimit.private recId getAuthBaseRecId(HRPAuthorityBasis _authBasis, RefRecId _positionId)
InterCompanySyncPurchTableType.setSalesTableData
InventCountCreate_Base.doCountingBasedOnCountCode
InventMov_Purch.updateAutoLossProfit
InventMov_Purch.updateLedgerFinancial
InventMovement.addLedgerPhysicalAmounts
InventMovement.addLedgerVoucherRevenueTransactionAmountsForFinancialUpdate
InventMovement.addLedgerVoucherRevenueTransactionAmountsForPhysicalUpdate
InventMovement.addLedgerVoucherTransactionAmountsForFinancialUpdate
InventMovement.addLedgerVoucherTransactionAmountsForPhysicalUpdate
InventMovement.checkUpdatePhysical
InventMovement.processLedgerPhysicalAmountList
InventMovement.setAutoReserving
InventMovement.setCostAmountPhysical
InventMovement.updateLedgerAdjust
InventMovement.updateLedgerFinancial
InventOnhandReserve.updateReserveLot
InventUpd_Estimated
InventUpd_Estimated.updateFieldsChange
JmgPayEventsExport_Std.run
JmgStampJournalTable.approve

REFACTORED METHODS
JmgStampJournalTable.transfer
LedgerAccrualTrans.post
LedgerAllocationBasisRules.createGeneralJournalAccountEntrySumQuery
LedgerAllocationController.allocateAmounts
LedgerAllocationProcessRequest.allocate
LedgerJournalCheckPost.checkJournal
LedgerJournalCheckPost.postJournal
LedgerJournalDistribute.createNewJournal
LedgerJournalEngine.calculateTaxForCompleteJournal
LedgerJournalEngine.initValue
LedgerJournalTable.deleteAllLines
LedgerJournalTrans.deleteTaxUncommitted
LedgerJournalTransDaily.LedgerJournalTrans.AmountCurCredit.validate
LedgerJournalTransDaily.LedgerJournalTrans.AmountCurDebit.validate
LedgerJournalTransType.validateVoucher
LedgerJournalTransUpdate.updateIntercompany
LedgerJournalTransVendPaym./Forms/LedgerJournalTransVendPaym/Design/ActionPane(ActionPane)/ButtonGroup(ButtonGroup)/buttonCreatePayment(MenuFunctionButton)/Clicked
LedgerTransListReportHelper.buildFieldMap
LedgerTransPerJournalDP.insertForLedgerBase
LedgerVoucherObject.checkBalance
LedgerVoucherObject.checkBalanceRound
LogisticsLocationFormHandler.callerResearch
LoyaltyCardBlance.MPOS_ExtensibleViews
Macros.InventSumFields
MainAccount.DimensionAttributeValue_ds/dimensionAttributeValueIsSuspended

REFACTORED METHODS
NumberSeqModuleProject.loadModule
PcSourceDocumentLineUtility.initialize
PdsRebateFindAndCreate.findPdsRebateAgreementAndCreateClaim + run
PriceDisc.findPriceAgreement
PriceDisc.FindPriceAgreement.mcrPriceDiscTablefound
PriceDiscResultFields.NA
ProdJournalBOM.insertJournalCreate
ProjAdjustment.splitLine
ProjAdjustmentSplit.calculateQty
ProjAdjustmentSplit.getNewTotalSaleAmount
ProjAdjustmentUpdate.newPostAdjustment
ProjAdjustmentUpdate.run
ProjAdjustmentUpdate.transCostNew / transEmplNew / transItemNew methods
ProjAdjustmentUpdate.transItemNew
ProjAdjustmentUpdate.updateAdjusted
ProjBudgetImport.SourceType - modified
ProjBudgetRevision.updateGridHelper
ProjectPosting.getProjectLedgerDimension
ProjForecastEmpl.initValue
ProjFormletterParmData.updateQueryBuild
ProjGrant.canSubmitToWorkflow
ProjInvoiceChoose.doCost
ProjInvoiceChoose.doEmpl
ProjInvoiceChoose.doItem
ProjInvoiceChoose.doOnAccount

REFACTORED METHODS
ProjInvoiceChoose.doRevenue
ProjInvoiceChoose.doSalesLine
ProjInvoiceChoose.psaAddEndDateToProposalJour
ProjInvoiceEditLines.Choose.clicked
ProjInvoiceEditLines.closeOk
ProjInvoiceProposalCreateLines.modifiedTransFilter
ProjInvoiceProposalCreateLines.run
ProjInvoiceProposalCreateLines.runSalesLineQuery
ProjInvoiceProposalInsertLines.doSalesLine
ProjInvoiceProposalInsertLines.setProjProposalJour
ProjInvoiceTable.createProposalJour
ProjLedgerUpdate.insert
ProjListTransDP.insertTmpTable
ProjPostItemPackingSlip .projTransCreate
ProjPostItemTransCost_Adj.projTransUpdate
ProjSplitBill.maxAllowedByLimits
ProjStatusTypeRule.enableRule
ProjTable.isCustomerTransferNeeded
ProjTableType.validateWrite
ProjValCheckTrans.validateMandatory
PsaProjAndContractInvoiceController.runPrintMgmt
PSAProjRetainerInvoicing.createTrans
PSAProjRetainerInvoicing.run
PurchAutoCreate_PurchReq.getPurchLineName
PurchAutoCreate_Sales.createLine

REFACTORED METHODS
PurchCopying.updatePriceDiscLineChangePolicy
PurchCreateFromSalesOrder.run
PurchCreateOrder.PurchTable.write
PurchEditLines.Choose_Button.clicked
PurchEditLines.run
PurchFormLetter.prePromptInit
PurchFormLetter.reSelect
PurchFormLetter::main
PurchFormletterParmDataInvoice.reSelectLines
PurchInvoiceJournalCreate.allocateNumAndVoucher
PurchReqAddItem.N/A: Variable Change, not Method
PurchRFQCaseTable.isCalledFromPurchRFQCTListPageProject
PurchTable.ConvertCurrencyCode
PurchTable.create
PurchTable.create (PurchTable datasource)
PurchTableType.validateDelete
ReqCalc.actionCalcItem
ReqCalc.covCalcDim
ReqCalc.covCodeQtyMinMax
ReqCalc.covCreatePlannedOrder
ReqCalc.covCreateSafetyInvent
ReqCalc.createSafetyInvent
ReqCalc.createSafetyInventKey
ReqCalc.deleteTransactionAndCoverage
ReqCalc.setParameters

REFACTORED METHODS
ReqCalc.writeInventSum
ReqTransCache.listCovDimSorted
ReqTransPoMarkFirm.create
RequisitionPurchaseOrderGeneration.updateEmptyVendAccountsForManualCreation
RequisitionPurchaseOrderGeneration.validatePurchReqLine
RetailInternalOrganization.insert
RetailKitAssemblyOrder.createOrUpdateBOMJournal
RetailKitAssemblyOrder.createOrUpdateBOMJournalLine
RetailStatementPost.postRetailSpecific
RetailStoresToDeploy.setAllowEditTrue
RetailTransactionSalesTransMark.findInventDimIdFromWorkingTable
RetailTransactionSalesTransMark.populateTransactionSalesLineWorkingTable
RetailTransactionServiceOrders.cancelCustomerOrder
RetailTransactionServiceOrders.createCustomerOrder
RetailTransactionServiceOrders.createLedgerJournalTransForPayment
RetailTransactionServiceOrders.createRetailOrderPayment
RetailTransactionServiceOrders.invoiceSalesOrder
RetailTransactionServiceOrders.settleCustomerOrder
SalesCopying.canClose
SalesCreateOrder.updateDeliveryAddress
SalesFormLetter.main
SalesFormLetter.mainOnServer
SalesFormLetter.reSelect
SalesInvoiceJournalCreateBase.createJournalHeader
SalesInvoiceJournalPostBase.postLine

REFACTORED METHODS
SalesInvoiceJournalPostBase.updateInventory
SalesLine.createLinesFromTmpFrmVirtual
SalesLine.runPriceDiscPolicyDialog
SalesLineType_ProjectSales.canBeInvoiced
SalesPurchLine.setPriceAgreement
SalesPurchLineInterface.setPriceAgreement
SalesPurchLineInterface.setPriceDisc
SalesQuotationEditLinesForm method createParmLine
SalesQuotationListPageInteraction.linkActive
SalesQuotationProjLinkWizard.endUpdate
SalesQuotationTable.convertCurrencyCode
SalesQuotationTable.modified (SalesQuotationLine_ItemId form control)
SalesQuotationTableType.numberSeqFormHandlerQuotationId
SalesQuotationTransferToProject.createForecastOnAcc
SalesQuotationTransferToProject.createProject
SalesTable.convertCurrencyCode
SalesTable.modified
SalesTable.updateDeliveryAddress
SmaServiceFunctionLine.getFromDialog
smmBusRelTable.updateCustTable
smmBusRelTable.updateVendTable
SourceDocumentBalanceProvider.calculateEncumberedAmount
Table/MyAddressBook.xds
Table/TrvExpTrans.update
Tax.allocateInTaxWorkTrans

REFACTORED METHODS
TaxCalculationJournal.saveTaxTransfer
TaxCashDisc.calcAndInsertTaxes
TaxData.find
TaxInventTransferInvoice_BR.post
TaxReversePrePayment.calcPostAndInsertTaxes
TaxReverseTax.insertTaxWorkTrans
TaxReverseTax.newTrans
TaxSettlement.retailCalcAndInsertTaxes
TaxWithHold.createTaxWithholdTrans
TaxWithhold.postTaxWithhold
TransactionReversal.updateTaxTrans
TransactionReversal_Vend.reversal
TransactionTxt.setKey1
TransactionTxt.setKey2
TransactionTxt.setKey3
TrvExpTrans.insertPerDiemDataLines
TrvPbsMainDataLines.clicked
TrvPostExpenseHeader.postCustVendTransactions
TSTimesheetTrans.getCostPrice
VendOutPaym_Cheque.generatePaymentLines
VendOutPaym_RBC.generatePaymentLines
VendOutPaymRecord_RBC_Credit.fillField03
VendOutPaymRecord_RBC_Credit.fillField07
WhsControlItemId.populate
WHSCycleCountCreatePlan.insertWorkLine

REFACTORED METHODS
WHSLoadLineAllocationProcessor.validateBatchDisposition
WhsLoadLineUpdater.initLoadLine
WHSMobileAppServiceXMLTranslator.createXML
WHSPack.packFromScanningFields
WhsrfControlData.allowMixedBatch
WhsrfControlData.allowMixedItem
WHSRFControlData.processLegacyControl
WhsWorkExecuteDisplay.buildGetVendBatchDetails
WHSWorkExecuteDisplay.buildLPControlFromPass
WHSWorkExecuteDisplay.buildPORecTrackingDimensions
WHSWorkExecuteDisplay.buildRemainingReceiptQtyCurrentLPLabel
WHSWorkExecuteDisplay.buildTrackingDimensions
WHSWorkExecuteDisplay.processWorkLine
WHSWorkExecuteDisplay.setBatchDetails
WhsWorkExecuteDisplayClusterPicking.clusterCompleted
WhsWorkExecuteDisplayMenu.buildMenu
WHSWorkExecuteDisplayPOReceiving.displayForm
WHSWorkExecuteDisplayUserDirected.displayForm
WhsWorkExecuteDisplayWarehouseTransfer.displayForm
WrkCtrScheduler_Proj.insertOrder

Other changes

The following table lists additional changes that have been made for extensibility.

CHANGE

- Create a SysQueryUpdateRecordSet class in AppCommon.
- Enable percent controlled for a catch weight item.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Extensibility changes in Dynamics 365 for Finance and Operations version 8.1.1

2/18/2021 • 2 minutes to read • Edit Online

This is a list of extensibility features that were implemented in Dynamics 365 for Finance and Operations version 8.1.1. For more information about the schedule of changes that support extensibility, see Application extensibility plans.

Enumerations made extensible

These enumerations have been made extensible in this update.

ENUMERATION
BankCodeType
CountryRegionType
MainAccountDimensionListProviderType
ProdSchedulingSortType
ProjAccountTypeSales
ProjBudgetBalancesGroupByOptions
ProjListStateType
ProjStatementType
SalesDeliveryDateControlType

Refactored methods

These methods have been refactored to support extensibility.

REFACTORED METHODS
[Extensibility] Method signature change: WHSWorkExecuteDisplayListWork.displayListWorkStep
[Extensibility] Refactor WhsWorkExecuteDisplayAdjustOut to ProcessGuide framework
AssetJournal
AXSalesQuotationTable.setQuotationId
BankStatementBankAccountIdentify.searchBankAccountTable

REFACTORED METHODS
BankStatementValidate.doValidate
BankStatementValidate.validateDate
BankStatementValidate.validatePeriodGap
BankStatementValidate.validatePeriodOverlap
BOM.validateWrite
BomCalcDialog.updateBomRoute
BOMCalcItem.createBomCalcItemAndAddToListBom
BOMCalcTable.transferToSalesLine
BOMCalcTable.transferToSalesQuotationLine
BomConsistOf.init
BomLevelCalc.loadDependencies
BOMReportFinishMax.init
BOMReportFinishMax.update
BOMReportFinishMax.updateBOMId
BudgetTransactionManager.checkBudgetTransactionNumberSequence
Commission.run
Cust/VendTableChangeProposalApply.apply
CustAccountStatementExtController.runPrintMgmt
CustDebitCreditNoteDP.insertForQuantity
CustDebitCreditNoteDP.insertForValue
CustInterestJour.findCustUnPostedInterestNote
CustOpenTrans.editMarkTrans
CustPackingSlipJour.PrintJounal
CustTable.openInvoiceBalanceMST
CustTable.openinvoiceBalanceMSTDoc

REFACTORED METHODS
CustTable.openPaymentBalanceMST
CustTable.openPaymentBalanceMSTDoc
CustTable.openPaymentBalanceMSTDue
CustVendCreatePaymJournal_Vend.searchTransactions
CustVendDisputeHelper.update
CustVendPaymProposalTransferToJournal.ClassDeclaration
CustVendPaymProposalTransferToJournal.updateSpecTransSet
CustVendPaymProposalTransferToJournal.updateSpecTransSingle
CustVendPrePaymentReversal.construct
CustVendSettle.settleForDifferentProfilesOrPrepayment
CustVendSumForPaym.run
CzCustPostAdvanceInvoice.run
DirPartyFormHandler.manageFields
EcoResProductCreate.close
EcoResProductCreate.templateRecords2Controls
EcoResProductDetailsExtended.InventTable.validateWrite
EssPersonSigningLimits/FormDataSourceRoot/HRPLimitRequestApproved.executeQuery
FormletterJournalPost.postLineDiscount
FormLetterParmData.updateQueryDocumentRanges
FormletterService.run
FormletterServiceBatchTaskManager.createFormletterParmDataTasks
FormletterServiceBatchTaskManager.createFormletterServiceTasks
FormletterServiceMultithread.newFormletterServiceMultiThread
FreeTextInvoiceController.preRunModifyContract
FreeTextInvoiceController.runPrintMgmt

REFACTORED METHODS
GeneralLedgerExtension.validateReferenceNumber
InterCompanyPost.formLetterCollect
InterCompanyPost.formLetterCollect
InterCompanySyncPurchLineType.createOrUpdateSalesLine
InterCompanySyncPurchLineType.synchronizeInTradeCompany
InterCompanySyncSalesLineType.classDeclaration
IntrastatTransfer.updateQuery
InventBatch.insert
InventBatchConsuptionValidator.ValidateExpiryDate
InventDimCtrl_Frm_OnHand.modifyQueryBasedOnDatasourceName
InventItemBarcode.validateWrite
InventItemPrice.init
InventItemPriceSim.moveSimulatedToCurrent
InventMov_Transfer.updateLedgerFinancial
InventMovement.costValueChanged
InventMovement.updateReservation
InventOnhandReserve.ReserveLine.clicked
InventQualityManagementCreate.createPerQualityAssociations
InventQualityManagementCreate.createPerQualityAssociations
InventQualityOrderValidate.main
InventShelfLifeCriteria.initFromMovement
InventSplitTrans.check
InventTableModule.initFromInventItemPriceSim
InventTableModule.update
InventTrans.setSumAmount

REFACTORED METHODS
InventTrans.updateSumUp
InventTransferOrders.InventBatchId.validate
InventTransferupd.createInventTransferJourLine
InventTransferUpd.createInventTransferJourLine
InventTransPick.ctrlUpdate.clicked
InventTransPick.InventDim.InventBatchId.Validate
InventTransPick.TmpInventDim.InventBatchId.validate
InventTransRegister.InventDim.InventBatchId.validate
InventTransRegister.TmpInventDim.InventBatchId.validate
InventTransWMS_Register.updateInventFromMovementServer
InventTransWMS_Register.updateInventFromMovementServer
InventUpd_Arrived.updateArrivedMorer
InventUpd_FinancialLite.updateTrans
InventUpd_Physical.displayErrorsIfIssueQtyGreaterThanPhysical
InventUpd_Physical.updateMovementBasedOnPhysicalQty
InventUpd_Picked.updatePickLess
InventUpd_Reservation.updateReserveMore
InventUpd_WHSReservation.updateReserveMore
InventUpdate.updateDimReserveChange
InventValueReportInit.initInstrumentation
JmgJobBundle.loadActiveJobs()
JmgJobBundle.private void loadActiveJobs
JmgJobBundleProjStartupForm.getTmpJobBundleProjStartup
JmgJobBundleProjStartupForm.onClose
JmgJobBundleProjStartupForm.validateCategoryId

REFACTORED METHODS
JmgPayAdjustment.insertAdjustment
JmgPieceRateCalc.calcPieceRate
JmgPieceRateCalc.insertEvents
JmgPostStandardSystem.createReportFinishedJournal
JmgProfileSpec.promptForAbsence
JmgStampJournalTrans.insert
JmgStampJournalTrans.update
JmgTransaction_Proj.postChange
JmgTransaction_Proj.postChange
LedgerAllocationController.allocateAmounts
LedgerAllocationRequest.closeOk
LedgerAllocationRequest.run
LedgerExchAdj.calculateAdjustments
LedgerExchAdj.constructTargetToSourceMap
LedgerJournalCheckPost.postJournal
LedgerJournalEngine.findSettledAmount
LedgerJournalTransUpdateVend.checkVoucher
LedgerParameters/FormDataSourceRoot/RDeferralsParameters.init
LedgerPostingGeneralJournalController.getLineValues
LedgerPostingGeneralJournalController.transferReferences
LedgerVoucher.check
LedgerVoucherTransObject.check
LedgerVoucherTransObject.check
MainAccount.init
MainAccount.MainAccount_ds/write

REFACTORED METHODS
MainAccount.MainAccountLegalEntity_DS/legalEntityIsSuspended
MainAccountTemplate.rolldownChanges
Markup.resolveOrigQty
MarkupAdjustment.run
MarkupAllocation.calculateValueNow
MarkupAllocation_VendInvoiceTrans.dialog
MarkupCopy.copyFromPurchOrder
MarkupTrans.checkKeep
Method signature change
OMLegalEntity.init
OMorganizationHierarchy.updatePreviewPane
PdsBatchAttribReserve.ReserveLine.clicked
PdsBatchAttributesInput.init
PdsRebateFindAndCreate.private void calculateSums()
PdsRebateFindAndCreate.protected void createZeroRebate(PdsRebateAgreement _pdsRebateAgreement)
PdsRebateFindAndCreate.resetTransSums
PdsResetDispositionStatus.main
pdsResetShelfDates.init
PdsResetDispositionStatus.run
PdsUpdateExpDate.run
PdsUpdateShelfAdvice.run
PriceConvert_Currency.parmPrice
PriceDisc.calcPriceAmount
PriceDisc.resetPrice
PriceDiscLine.hasOnlyLineAmount

REFACTORED METHODS
PriceDiscLine.lineAmountModified
ProdJournalCheckPostRoute.postTransLedger
ProdJournalCreateBOM.createSingleLineProdBOM
ProdUpdCostEstimation.createProdTable
ProdUpdCostEstimation.pmfCreateSubProdTable
ProdUpdReportFinished.updateBOMConsumption
ProdUpdStartUp.updateBOMConsumption
ProjBudgetTransactionManager.getTotalTransactionBudget
ProjControlPosting.queryNext
ProjFormLetter.run
ProjGroupChange.run
ProjInvoiceChooseNormal.doProposal
ProjInvoiceChooseNormal.initQuery
ProjInvoiceJournalCreate.exchRateSet
ProjInvoiceJournalPost.insertProforma
ProjInvoiceJournalPost.matchInvoicePackingSlip
ProjInvoiceJournalPost.postCustVend
ProjInvoiceProposalCreateLinesBase.doDeduction
ProjInvoiceProposalCreateLinesBase.doSalesLine
ProjInvoiceProposalInsertLines.doRevenue
ProjInvoiceSelect.queryBuild
ProjInvoiceSelect.run
ProjPost.newCreateProjTransItemCostAdjustNeg
ProjPost.postCost
ProjPostCostTransCost_Adj.projTransUpdate

REFACTORED METHODS
ProjPostCostTransSale_Adj.projTransUpdate
ProjPostEmplJournal.projTransCreate
ProjPostEmplTransCost_Adj.projTransUpdate
ProjPostEmplTransSale_Adj.projTransUpdate
ProjPostItemTransSale_Adj.projTransUpdate
ProjPostRevenueJournal.projTransCreate
ProjProposalJour.insert
ProjSalesItemReq.clicked
ProjSalesItemReq.run
ProjStatusUpd.main
ProjTable.checkAccount
ProjTable.createSalesTable_ItemReq
ProjTable.initFromCustTable
ProjTable.insert
ProjTable.update
ProjTable.numberSeqFormHandler
ProjTable.validateWrite
ProjTable/FormDataSourceRoot/ProjTable.createFindRanges
ProjTableCreate.initValue()
ProjTableWizard.editProject
ProjTableWizardCtrl.createProject
ProjWorkBreakdownStructureV2.updateControls
PsaQuotationsController.quoteLanguageId
PurchAutoCreate method setPurchTable
PurchAutoCreate_PurchReq.create

REFACTORED METHODS
PurchCreateFromSalesOrder.ChkIncluded_CheckBox.clicked
PurchCreateFromSalesOrder.included
PurchCreateFromSalesOrder.initFields
PurchCreateFromSalesOrder.SalesLine_ds.checkAllowCreate
PurchCreateFromSalesOrder.SalesLine_ds.included
PurchCreateFromSalesOrder.SalesLine_ds.specifyMinMaxQty
PurchCreateFromSalesOrder.SalesLine_ds.specifyPriceComponent
PurchCreateFromSalesOrder.SalesLine_ds.specifyVendAccount
PurchFinalizeServiceTask.checkAccountDate
PurchFormletterParmDataInvoice.createParmLine
PurchInvoiceJournalPost.lateMatchPackingSlip
PurchInvoiceJournalPost.postInventory
PurchInvoiceJournalPost.updateJournalTable
PurchInvoiceJournalPost.updateSourceLine
PurchInvoiceJournalPost.updateSourceLine
PurchLine.checkInvoiceConstraints
PurchLine.createFromTmpFrmVirtual
PurchLine.deleteSoft
PurchLine.deleteSoftClearValues
PurchLine.initFromSalesLine
PurchLine.itemName
PurchLineBackOrder.project
PurchLineType.statusChangeAllowed
PurchLineType.updateApprovedLine
PurchPurchOrderJournalCreate.initJournalHeader

REFACTORED METHODS
PurchRFQLine.createPurchRFQReplyLine
PurchTable.delete
PurchTable.delete
PurchTable.getFinalDiscPriceDateDelegate
PurchTable.initFromVendTableIL
PurchTable.modifiedFieldWithUserInput
PurchYearEndProcess.processPurchOrder
ReqCalc.allowBatch
ReqCalc.checkInsertInventTransRecord
ReqCalc.covCreatePlannedOrder
ReqCalc.pmfCoCovCreatePlannedOrder
ReqCalcScheduleItemTable.createLoopMapFromQuery
ReqCalcScheduleItemTable.insertDataCompleteNetChange
ReqItemJournalUpdate.updateLines
ReqItemJournalUpdate.validate
ReqTransCache_Periodic.insertProcessItemsFromQuery
ReqTransPOCreate.insertFromReqPo
ReqTransPoMarkChangeType.updateType
ReqTransPoMarkFirm.createInventTransfer
ReqTransPoMarkFirm.createInventTransferJournal
ReqTransPoMarkFirm.createProdBOM
ReqTransPoMarkFirm.createProdTable
ReqTransPoMarkFirm.initInventTransferLine
ReqTransPoMarkSumUp.updateSumUp
ReqTransUpdate.initShelfLifeRef

REFACTORED METHODS
ReqTransUpdate.mustUpdateQty
SalesAutoCreate.setSalesTable
SalesCalcTax_Sales.calcTax
SalesCopying.copyFromSourceTable
SalesCopying.init
SalesCopying_CreditNote.updateInvoiceCreditCopy
SalesCreateOrderFromCustomer.create
SalesEditLines/FormDataSourceRoot/CustAdvanceInvoiceTable/Method/init
SalesFormletterParmData.createParmLine
SalesFormletterParmData.initSalesParmUpdateFormletter
SalesFormletterParmData.updateQueryBuild
SalesInvoiceController.preRunModifyContract
SalesInvoiceController.runPrintMgmt
SalesInvoiceDPBase.getMarkUpTaxCode
SalesInvoiceDPBase.initLocalizationData
SalesInvoiceJournalCreateBase.createJournalHeader
SalesInvoiceJournalPostBase.createReportData
SalesInvoiceJournalPostBase.postLine
SalesInvoiceJournalPostBase.updateJournalLine
SalesInvoiceJournalPostBase.updateJournalTable
SalesJournalSelect_Invoice.closeOK
SalesLine.returnUpdateBasedOnDispcode
SalesLineCopyFromSource.updateSalesLine
SalesLineType.formProduction
SalesLineType.initFromCustInvoiceTrans

REFACTORED METHODS
SalesLineType.initFromSalesBasketLine
SalesLineType.initFromSalesLine
SalesLineType.InitFromSalesTable
SalesLineType.initReleasedProductSpecificDefaulting
SalesLineType.initStorageDimensionsFromSalesTable
SalesLineType.pmfValidateBatchId
SalesLineType.setSalesStatusNonInventoried
SalesLineType.validateWrite
SalesLineType_Project.validateWrite
SalesPackingSlipJournalPost.createReportData
SalesPackingSlipJournalPost.PostInventory
SalesPackingSlipJournalPost.updateSourceLine
SalesPackingSlipJournalPostProj.writeProjTrans
SalesParmTable.createPaymentSched
SalesQuotationCopying.copyServer
SalesQuotationEditLinesForm.initializeAndRun
SalesQuotationEditLinesForm.initializeAndRun
SalesQuotationEditLinesForm_Proj_Confirm.queryBuildSalesQuotationTable
SalesQuotationEditLinesForm_Sales_Confir.numRefSalesId
SalesQuotationJumpRef.main
SalesQuotationLineCopyFromSource.updateAfterCopy
SalesQuotationLineType.salesQtyAllowEdit
SalesQuotationLineType_Proj.initFromSalesQuotationLine
SalesQuotationProjLinkWizard.endUpdate
SalesQuotationProjLinkWizard.linkQuotationToProject

REFACTORED METHODS
SalesQuotationProjLinkWizard.next
SalesQuotationTable.clicked
SalesQuotationTable.initFromBusinessRelationTable
SalesquotationTable.initFromCustTable
SalesQuotationTable.initFromSalesQuotationTable
SalesQuotationTable.modifiedField
SalesQuotationTable.modifiedFieldDDC
SalesQuotationTable.validatewrite
SalesquotationTable.writeCreateQuotation
SalesQuotationUpdate.main
SalesTable.clicked
SalesTable.SalesTable_ds.Create
SalesTableForm.enableUpdateJournalButtonsMultipleOrders
SalesTableType.modifiedField
SalesTableType.modifiedField
SalesTableType.validateDelete
SalesTotals.showTax
SalesTotals.showTaxLine
SalesTotals_Sales.calculateFreeValue
SubledgerJournalAccountEntryTmpSummary.getCopy
SubledgerJournalEntryBalance.initBalances
SubledgerJournalizer.validateDebitCreditBalance
SubledgerJournalizer.validateTransferEntriesBalance
SubledgerJourPennyDiffRecognizer.recognizePennyDifference
SubledgerJourSummaryRptCurRoundAdjRcgnzr.recognizeRoundingAdjustment

REFACTORED METHODS
Table/ProjTable.isCustomerTransferNeeded
Table/PurchTable.checkUpdate
Table/TrvExpTrans/Method/setDefaultProjectFromExpenseReport
TaxCalculationAdjustment.adjustBaseForAllLines
TMSMiscellaneousCharge.ValidateChargeCode
TmsProcessXML_Base.readAppPurchLine
TmsProcessXML_Base.writeShipManualAccessorials
TransactionReversal_Asset.reversalBook
TransactionReversal_Ledger.createGeneralJournal
TSTimesheetEntryQuery.initializeQuery
TsTimesheetsPost.postNoNeverLedgerTrx
VendDocumentLineType_Invoice.validateRow
VendOpenTransReverse.initFromCommon
VendorInvoiceLineSourceDocLineItem.hasMainAccDerivationInputChanged
VendPurchOrderJour.printJournal
VendReport_LedgerReconciliation.insertLedgerTransactions
VendTable.createRecord
WhsControlQty.process
WhsDocumentRouting.getRoute
WHSDocumentRouting.translate
WHSLocationDirective.validateBatchMixingOnLocation
WHSLocationDirective.validateMixingRulesAndStockingLimit
WHSPostEngineBase.prodPickQty
WHSProdTable.stopAndUnpick
WHSReverseSalesWork.createWorkToMoveItemsBack

REFACTORED METHODS
WHSRFControlData.populateData
WhsrfControlData.processDataInternal
WHSRFControlData.processLegacyControl
WHSSplitWork
WHSWarehouseRelease.createLoadLines
WhsWaveFormActions.printPickList
WHSWaveTable.createWaveTableFromTemplate
WhsWorkCreateProdPut.createOrUpdateBatch
WhsWorkCreateReceiving.createBatch
WhsWorkExecute.createAndPostTransferJournal
WHSWorkExecute.CreateDimTrackingRecord
WHSWorkExecuteDisplay.getNextFormState
WHSWorkExecuteDisplay.processTrackingDimDetails
WHSWorkExecuteDisplay.processTrackingDimDetails
WhsWorkExecuteDisplayAdjustIn.displayForm
WhsWorkExecuteDisplayCycleCountGrouping.getCycleCountWorkId
WHSWorkExecuteDisplayListWork.addWorkListFieldForWork
WHSWorkExecuteDisplayListWork.buildTableContents
WHSWorkExecuteDisplayListWork.getWorkQuery
WhsWorkExecuteDisplayLPReceiving.buildReceivingLPInfoFromASNItem
WhsWorkExecuteDisplayPOItemReceiving.buildPOReceiving
WHSWorkExecuteDisplayPOReceiving.buildLicensePlateLabels
WHSWorkExecuteDisplayReportAsFinishedBySerial.createPutWork
WHSWorkInventTransReservationCollectionBuilder.canMoveReservationFromWorkLine
WHSWorkUser.changePassword

REFACTORED METHODS
WHSWorkUserAuthenticator.authenticate
WmsArrivalCreateJournal.createWMSJournalTransFromArrivalDetails
WmsJournalFormTrans.promptSplitReturnLine
WmsJournalTransSplit.serverRun
WMSOrder.updateReservOrderedDim
WmsPickingRoute.finishMulti
WrkCtrCapResHandler.new

Metadata changes

These metadata changes have been made in this update.

OPERATION
/Data Entities/PdsItemBatchAttributeEntity.IsPublic
/Forms/WHSMobileAppField/FormDesign/AppBar/CreateDefaultButtonGroup/CreateDefaultButton.NeededPermission
/Forms/WMSPickingRegistration/Design/Tab(Tab)/Details(TabPage)/HeaderDetails(Tab)/PickingLinesPage(TabPage)/PickLinesGrid(Grid)/InventoryDimensionsGrid(Group).DataGroup
/Table/FreeTextInvoiceLocalizationTmp.Visible
/Tables/MarkupAutoTable/Indexes/MarkupIdx.MarkupIdx
Data types/Extended data types/ItemVolume.NoOfDecimalsIsExtensible
DataEntityView/EcoResProductCategoryAssignmentEntity is OData enabled
DataEntityView/EcoResProductEntity is OData enabled
DataEntityView/EcoResReleasedProductEntity is OData enabled
Enum/InvoiceReferenceNumberFormulaType_FI.Country region code
Enum/InvoiceReferenceNumberFormulaType_FI/EnumValue
RouteOprTime.NoIfDecimalsExtensible
Table/HRPDefaultSigningLimitRuleCompensationTmp.String size
Table/PSAProjInvoiceTmp/Properties.Title1, Title2
Table/VendUnrealizedRev/Field/ReversalDate.Allow edit

Additional extensibility enhancements

In addition to the refactored methods, the following extensibility enhancements have been made.

- Dimension based discount
- Redesign InventPosting searching algorithms

Extensibility changes in Dynamics 365 for Finance and Operations version 8.1

2/18/2021 • 4 minutes to read • Edit Online

This is a list of extensibility features that were implemented in Dynamics 365 for Finance and Operations version 8.1. For more information about the schedule of changes that support extensibility, see Application extensibility plans.

Refactored methods to support extensibility

These methods have been refactored to support extensibility through chain of command, delegates, or by providing access to members.

METHOD
AssetPost.createTrueUpDepreciation
AssetPost.createTrueUpDepreciation
AssetPostDisposal.post
AssetPostDisposal.postVoucherTransactions
AssetPostDisposal.postVoucherTransactions
AssetTable.buildComposedOf
AssetTable.createStruct
AxInventDim_SalesLine.setInventSiteId
BankChequePrint.printDocument
BankCodaProcessing.custSettlement
BankDeposit.InitFromLedgerJournalTrans
BankExchAdj_RU.calcAndPostCurrency
BankExchAdj_RU.calcAndPostCurrency
BankExchAdj_RU.calcBalance
BankPrintTestCheque.printCheque
BankPrintTestCheque.printCheque
BankPrintTestCheque.printCheque

METHOD
BankPrintTestCheque.printCheque
BlindCloseView.MPOS_ExtensibleViews
BudgetAnalysisInquiryHelper_PSN.insertTreeNodes
BudgetAnalysisInquiryHelper_PSN.insertTreeNodes
BudgetAnalysisInquiryProcessor_PSN.getBudgetAnalysisLedgerDimensions
CAMStatisticalEntryTransferJournalEntryDataMapper.newFromParameters
CaseUpdateStatus_Close.changeStatus
CashManagementView.NewExtension
ChequeController.init
ChequeDP.insertChequeTmp
Class InventTransferEstimation.updateEstimatedPre
class Markup.insertReturnMarkupTrans
class MarkupTransInsert.insertForCodes
Class PurchLineType.updateSalesLine
class PurchPackingSlipJournalPost.postInventory
Class SalesQuantity_PackingSlip.calcQtyInvent
Class SalesQuantity_PackingSlip.calcQtySales
Class/AssetPost.post
Class/FormLetterJournalPost.post
Class/PurchReqTableListPageInteraction.applyFilter
Class\PurchFormLetter_PackingSlip.checkFormLetterId
Class\PurchFormletterProvider.checkLines
Class\TaxCalculationAdjustment.calcManualInserted
CompanyInfoHelper.onValidateField
CostSheetDesigner.DataSource:CostSheetCalculationFactor.validateWrite

METHOD
CustAccountStatementExtController.initAgingBalances
CustAccountStatementExtController.preprocessParty
CustAccountStatementExtController.processParty
CustAccountStatementExtController.processTrans
CustAccountStatementExtController.runPrintMgmt
CustCollectionLetterCreate.run
CustCollectionLetterPost.RUN
CustInterestAdjust.createCustInvoiceTable
CustInterestCreate.createJournal
CustInterestCreate.logDateRecordError
CustInterestCreate.newAccount
CustInterestCreate.resetCustInterest
CustInterestCreate.runOnce
CustInvoiceCalcTax_Invoice.updateTaxWriteCode
CustInvoiceLine.Insert
CustOutPaym.generatePaymentLines
CustQuotationJour.printJournal
CustTable.CanSubmitToWorkflow
CustTrans.documentDateModified
CustTrans.documentDateModified
CustTransSettlement.insertSettlementLines
CustVendCheque.createBankChequePaymentTrans
CustVendCheque.initTmpChequePrintout
CustVendCheque.output
CustVendChequeSlipTextCalculator.seebelow

METHOD
CustVendCreatePaymJournal_Vend.shouldAddCustVendTransOpen
CustVendPaymProposalTransferToJournal.transferProposal
CustVendPaymSched.construct
CustVendPaymSched.initCalcPaymSched
CustVendReversePosting.restoreCustVendTransOpen
CustVendVoucher.post
DataEntityView/SalesOrderLineEntity/Method/validateWrite
DataEntityView/SalesQuotationLineEntity/Method/validateWrite
EFDocEmailProcessor_BR.saveReceivedXmlData
EFDocMsgFormat_XmlBase_BR.createElementWithValue
Extract into method: SalesParmTable.GetPaymentSched
Form InventJournalAsset\Methods\init
Form InventJournalCount\Methods\init
Form InventJournalMovement.init
Form TrvExpenses.confirmCategoryChange
Form\SalesEditLines.closeOk
FormletterJournalPost.newPostPurch
FreeTextInvoiceDP.Variable declaration
HcmActionTypeSetup.lookupWorkflowTable
HcmPosition_HcmPositionDefaultDimension_formDatasource.selectionChanged
HcmPositionTransition:: createHcmPositionWorkerAssignment
HcmPositionWorkerAssignmentDialog:: createWorkerActionOk
HRMCompFixedPlanTable::enforcePayRateTolerance
InterCompanyPostPurch.construct
InterCompanyPostPurch.formLetterUpdate

METHOD
InterCompanyPostSales.formLetterUpdate
InterCompanySyncPurchTableType.setSalesTableData
IntrastatCheck.Run
IntrastatTransfer.calcAmountsAndMarkups
IntrastatTransfer.calcValuesSign
IntrastatTransfer.distributeIntrastatAddValueLV
IntrastatTransfer.run
IntrastatTransferIT.calcCounty
IntrastatTransferIT.updateTransactionCurrencyAmount
InventAdj_Transact.run
InventCostPost.postInventCostTransVariance
InventMov_Purch.updateAutoLossProfit
InventMov_Purch.updateBuffer
InventMovement.checkUpdatePhysical
InventoryLookupMatrixView.ExtensibleViews
InventTable.pdsValidateBestBeforeDays
InventTransWMS_Register.updateInventFromMovementServer
InventUpd_ChildReference.updateLessIssue
InventUpd_ChildReference.updateLessReceipt
InventUpd_ChildReference.updateMoreIssue
InventUpd_ChildReference.updateMoreReceipt
InventUpd_Financial.newSalesInvoice
InventUpd_Physical.updatePhysicalIssue
InventUpd_Physical.updateTransPhysicalReturnedReceipt
InventUpd_Physical::newSalesPackingSlip

METHOD
InventUpd_Reservation.updateNow
InventUpd_Reservation.updateReserveMore
InventUpdate.updateInventTransPosting
InventUpdate.writeInventTrans
InventUpdate.writeInventTrans
InventUpdateMarking.addMarking
InventUpdateMarking.removeMarking
InventUpdateReserveMore.createQueryRuns
JmgCalcApprovePickDialog.closeOk
JmgJobBundle.postTime
JmgJobBundleProdFeedbackForm.getTmpJobBundleProdFeedback
JournalStaticDataModel.initializeJournalTableFields
Ledger.populateTmpTable
Ledger::populateTmpTable
LedgerAccrualTrans_Calendar.allocate
LedgerAccrualTrans_Calendar.allocate
LedgerAccrualTrans_Fiscal.allocate
LedgerAccrualTrans_Fiscal.allocate
LedgerAutomaticTransactionAccountEntity
LedgerCreatePeriodBalances.createPeriodBalancesMainAccount
LedgerExchAdj.postAdjustment
LedgerExchAdj.run
LedgerFiscalJournalDP_IT.getDisplaySequenceNumber
LedgerJournalCheckPost.checkJournal
LedgerJournalCheckPost.postJournal

METHOD
LedgerJournalCheckPost.runInternal
LedgerJournalEngine.accountModified
LedgerJournalEngine.calculateTaxForCompleteJournal
LedgerJournalEngine.findSettledAmount
LedgerJournalEngine.formSettlement
LedgerJournalEngine.newVoucher
LedgerJournalPeriodicCopy.journalTransCopy
LedgerJournalTrans.checkVATTransaction
LedgerJournalTrans.checkVatTransaction
LedgerJournalTrans.validateWrite_Server
LedgerJournalTrans_Project.checkCategoryId
LedgerJournalTransUpdateAsset
LedgerJournalTransUpdateLedger.updateNow
LedgerJournalTransVoucherTemplates.createVoucherTemplate
LedgerJournalTransVoucherTemplates.createVoucherTemplate
LedgerJournalTransVoucherTemplates.saveVoucherTemplate
LedgerJournalTransVoucherTemplates.saveVoucherTemplate
LedgerJournalTransVoucherTemplates.SaveVoucherTemplate, CreateVoucherTemplate
LedgerPostingGeneralJournalController.addLine
LedgerPostingGeneralJournalController.getLineValues
LedgerTransferOpening.getMainAccountStorageSegment
LedgerTransferOpening.processNewClosingRecordsForOpening
LedgerTransferOpening.processQueryForPublicSector
LedgerTransModule.tax
LedgerVoucherObject.allocateTransaction

METHOD
LedgerVoucherObject.updateLedgerPostingJournal
LogisticsEntityLocationFormHandler.manageControls
LogisticsPostalAddressFormEventHandler.updatePrimaryControl
MainAccount.canSubmitToWorkflow
Move variable declaration to first assignment when possible
NewElement.NewMethod
OmOperatingUnit.getDimensionViewId
Originaldocuments.findFromCustTrans
Originaldocuments.findFromGeneralJournal
PaymSchedCalc.createCustVendTransaction
PaymSchedCalc.initFromPurchTotals
PaymSchedCalc.initFromSalesTotals
POSApidts.NewTrigger
PosAPidts.NewTrigger
PriceDiscAdmCheckPost.checkForOverlapsAndGaps
PriceDiscAdmCheckPost.runFromContract
PriceDiscTable.buildSearchFilter
PrintableReceipt
ProjAdjustmentUpdate.newPostAdjustment
ProjAdjustmentUpdate.projTrans
ProjAdjustmentUpdate.transEmplNew
ProjAdjustmentUpdate_Post.Post
ProjAdjustmentUpdate_Post.update
ProjBudgetManager.createBudgetRevenueForecast, createBudgetSalesForecast
ProjBudgetManager.insertProjBudgetLine

METHOD
ProjBudgetManager.insertProjBudgetLine
ProjBudgetManager.updateProjBudgetLinesWithAmt
ProjCategory.categoryType2CategoryEmplOption
ProjCategory.categoryType2CategoryEmplOption
ProjCategory.categoryType2TransType
Projcategory.transType2CategoryType
Projcontrol.categoryType2CostType
Projcontrol.costType2CategoryType
ProjControlPosting.insertControlTrans, processSalesValue
ProjectSourceDocumentLineItemHelper.projOrigin and projTransType
ProjForecastListPageInteraction.runForcastFormBasedOnForecastUnion
ProjForecastReduce.newProjPost
ProjFormLetter method main/mainOnServer
ProjInvoiceCancel.cancelProposal
ProjInvoiceChoose.run
ProjInvoiceJournalCreate.createJournalHeader
ProjInvoiceLines.run
ProjInvoiceProposalCreateLines.loadLastValue
ProjInvoiceProposalCreateLines.runItemQuery
ProjInvoiceProposalInsertLines.run
ProjInvoiceProposalInsertLines.setProjProposalJour
ProjInvoiceProposalInsertLines.setProjProposalJour
ProjInvoiceProposalInsertLines.setProjProposalTotals
ProjInvoiceTableCreate.initializeValues
Projparameters.defaultProjCategory

METHOD
ProjPeriodPostingLedgerCost.updateTrans
ProjPost.newCheckTrans
Projpost.newCreateProjTransAndLedger
Projpost.newCreateProjTransAndLedgerAdj
Projpost.newTransAdjNegativeJournal
Projpost.postcost
ProjSalesItemReq.modified
ProjSplitBill.transType
ProjSplitBill.transType
ProjTable.generateNextSubProjectId
ProjTableCreate.canClose
ProjTableLookup.isJournal
ProjTableWizardCtrl.createProject
PsaProjAndContractInvoiceController.getReportTitle
PsaProjAndContractInvoiceController.getReportTitle
PsaProjAndContractInvoiceController.getReportTitle
PSAProjRetainerInvoicing.run
PsaQuotationsController.getReportTitle
PurchaseOrderResponseConsume.consumeChangesToPurchLineAndUpdateResponeLineConsumptionState
PurchaseOrderResponseConsume.consumeChangesToPurchLineAndUpdateResponeLineConsumptionState
PurchaseOrderResponseConsume.consumeChangesToPurchLineAndUpdateResponeLineConsumptionState
PurchAutoCreate_Sales.createPurchTable
PurchAutoCreate_Sales.setPurchTable
PurchCalcTax_Invoice.updateTaxWriteCode
PurchCopying.copyLine

METHOD
PurchCreateFromSalesOrder.autoCreatePurchOrder
PurchFormletter_Invoice.newinvoice
PurchFormletter_Packingslip.runProjectPostings
PurchFormletterParmData.createParmLine
PurchFormletterParmData.newData
PurchFormLetterParmData.reSelectLines
PurchFormletterParmDataInvoice.chooseLinesNext
PurchFormletterParmDataInvoice.cleanupChooseLines
PurchFormletterParmDataInvoice.copyMarkupFromPurchOrder
PurchFormletterParmDataInvoice.processAdditional
PurchFormletterProvider.checkLines
PurchInvoiceJournalCreate.checkInvoice
PurchInvoiceJournalPost.postInventory
PurchLine.delete
PurchLine.setProjSalesPrice
PurchLine.update
PurchLine.validateWrite_Server
PurchLineCopy.canClose
PurchLineType.syncSalesLine
PurchLineType.updateInventory
PurchLineType.updatePurchStatus
PurchLineType.validateDelete
PurchPackingSlipJournalPost.postInventory
PurchPackingSlipJournalPost.updateSalesLine
PurchPackingSlipJournalPost.updateSalesTable

METHOD
PurchPackingSlipJournalPost.updateSourceLine
PurchQuantity_PackingSlip.calcQtyInvent
PurchQuantity_PackingSlip.calcQtyPurch
PurchReqAddItems.init
PurchReqTable.init
PurchReqWFExpendiParticipantProvider.resolve
PurchSelectLinesManager.mark
PurchTableForm.purchLine_WritePreSuper
PurchTableInteractionHelper.getDeliveryScheduleEnabled
PurchTableInteractionHelper.getHasMultipleDeliveries
PurchUpdateRemain.updateRemainPhysical
ReqCalc.insertItemInventSum
ReqTransPlanIdFilter.setPlanIdOnQueryRange
ReqTransPoMarkFirm.createPurchLine
ReqTransPoMarkFirm.createPurchTable
ReqTransPoMarkFirm.createPurchTable
ReqTransPoMarkFirm.dialog
RetailAssortmentLookupTask.CreateChannelGroups
RetailKitAssemblyOrder.createOrUpdateBOMJournal
RetailPackage.close
RetailProductPropertyManager.saveProductDimensions
RetailStatementPostChecker::checkStatement
ReturnTable.isDispositionCodeValid
RouteCopyToRoute
SalesCalcAvailableDlvDates.newCommonSalesDlvDateType

METHOD
SalesCalcAvailableDlvDates.newCommonSalesDlvDateTypeByEntity
SalesCopying.copyLines
SalesEditLines.FormDesign/FormMenuFunctionButtonControl/ButtonPaymentSched/Method/clicked
SalesFormLetter.onCreatePaymSched
SalesFormLetterParmDataPackingSlip.selectChooseLines
SalesFormletterProvider.checkLines
SalesFormletterProvider.checkSalesLineChanged
SalesInvoiceController.initFormLetterReport
SalesInvoiceJournalPost.postInventory
SalesInvoiceJournalPostProj.updateInventory
SalesLine.createAlternativeItem
SalesLine.delete
SalesLine.returnLineUpdate
SalesLineCopy.canClose
SalesLineType.initFromCustInvoiceTrans
SalesOrderEntryStatistics.createOrderEntry
SalesOrderEntryStatistics.deleteOrderEntry
SalesPackingSlipDP.itemId
SalesPackingSlipJournalPost.postInventory
SalesParmLine.setQty
SalesQuantity_Invoice.calcQtyInvent
SalesQuantity_Invoice.calcQtySales
SalesQuotationEditLinesForm_Sales_Confir method createSalesLines
SalesQuotationEditLinesForm_Sales_Confir method createSalesTable
SalesQuotationEditLinesForm_Sales_Confir.createSalesLine

METHOD
SalesQuotationEditLinesForm_Sales_Confir.createSalesLines
SalesQuotationLine.createLine
SalesQuotationLine.createQuotationLineFromTemplate
SalesQuotationLine.createQuotationLineFromTemplate
SalesQuotationLine.createQuotationLineFromTemplate
SalesQuotationLine.delete
SalesQuotationLine.setCostSalesPrice
SalesQuotationLine.updateSalesQuotationTable
SalesQuotationListPageInteraction.initIsSalesQuotation
SalesQuotationListPageInteraction.setButtonFollowup
SalesQuotationListPageInteraction.setButtonQuotation
SalesQuotationTableForm_DlvSchedule.updateSalesQuotationLineTable
SalesQuotationTableType.validateDelete
SalesTable.update
SalesTableForm_DeliverySchedule.updateSalesLineTable
SalesTableForm_ProjectSalesItem.resetSalesLine
SalesTableInteractionHelper.isOpenOrderNotReturnNotProjectRelatedSalesLine
SalesTableInteractionHelper.notPartiallyPickedPackedOrInvoiced
SalesTableInteractionHelper.notReservedOrderedNorPhysical
SalesTableType.checkUpdate
SalesTableType.checkUpdate
SalesTableType.checkUpdate
SalesTaxDeclarationInformationReportService.processReport
SettlementPair_Vend.fetchPayment
smmActivityParentLinkTable.insert

METHOD
smmBusRelTable.convert2Customer
smmBusRelTable.convert2Customer
SmmBusRelTable.convert2Vendor
SmmBusRelTable.convert2Vendor
smmBusRelTable.createConvertedBusRel
smmLeadTable.createBusRelRecord
SmmProjectCreate.createProjectGroup
SpecTransInsertSetManager.insertDatabase
SubledgerJournalizerProjectExtension.createProjectActualCostDetail, createProjectActualHeader
SubledgerJournalizerProjectExtension.getProjectActualMap
SuppItemSales.initSalesLine
Table PurchLine.insert
Table PurchLine.insert
Table PurchLine.update
Table PurchTable.update
Table PurchTable.update
Table SalesLine.insert
table SalesLine.projIdModified
table SalesLine.salesQtyModifiedInteraction
Table SalesLine.update
Table SalesQuotationLine.insert
Table SalesQuotationLine.update
Table SalesQuotationTable.update
Table SalesQuotationTable.update
Table SalesTable.update

METHOD
Table SalesTable.update
Table\VendTable.name
Table\VendTable.updateOnHold
Tax.createOrphanLinkInsteadPost_RU
Tax.distributeTotalTax
Tax.distributeTotalTax
Tax.InsertIntersection
Tax.insertIntersection
Tax.post
Tax.Post
Tax.postCharge
Tax.postTaxProjInvoice_IN
Tax.postTaxPurchInvoice_IN
Tax.postTaxSalesInvoice_IN
Tax.saveAndPost
Tax.taxTotals
Tax.taxTotals
Tax.taxTotals
Tax.taxTotalsPosted
Tax.taxTotalsPosted
Tax.taxTotalsPosted
TaxBookSection.checkTaxBookSection
TaxCalculationAdjustment.calcManualInserted
TaxCalculationJournal.saveTaxTransfers
TaxFreeInvoice_Invoice.updateAndPost

METHOD
TaxInvoiceSpec.parmTaxSpec
TaxListDP.insertTaxListTaxTmpData
TaxListDP.updateTaxListTaxTmpData
TaxMainAccDimensionListProvider.populateMainAccountDimensionList
TaxPost.postToLedger
TaxReportController_US.init
TaxReportInclAdjustmentDP.insertTaxReportInclAdjustmentTmp
TaxReportingDP.insertTaxReportingTmp
TaxReverse.adjustTaxTransDueToExchangeRateGainLoss
TaxReverse.postAccountingCurrency
TaxReverse.postAccountingCurrency
TaxReverse.postAccountingCurrency
TaxReverse.postCharge
TaxReverse.postGainLossInReportingCurrency
TaxSales.calc
TaxSales.calc
TaxSales.calcMarkup
TaxSales.configureTaxForSalesLine
TaxVoucherService.taxAmountForLedgerType
TmpCustVendTrans.CustTransBalanceCurrency
TmpProjAdjustment.adjustmentType2JournalType
TmpProjAdjustment.adjustmentType2TransType
TmpProjAdjustment.updateFundingLimits
TmpProjAdjustmentCreate.salesPrice
TrvExpenseLinesVisibilityController.isVisibilityResetRequired

METHOD
TrvExpenses.updateFormVisibilityOnCategoryChange
TrvExpTrans.calcTaxAmount
TrvTaxExpense.calculateTax
TSTimesheetEntry.init, initFields, setHeaderObjects, validateWrit, lookup
UnitOfMeasureConverter.Convert
VendInvoicePaymentAuthorizationTask.postSavedInvoice
VendPackingSlipTrans.unpostedInvoicePurchQtyServer and VendPackingSlipTrans.unpostedInvoiceInventQtyServer
VendTable.checkVATNumUsed
VendTax1099Update.calcMiscChargeAmountTax
VendTrans.documentDateModified
VendTrans.documentDateModified
VendVoucher.createTransOpen
VendVoucher.createTransOpen
WHSContainerTable.closeContainer
WHSDocumentRouting.translate
WHSLoadLine.orderHeader
WHSLoadTable.validateInventTransTypeMatches
WHSLoadTableAssignOriginInfo.classDeclaration
WmsArrivalCreateJournal.createWMSJournalTransFromTmp
WMSOrderTrans.split
WmsOrderTransType_Output.updateReservations
WorkflowHierarchyProviderHelperEventHandler::getPersonnelNumberIdBySysDictTypeDelegate
WorkTimeTable.removeDisplayCache
WrkCtrResourceAbilityMapController.insertData
Enable increase of decimal precision through extensiblity for quantities

Enumerations made extensible

These enumerations have been made extensible in this update.

ENUMERATION
AssetPostValue
AssetPropertyType
AssetStatus
CaseStatus
CommissionSalesRepCode
DirSubNameSequenceType
FreightSlipType
HRPLimitDocumentType
IntrastatPaymentMethod_IT
InventBlockingType
JmgPaySpecTypeEnumPick
KMQuestionAnswerInputType
LedgerJournalType
LogisticsAddrZipCodeImportCountryRegion
LogType
MarkupModuleType
MarkupModuleType
MCRBrokerValueType
PaymentType
PDSCalcElementTypeBase
PdsStatus
PdsVendorCheckItem
ProdStatus
ProjActiveAll

ENUMERATION
ProjAdjustmentType
ProjAllTrxType
ProjBudgetAction
ProjCostType
ProjForecastInvoiceFrequency
ProjLinePropertySearch
ProjSalesPriceMarkup
ProjSortValue
ProjTableEditSubProj
ReqCovType
ResCharacteristicSetEnum
SettlementType
smmShowTimeAs
SourceDocumentRelationType
TaxRegistrationTypesList
TaxReportLayout
TradeWorkflowState
WMSExpeditionStatus

Extensible SQL Operations

These SQL operations have been made extensible in this update.

OPERATION
CustInterestPost.postVoucherPerTransaction
InventMov_ProdLine_JournalBom.insertChildBuffer
InventUpdate.initInventTransToReceiveList
ProjAdjustmentUpdate_PostAsync.postAsync

OPERATION
ProjBudgetManager.createBudgetCostForecast
ProjBudgetManager.createBudgetEmplForecast
ProjBudgetManager.createBudgetRevenueForecast
ProjBudgetManager.createBudgetSalesForecast

Metadata changes

These metadata changes have been made in this update.

OPERATION
/Data entities/ReqPlannedOrderEntity.Is Public
/DataTypes/Extended Data Types/AmountQty.NoOfDecimalsIsExtensible
/DataTypes/Extended Data Types/AssetDepreciationAmountUnitReportingCurrency.NoOfDecimalsIsExtensible
/DataTypes/Extended Data Types/BOMProductQuantity.NoOfDecimalsIsExtensible
/DataTypes/Extended Data Types/CostPriceNonMonetary.NoOfDecimalsIsExtensible
/DataTypes/Extended Data Types/CostQuantity.NoOfDecimalsIsExtensible
/DataTypes/Extended Data Types/MCRRoyaltyValue.NoOfDecimalsIsExtensible
/DataTypes/Extended Data Types/PdsRebateValue.NoOfDecimalsIsExtensible
/DataTypes/Extended Data Types/PriceDiscAmount.NoOfDecimalsIsExtensible
/DataTypes/Extended Data Types/PriceQty.NoOfDecimalsIsExtensible
/DataTypes/Extended Data Types/PriceUnit.NoOfDecimalsIsExtensible
/DataTypes/Extended Data Types/PriceUnit.Scale
/DataTypes/Extended Data Types/ProductQuantity.NoOfDecimalsIsExtensible
/DataTypes/Extended Data Types/ProductQuantityHourValue.NoOfDecimalsIsExtensible
/DataTypes/Extended Data Types/ProjName.Extends
/DataTypes/Extended Data Types/TAMRebateValue.NoOfDecimalsIsExtensible
/DataTypes/Extended Data Types/UnitAmountCur.NoOfDecimalsIsExtensible
/DataTypes/Extended Data Types/UnitAmountMST.NoOfDecimalsIsExtensible

OPERATION
/DataTypes/Extended Data Types/WeightBase.NoOfDecimalsIsExtensible
/Tables/LedgerJournalTrans_Project/Relations/LedgerJournalTrans.createNavigationPropertyMethod
/Tables/PriceDiscGroup.Cache Lookup
/Tables/VendInvoiceJour/Fields/DefaultDimension.Visible
/Tables/VendInvoiceTrans/Fields/DefaultDimension.Visible
/Tables/WMSAisle.Cache Lookup
/Tables/WorkCalendarTable.Create Rec Id Index
/Tables/WorkTimeLine.Cache Lookup
/Tables/WorkTimeTable.Cache Lookup

Additional extensibility enhancements

In addition to the refactored methods, the following extensibility enhancements have been made.

- Bug request: "CustCollectionLetterTrans -> CollectionLetterNum" Relation properties
- Enable increase of decimal precision through extensiblity for prices
- Enable increase of decimal precision through extensiblity for weights
- Map Extension: LogisticsEntityLocationMap
- OMOperatingUnit should provide user friendly and defined value for DimAttributeOMDepartment.Value
- Redesign how InventPosting finds LedgerDimension
- Refactor WhsWorkExecuteDisplayAdjustIn to ProcessGuide framework
- Refactor WHSWorkExecuteDisplayChangeWarehouse to ProcessGuide framework
- Refactor WhsWorkExecuteDisplayInquiryItem to ProcessGuide framework
- Refactor WhsWorkExecuteDisplayInquiryLocation to ProcessGuide framework
- Refactor WhsWorkExecuteDisplayInquiryLP to ProcessGuide framework
- Refactor whsWorkExecuteDisplayReprintLabel to ProcessGuide framework
- Retail channel: Support BankDropOperationRequest

Extensibility changes in Dynamics 365 for Finance and Operations update 8.0.4

2/18/2021 • 2 minutes to read • Edit Online

This is a list of extensibility features that were implemented in Dynamics 365 for Finance and Operations update 8.0.4. For more information about the schedule of changes that support extensibility, see Application extensibility plans.

Refactored methods to support extensibility

These methods have been refactored to support extensibility through chain of command, delegates, or by providing access to members.

METHOD
AgreementHeader.getModuleType
AssetSplit.construct
BankDepositSlipController.main
BankPositivePayExport.sendFileToUser
CaseDetailForm.getRecordsFromDataSource
CostSheetDesigner.DataSource:CostSheetCalculationFactor.validateWrite
CostSheetNodeCalculation.validate
CostSheetNodeCalculationRate.calcLowestLevel
CostSheetNodeCalculationSurcharge.equal
CustAccountStatementExtController.processParty
CustAccountStatementExtDP.insertNewRecords
CustAccountStatementExtDP.setSysDocuBrandDetails
CustBillOfExchangePost.postNextStep
CustBillOfExchangePostProtestHonored.postNextStep
CustCollectionJourController.runPrintMgmt
CustCollectionLetterCreate.createJournal
CustCollectionLetterCreate.run
CustCollectionLetterNote.CustCollectionLetterJour.active

METHOD
CustCollectionLetterPost.processRow
CustCollectionLetterPost.validateCollectionLetter
CustFreeInvoiceCorrection.createInvoiceLines
CustInterestPost.main
CustInterestPost.validateInterestTrans
CustInvoiceJour.printJournal
CustInvoiceTable.calcDue
CustOpenBalanceCurrency.Data Sources – VendTrans – init
CustOpenTrans.editMarkTrans
CustPackingSlipJourFormHelper.areCancelCorrectButtonsEnabled
CustPaymEntry.editIsMarkedForSettlement
CustPostInvoice.main
CustPostInvoice.run
CustPostInvoice.validate
CustPostInvoiceJob.custPostInvoiceUpdate
CustPostInvoiceJob.initializeCustPostProcess
CustPostInvoiceJob.main
CustPostInvoiceJob.processCustPostInvoiceUpdate
CustProvisionalBalanceDP.calculateAmounts
CustProvisionalBalanceDP.insertCustProvisionalBalanceTmp
CustProvisionalBalanceDP.populateCustProvisionalBalanceTmpProcessing
CustProvisionalBalanceDP.processReport
CustProvisionalBalanceDP.translateMainAccountNamesOnCustProvisionalBalanceTmpProcessing
CustQuotationJournal.launchReport
CustSettlementPriorityProcessing .createTempData
CustSettlementPriorityProcessing .setPaymentAmount

METHOD
CustSettlementPriorityProcessing .updatePartialTrans
CustSettlementPriorityProcessing.classDeclaration
CustSettlementPriorityProcessing.constructCustOpenTrans
CustSettlementPriorityProcessing.constructCustPaymEntry
CustSettlementPriorityProcessing.constructOffsetVoucherCust
CustSettlementPriorityProcessing.getBillingPriorities
CustSettlementPriorityProcessing.getSettlementQuery
CustSettlementPriorityProcessing.initCustTransOpen
CustSettlementPriorityProcessing.insertAllLinesAccrossInvoices
CustSettlementPriorityProcessing.markTransactions
CustSettlementPriorityProcessing.markTransByCreditNoteOnBillingClasses
CustSettlementPriorityProcessing.validateMarkedTransactionOpenTrans
CustStatisticsUS - method calcStatistics
CustTable.validateCNPJCPF_BR
CustTrans.checkReversal
CustVendCheque.processChequeNum
CustVendCreatePaymJournal_Vend.searchTransactions
CustVendExchAdjPostingEngine.addExchangeAdjustment
CustVendFindSettlements.getTmpTrans
CustVendPaymProposalTransferToJournal.transferProposalLineToJournal
CustVendSettle.createSummaryAccountReliefTransactions
CustVendSettle.mustOffsetOriginalSummaryDistributions
CustVendSettle.postingProfileSettle_CreateDistributions
CustVendSettle.settleNow
CustVendTransDistributionController.getDistributionFactorsForPostingTypes
CustVendTransExchAdjDistController.getDistributionFactorsForPostingTypes

METHOD
CustVendTransSettle.post
CustVendVoucher.initLedgerPosting
CustWriteOff.createInterestWriteOffJournalForInterestTrans
DataFileImportExportUtils.readStreamWriterAndWriteToStreamWriter
EcoResProductCreate.initDefaultControlValues
EcoResProductReleaseManager.releaseToLegalEntity
ExchangeRateImportOperation.saveRates
FormletterJournalCreate.newPurchJournalCreate
FulfillmentLineView.NewExtension
InterCompanyPost.formLetterCollect
InventCostIndirectFinancial.remainingQty
InventCostItemDim.load
InventCostJournalIndirectCost > addTrans
InventCostTrans.setRefTypeFromInventTransType
InventDimCtrl_Rep_Sales.initDimParmFormletter
InventDimCtrl_Rep_Sales.mustShowField
InventDimCtrl_Rep_Sales.reportStrItemId
InventDistinctProductOrderDefaultingController.construct
InventItemLocationCountingStatus.updateStopCountingJournal
InventItemPriceActivationJob.activateCostSheetCalculationFactor
InventJournalCheckPost_Movement.postTransLedgerMovement
InventJournalFormTrans_Movement.initReleasedProductSpecificDefaulting
InventoryMainAccDimensionListProvider.ledgerPostingType2InventAccountType
InventQualityOrderTable.setTestResult
InventSerial.init
InventSumPhysicalSpec.setValueQty

METHOD
InventTable.insertInventItemOrderSetup
InventTrans.updateSumUp
InventTransferLine.updates
InventTransferUpd.beginLedger
InventTransIdSum.update
InventUpd_Estimated.updateFieldsChange
InventUpd_Financial.updateFinancialIssue
InventUpd_Financial.updateNow
InventUpd_Physical.updatePhysicalReturnedReceipt
InventUpd_Registered.pickReleatedIssueTransMore
InventUpd_Reservation.updateReserveMore
InventUpdateMarking.updateReservations
JmgAbsenceCalendar
JmgMESSwitchCode.init
LedgerExchAdj.postAdjustment
LedgerExchAdj.run
LedgerJournalEngine.initTaxGroup
LedgerJournalEngine_CustBillSettle.initCustOffsetAccount
LedgerJournalTransCustPaym.LedgerJournalTrans.CustVendBankAccountId.jumpRef
LedgerJournalTransUpdateCust
LedgerJournalTransUpdateLedger.updateNow
LogisticsPostalAddress.whsAddressFormatValidation
LogisticsPostalAddressFormEventHandler.updatePrimaryControl
Markup.calc
MCRCustPaym.ValidateWrite
McrCustPaymTotals_Sales.allPaymentsSubmitted

METHOD
OffsetVoucher.updateNow
PmfCoByProdCalcTrans.updateRealCalcIndirect
PmfCoByProdCalcTrans.updateRealCalcIndirect
POSAPIdts.New Trigger
PriceDiscAdmCheckPost.checkForOverlapsAndGaps
PriceDiscAdmTrans.canEditPriceDiscValueField
ProcCategoryHierarchyManagement.init
ProdCalcTrans > method updateRealCalcIndirect
ProdIndirectTrans > method type2ItemCalcType
ProdJournalCheckPostProd.postTransLedger
ProdTableForm.handleProdTableCreatePreSuper
ProdUpdReportFinished.updateBOMConsumption
ProdUpdReportFinished.updateBOMConsumption
ProjAdjustmentUpdate.deleteJournal
ProjFundingLimitTrackingManager.updateUsingSourceDocument
ProjInvoiceProposalInsertLines.run
ProjInvoiceTableCreate.canClose
ProjInvoiceTableCreate.initializeValues
ProjPlanVersionsManager.createTemplateHierarchy
ProjPostCostJournal.projTransCreate
ProjSalesItemReq.SalesLine.linkActive
ProjTableType_TimeMaterial.validateWrite
PurchAgreement.applyQueryRanges
PurchApproveJournalPost.postPurgeLedgerAccount
PurchaseOrderResponseService.shouldPurchaseOrderBeAutoConfirmed
PurchAutoCreate_PurchReq.initializeAndCreatePurchLine

METHOD
PurchAutoCreate_PurchReq.initializeAndCreatePurchLine
PurchAutoCreate_Sales.createLine
PurchCalcTax.construct
PurchCancel.run
PurchCreateFromOrder.insertMinMaxQty
PurchCreateFromSalesOrder.insertIntoTmpPurchLinePrice
PurchCreateFromSalesOrder.refreshCallerDataSource
PurchCreateFromSalesOrder.Salesline.specifyMinMaxQty
PurchCreateFromSalesOrder.SalesLine.specifyVendAccount
PurchCreateFromSalesOrder.validateSalesLine
PurchEditLines.canClose
PurchEditLinesForm.construct
PurchFormLetter.construct
PurchFormLetterContract.newFromPackedVersion
PurchFormletterParmDataInvoice.selectFromJournalLines
PurchFormLetterProvider.checkPurchLineChanged
PurchInvoiceJournalPost.updateSourceLine
PurchLine.checkInvoiceConstraints
PurchLine.ledgerDimensionItem
PurchLine.ledgerDimensionReceipt
Purchline.unLinkAgreementLinePrompt
PurchLine.validateField
PurchLine::setProjSalesPrice
PurchPrepayTable.updateAdvanceApplicationRemaining
PurchPurchOrderJournalPost.updateSourceTable
PurchReqCreate.init

METHOD
PurchReqLine.setDefaultDimension
PurchReqLine.validateWrite
PurchReqTable.init
PurchReqTableForm.new
PurchRFQCaseTable.init
PurchTable.jumpRefIntercompanySalesOrder
PurchTableForm_DeliverySchedule.updatePurchLineTable
PurchTableInteraction.enableHeaderReceive
PurchTableType.validateDelete
PurchTableUpdateFromPurchReqLineMap.update
ReqTransPoMarkFirm.setPurchBuyerGroupId & updatePurchBuyerGroup
RetailGroupMemberLineHelper.internalCreateOrUpdateOrRemoveRetailGroupMemberLine
RetailLabelDP.insertTmpTable
SalesCalcAvailableDlvDates.mainOnServer
SalesConfirmJournalPost.updateSourceTable
SalesCopying.editCopy
SalesCopying.editMarkAll
SalesCopying.initReturnOrderFromCustomer
SalesCreateQuotation.canClose
SalesDropShipmentCancel.removeMarking
SalesEditLines.canClose
SalesFormletterParmData.reArrangeLines
SalesFormletterParmData.reArrangeSplit
SalesFormletterParmData.reArrangeSplit
SalesFormLetterProvider.checkJournal
SalesInvoiceDP.insertGiroInformation

METHOD
SalesInvoiceJournalCreate.checkDocumentData_PL
SalesInvoiceJournalPostBase.postLine
SalesLine.resetInvent
SalesLineType.initDimensionsSpecificDefaulting
SalesLineType.initReleasedProductSpecificDefaulting
SalesPackingSlipDP.setSalesPackingSlipDetailsTmp
SalesPackingSlipJournalCreate.updateJournalLine
SalesPackingSlipJournalCreate.updateJournalLine
SalesPackingSlipJournalPost.insertBackorderLine
SalesPackingSlipJournalPost.interCompanyPost
SalesPackingSlipJournalPost.updateJournalLine
SalesPurchSummaryModel_Account.createNewJournal
SalesQuotationCalcTax_Sales.construct
SalesQuotationEditLinesForm.postUpdate
SalesQuotationLineType.initFromProjTable
SalesQuotationLineType.initReleasedProductSpecificDefaulting
SalesQuotationTable.modifiedField
SalesQuotationTableForm.createFromTemplate
SalesQuotationUpdate_Cancelled.run
SalesQuotationUpdate_Lost.run
SalesTable.initFromCustTableMandatoryFields
SalesTable.jumpRefIntercompanyPurchaseOrder
SalesTable.setShipCarrierFromLogisticsLocation
SalesTable.update
SalesTableForm.interCompanyAutoCreateOrders
SettlementPair.createSettlementForDebitOrCreditTrans

METHOD
SmaServiceFunctionLine_Transfer.createJournalLine
SmaServiceFunctionLine_Transfer.postJournalTransType
SmaServiceOrderCreate.createServiceOrderLine
SmaSubscriptionGenerator::postTrans
smmBusRelTable.relation2Vendor
smmBusRelTable.updateQuotations
SmmCampaignBroadcast::validate
SmmOpportunityStatusUpdate.updateOpportunity
smmOpportunityTable\Methods\openQuotation
SmmProjectCreate.createSingleProject
SmmProjectCreate.createSingleProject
SmmUpdateBusRel.updateFromVendTableSFA2
SubledgerJournalTransferController.run
SuppItem.calcSuppItem
SuppItem::newSuppItem
SuppItemCreate::newSuppItemCreate
Tax.post
Tax.saveAndPost
TaxFreeInvoice_Invoice.updateAndPost
TaxPost.moveTaxLineToNewOwner
TaxPost.saveAndPostFromTmpTaxWorkTrans
TaxPost.saveAndPostFromTmpTaxWorkTrans
TaxVoucherService.postTaxOnErrorAccount
TradePackingSlipJourChain.createRelationship
TradeTotals.calc
TradeTotals.updateOrderBalances

METHOD
VendAccountStatementIntDP.processReport
VendEditInvoice\DataSource\VendInvoiceInfoTable\Methods\write
VendInvoiceMatching.initExpectedValues
VendInvoiceWFParticipantProviderExpend.resolve
VendOpenTrans.editMarkTrans
VendorInvoiceLineSourceDocLineItem.calculateSourceDocumentAmountMap
VendPaymentJournalDP.insertDataFromLedgerJournalTrans
VendPaymentJournalDP.insertDataFromSpecTrans
VendPromissoryNotePost.postNextStep
VendProvisionalBalanceDP.calculateAmounts
VendProvisionalBalanceDP.insertVendProvisionalBalanceTmp
VendProvisionalBalanceDP.processReport
VendTransListDP.ProcessReport
VendVoucher.createInvoiceJournal
WHSDocumentRouting.printDocument
WhsLoadLineInventTransValidator.checkLoadLineInventTransConsistencyOnInventoryUpdate
WHSLoadLineUpdater.initAndInsertLoadLine
WhsShipConfirm.createASNItems
WhsWarehouseRelease.createLoadLines
WHSWorkExecute.executeShortPick
WHSWorkExecuteDisplayLPReceiving.displayForm
WHSWorkExecuteDisplayLPReceiving.displayNextForm
WHSWorkExecuteDisplayMovementByTemplate.displayForm
WhsWorkExecuteForm.createLabel
WmsJournalCheckPostReception.postTrans
WmsOnlineCountingServer.getMovement

METHOD
WmsOnlineCountingServer.handleLine
WmsOrderCreate.updateCreatewmsOrder
WrkCtrResourceAbilityMapController.loadData

Enumerations made extensible

These enumerations have been made extensible in this update.

ENUMERATION
ABCModel
AccountOrder
AmountUnit
CostCalculationRateSubtype
CurrencyGainLossAccountType
CustInterestCodeSource
CustPaymentType
DirViewLocationNodeType
InterestCalcAccountChoice
InventTransPostingType
MCRCustSearchType
PaymentStub
PaymentStubInclAll
ProjCompletePrincip
ProjMatchingPrincip
RetailPOSSeedDataType
SalesQuotationStatus
TMSApptStatus
TMSCommunicationType

Additional extensibility enhancements

In addition to the refactored methods, the following extensibility enhancements have been made.

- CustVendSettle: set variables protected instead of private.
- Data manipulation method not raising event: WHSLocationDirective.loopLocDirLines.
- Data manipulation method not raising event: WhsWorkCreate.createTempLine.
- Map extension: LogisticsPostalAddressMap.
- Map extension: PurchReqLineMap.
- Metadata change: DataEntityView/ProjWBSActivityEstimatesEntity.Is Public = Yes.
- Metadata change: DataEntityView/ProjWBSActivityEstimatesEntity.Public Collection Name = ProjWBSActivityEstimates.
- Metadata change: DataEntityView/ProjWBSActivityEstimatesEntity.Public Entity Name = ProjWBSActivityEstimates
- Metadata change: /Data Model/Data Entities/EcoResProductAttributeValueEntity.IsPublic = Yes.
- Metadata change: Form/WHSLoadPlanningListPage/FormDesign/FormDesign/FormActionPaneTabControl/ActionPaneTabShipReceive.Neede permission.
- Metadata change: WHSContainerLine, Relations WHSLoadLine & WHSShipmentTable.On Delete.
- Metadata change: WHSContainerTable, Relation WHSShipmentTable.On Delete.
- Project pricing: complete uptake of new pricing find methods.

Extensibility changes in Dynamics 365 for Finance and Operations update 8.0.3

2/18/2021 • 2 minutes to read • Edit Online

This is a list of extensibility features that were implemented in Dynamics 365 for Finance and Operations update 8.0.3. For more information about the schedule of changes that support extensibility, see Application extensibility plans.

Refactored methods to support extensibility

These methods have been refactored to support extensibility through chain of command, delegates, or by providing access to members.

METHOD	
AccountingSourceExplorerProcessor.filterEntries	
AgreementClassification.init	
AgreementConfirm.createLineVolumeCommittmentHistory	
AgreementConfirm.newAgreementConfirm	
Agreementline.findLineForAutoMatch	
Agreementline.getAgreementLinesForOrderLine	
AgreementLine.getAgreementLinesForPurchReqLine	
Agreementline.getAgreementLinesList	
Bank_FR.checkControlText	
Bank_IT.checkCIN	
Bank_IT.checkRegistrationNum	
BankAccountTrans.insert	
BankAccountTrans.update	
BankChequeCopy.fillTmpChequePrintout	
BankChequePrint.printDocument	
BankPaymAdviceReportGeneratorVend	

METHOD	
BankReconciliationMatchRuleLine.getFieldsOfSysGenMatchRuleLineOfDoc	
BankReconciliationMatchRuleLine.getFieldsOfSysGenMatchRuleLineOfDoc	
BankReconMatchingRuleAutoProcessor.getSearchedDocumentIdList	
BankReconMatchingRuleAutoProcessor.getSearchedDocumentIdList	
BankVoucher.createBankAccountTrans	
BankVoucher.createBankAccountTrans	
BomCopyToProd.copyTo	
BudgetPlanLineFieldActiveViewMapping.getBudgetPlanLineFieldName	
BudgetTransaction.openLinesInExcel	
ChequeController.init	
CustAccountStatementExt.main	
CustAccountStatementExtController.includeOnStatement	
CustAccountStatementExtController.insertCustAccountStatementExtTmp	
CustAccountStatementExtController.setCommonData	
CustAccountStatementExtController.tmpCustVendTrans	
CustAccountStatementExtUIBuilder.build	
CustAuditorDP.setCustAuditorTmp	
CustCollectionJourDP.insertCustCollectionJourDP	
CustCreditLimit.Balance	
CustInterestNoteDp.processReport	
CustInvoiceJour.printJournal	
CustInvoiceTable.checkCreditLimit	
CustPackingSlipJour.interCompanyUpdate	

METHOD	
CustPaymEntry.updateConditionalControls	
custPostInvoicejob.custPostInvoiceUpdate	
CustTrans.reverseTransact	
CustVendCheque.createBankChequePaymentTrans	
CustVendCheque.createBankChequePaymentTrans	
CustVendCheque.initTmpChequePrintout	
CustVendCheque.output	
CustVendCheque.output	
CustVendChequeSlipTextCalculator.getChequeDocLength	
CustVendSumForPaym.validateSEPATransaction	
CustVendSumUpJournal.createTrans	
CustVendVoucher.post	
DimDerDistRuleProjectTimesheetsExt.populateDimAllocListIntercompany	
DimDerJourRuleProjectTimesheetsExt.getDefaultDimensionAllocation	
DirPartyVerification.selectionChanged	
EcoResCategoryTreeDatasource.initializeAvailableCategoriesMap	
EcoResProductCreate.writeMoreFields	
EcoResProductDetailsExtended.initInventDimensionsMetadataEntries	
ElectronicPaymentRemitExport_BR.construct	
ForecastPuch	
ForecastSales.accountConsumption	
ForecastSales.accountDisc	
ForecastSales.accountIssue	
ForecastSales.accountSales	

METHOD	
InventPosting.accountItemLedgerDimension	
InventSupply.init	
InventTrans.insertReturnTransOrigin	
InventTransferParmLine - several methods	
InventTransferUpd::updateLines	
InventTransFormHelper.formQueryAddDynalink	
InventTransWMS_Pick::updateInventServer	
InventUpd_Physical::updatePhysicalReceiptTrans	
InventUpdate.writeInventTrans	
InventUpdate::createInventTransOriginAndReferences	
InventValueReportPopulateItem::findReportLine	
JmgRegistration.JmgJobTable	
JournalizingDefinitionManager.newJournalizingDefinitionManagerPurch	
JournalStatic.initializeDataModel	
LedgerFinancialJournalReportDPBE.calcDebCredTotals	
LedgerFinancialJournalReportDPBE.processReport	
LedgerJournalDP.insertJournalTransForLedgerJournalTable	
LedgerJournalDP.insertLedgerJournalTmp	
LedgerJournalEngine.newJournalActive	
LedgerJournalTrans checkAllowPosting	
LedgerJournalTransUpdateBank.setBankVoucherSource	
LedgerJournalTransUpdateBank.updateNow	
LedgerJournalTransUpdateBankLC.addBankVoucher	
LedgerPostingGeneralJournalController.transferLines	
LedgerPurchaseJournalReportDPBE.insertIntoTempTable	

METHOD	
LedgerSalesJournalReportDPBE.processReport	
LedgerTransFurtherPosting.createLedgerJournalTransFromGenJour	
LedgerTransVoucher.getSubledgerVoucherLinkDataSource	
LedgerTransVoucher.getSubledgerVoucherLinkDataSource	
LedgerTransVoucher.getVoucherDateRange	
LedgerVoucherObject.updateLedgerPostingJournal	
LedgerVoucherTransObject.checkRounding	
Markup.insertMarkupTrans	
MarkupTrans.MarkupTable.MarkupCode.Lookup	
PaymSchedCalc::init*	
PaymSchedCalc_Line::createTransaction	
PdsApprovedVendorListCheck.newBasedOnTableType	
PmfFormulaCoBy.run	
PmfFormulaCoBy.ValidateField	
PmfProdCoBy.ValidateField	
PmfProdCoBy.ValidateWrite	
PriceDiscAdmSearch	
PriceDiscPolicyDialog.runPolicyDialog	
ProdBOM.checkIsItemsReleased	
ProdBOM::update	
ProdJournalProd.Insert	
ProdPurch.createPurchTable	
ProdUpdHistoricalCost_Process.checkValidCoBy	
ProdUpdReportFinished::updateBOMConsumption	
ProdUpdStartUp,getListOfBOMJournals	

METHOD	
ProdUpdStatusDecrease_StartUp.reverseBOMStartUp	
ProjBudgetParticipantProvider.resolveByProject	
ProjBudgetParticipantProvider.resolveByProjectHierarchy	
ProjBudgetParticipantProvider.resolveByRootProject	
ProjCaseActivitiesHandler.smmActivities_onValidatedDelete	
ProjControlPeriod.forecast	
ProjControlPeriod.forecast	
ProjControlPeriodCostGroup.totalBudgetMinusActual	
ProjControlPeriodCostGroup.totalBudgetMinusActual	
ProjectPosting.costLedgerDimension.	
ProjectPosting.getProjectLedgerDimension.	
ProjForecastEmpl.initValue	
ProjForecastReduceHour.constructQuery	
ProjFundingSource.setInvoiceLocation	
ProjGroup.initFromProjType	
ProjIntercompanyCustomerInvoiceCreator.createInvoiceLine	
ProjIntercompanyTransactionSelection.runQuery	
ProjIntercompanyTransQuery.buildExpenseQuery	
ProjIntercompanyTransQuery.buildHoursQuery	
ProjIntercompanyTransQuery.buildVendorInvoiceLinesQuery	
ProjInventJournalTransMapForm.checkActivity	
projInvoiceChooose.setProposalJour	
ProjInvoiceChoose.doRevenue	
ProjInvoiceChoose.updateInvoiceTotal	
ProjInvoiceProposalCreateLines.isRevenueTrans	

METHOD	
ProjInvoiceProposalCreateLinesBase.createProposalTrans	
ProjInvoiceProposalCreateLinesBase.doOnAccount	
ProjInvoiceTable	
ProjLedger.classdeclaration	
ProjPostItemJournal.projTransCreate	
ProjProjectsListPage.CtrlStages	
ProjProjectsListPageInteraction.enableButton	
ProjProjectsListPageInteraction.showButton	
ProjStatusUpd.main	
ProjStatusUpd.new	
ProjTable - ProjTable datasource.write	
ProjTable.clicked	
ProjTable.editSubProj	
ProjTable.editSubProj	
ProjTableCreate.close	
ProjTableCreate.run	
ProjTableCreate.write	
ProjTableCreate.write	
ProjTableLookup.ProjProjectLookup.init	
PSAProjInvoiceDP.insertPSAProjInvoiceHeaderTmp	
PSAProjInvoiceTaxTmp.insertPSAProjInvoiceTmpForTax	
PsaProjProposalSelection	
PurchAgreementAutoCreate::construct	
PurchAutoCreate.setPurchTable	
PurchAutoCreate_PurchReq.initializeAndCreatePurchLine	

METHOD	
PurchAutoCreate_PurchReq.initializeAndCreatePurchLine	
PurchAutoCreate_ReleaseFromAgreement.updateFinDimFromAgreemHeader	
PurchCreateFromSalesOrder.shouldCreatePurchOrder	
PurchFormLetter::main	
PurchFormLetter::main	
PurchFormletterParmDataPackingSlip::reSelectLines	
PurchFormletterParmDataPackingSlip::selectChooseLines	
PurchFormletterParmDataPurchOrder::selectChooseLines	
PurchInvoiceJournalPost.checkBeforePostingLine	
PurchInvoiceJournalPost.updateSourceLine	
Purchline.createline	
PurchOrderLineBudgetControlPolicy.canCheckBudget	
PurchReceiptsListDP.setPurchReceiptsListDetailsTmp	
PurchReceiptsListDP.setPurchReceiptsListHeaderTmp	
PurchRFQAcceptJournalPost.updatePurchReq	
ReqCalc.covCalcDim	
ReqTrans.createTransferDemand	
ReqTransPoMarkFirm.createProdRoute	
RetailPeriodicDiscount.ClassDeclaration	
RetailTransactionServiceOrders.cancelCustomerOrder	
Return.ReturnDispositionCodeId::validate	
SalesAutoCreate::construct	
SalesFormLetter.mainOnServer	
SalesFormLetter.mainOnServer	
SalesFormLetter::main	

METHOD	
SalesFormletterParmDataConfirm::selectChooseLines	
SalesFormletterParmDataInvoice::mayJournalTransBePosted	
SalesFormletterParmDataInvoice::selectChooseLines	
SalesFormletterParmDataPackingslip::selectChooseLines	
SalesInvoiceDP.insertIntoSalesInvoiceTmp,insertIntoSalesInvoiceHeaderFooterTmp	
SalesInvoiceJournalCreate.createJournalLine	
SalesLine.CheckItemId	
SalesLine.ValidateWrite_Server	
SalesLine::calcLineAvailQty	
SalesLine::createFromTmpFrmVirtualIL	
SalesLineType.SalesLineType	
SalesPackingSlipDP.setSalesPackingSlipDetailsTmp	
SalesPackingSlipDP.setSalesPackingSlipHeaderTmp	
SalesPackingSlipDP.setSysDocuBrandDetailsRegular	
SalesPackingSlipDP.setSysDocuBrandDetailsRegular	
SalesPackingSlipJournalPost.updateInventory	
SalesQuotationLineType_Proj.validateProjTransTypeItem	
SalesQuotationProjTable data source SalesQuotationline	
SalesQuotationTableForm.createABSFromTemplate	
SalesTable.setLocation	
SalesTable2LineUpdate.update	
SalesTable2LineUpdate.update	
SalesTable2LineUpdatePrompt.initpriceDiscUpdateTriggers	
smmActivitiesEventHandler	
SuppitemTable Table Cache Lookup property	

METHOD	
Table PurchPrepayTable.updateAdvanceApplicationRemaining	
TransactionReversal_Cust.reversal	
TransactionReversal_Cust.reversal	
TransactionReversal_Vend.reversal	
TransactionReversal_Vend.reversal	
TransactionReversal_Vend.reversal	
TsTimesheetAddFavorites.addToFavorites	
TsTimesheetCreate.createTimesheetLine	
TSTimesheetEntry.initFields	
TSTimesheetFavorites.createTimesheetLines	
TSTimesheetLine.setCategoryIdFromActivity	
VendInvoiceDocumentDP.insertVendInvoiceDocumentTmp	
WHSLoadLine.validateStatus	
WHSLoadLineAllocationProcessor.allocateLoadLine	
WHSPostEngine.validateAnyDimAboveLocationMissing	
WhsWarehouseRelease.createShipmentsForAllSalesOrders	
WhsWarehouseRelease.createShipmentsForTransferOrders	
WhsWorkCreateLP.createTempTable	
WHSWorkCreateProdPut.createTempTable	
WHSWorkExecuteDisplay.buildNextDimensionCaptureControl	
WHSWorkLine::cancelLine	
WmsArrivalCreateJournal::createWMSJournalTransFromTmp	
WmsArrivalOverviewGeneration::updateOverviewInformation	
WmsJournalCheckPostReception::initJournal	

METHOD	
WMSOrderTrans::adjustQtyWMSOrderTrans	
WMSOrderTrans::createNewWMSOrderTrans	
WMSOrderTrans::insertOrUpdate	
WMSOrderTrans::updateWMSOrderTrans	
WmsPickingList_OrderPickDP.insertIntoTempTable,setWMSPickingList_OrderPickTmpTemplate	
WmsPickingList_OrderPickDP.setWMSPickingList_OrderPickTmpTemplate	
WrkCtrlScheduler_Proj.loadJob	
WrkCtrScheduler_Prod.saveOperation	
WrkCtrScheduler_Prod.saveOrder	

Enumerations made extensible

These enumerations have been made extensible in this update.

ENUMERATION
BankReconMatchRuleLineSysGeneratedType
BankReconMatchRuleLineSysGeneratedType
BankReconMatchRuleLineSysGeneratedType
ItemNumAlternative
JmgRegistrationErrorMode
MCRCustSearchType
ModuleSalesPurch
ModuleSalesPurch
ProjStatusRule
PurchRFQUpdateType
TAMVendRebateItemCode
TMSLoadBuildSupplyDemandType

Additional extensibility enhancements

In addition to the refactored methods, the following extensibility enhancements have been made.

- Increase EDT string size for EcoResProductSearchName
- Change CacheLookup property to NotInTTS for AssetLedgerAccounts
- Change CacheLookup property to Found on TaxOnItem, TaxJurisdiction, TaxGroupData, and TaxData, and AssetLedgerAcounts

Extensibility changes in Dynamics 365 for Finance and Operations update 8.0.2

2/18/2021 • 2 minutes to read • Edit Online

This is a list of extensibility features that were implemented in Dynamics 365 for Finance and Operations update 8.0.2. For more information about the schedule of changes that support extensibility, see Application extensibility plans.

Refactored methods to support extensibility

These methods have been refactored to support extensibility through chain of command, delegates, or by providing access to members.

METHOD
AccDistProcessorProjectExtension.allocateExistingDistribution
AccDistProcessorProjectExtension.createDistributionLists
AccDistProcessorProjectExtension.ledgerDimensionAllocationList
AgreementConfirmationDP.processReport
Bank_FR.checkControlText
Bank_IT.checkCIN
Bank_IT.checkRegistrationNum
CustVendCheque.initTmpChequePrintout
FBSpedFileCreator_Contabil_BR.createRecordI052
HierarchyTemplateCopying_proj.createFromHierarchySource
HierarchyTreeLookup.datasource smmActivities.init
InventDim.validateFieldCombination
InventTransWMS_Register
InventUpd_Financial.updateFinancialIssue
InventUpd_Registered
JmgPostStandardSystem.PostProjTime
MarkupAllocation.run

METHOD
MarkupTrans. MarkupTrans(datasource).active
PdsBatchAttribByItem.checkDuplicateAttributes
PdsBatchAttribByItem.validateFieldValue
PdsBatchAttribReserve.linkActive
PdsBatchAttributes.Datasource.PdsBatchAttributes.linkactive
ProjInvoiceProposalCreateLines.closeOK
ProjInvoiceProposalInsertLines.doCost
ProjInvoiceProposalInsertLines.doEmpl
ProjInvoiceProposalInsertLines.doItem
ProjInvoiceProposalInsertLines.doOnAccount
ProjPeriodPostingLedgerSales.enteItem
ProjPeriodPostingLedgerSales.enterCost
ProjPeriodPostingLedgerSales.enterEmpl
ProjPeriodPostingLedgerSales.enterRevenue
ProjPeriodPostingLedgerSales.run
ProjPlanVersionsManager.CopyHierarchy
ProjPlanVersionsManager::copyHierarchy
ProjPlanVersionsManager::createDraftVersion
ProjPlanVersionsManager::createTemplateHierarchy
PurchAutoCreate_PurchReq.initializeAndCreatePurchLine
PurchOrderLineSourceDocumentLineItem.calculateSourceDocumentAmountMap
PurchRFQPriceDiscAdmCreate.createPriceDiscAdmTrans
SalesFormLetter.validate
SalesTable2LineUpdatePrompt.salesTableFieldModifiedHandler
SalesUpdateRemain.cancelOpenOrderLinesDeliveryRemainder

METHOD
WHSShipmentTable.createShipmentNotes
WHSUnShipLoadLineTmpDataCreator.createTmpLoadLineInventoryFromContainerLines

Additional extensibility enhancements

In addition to the refactored methods, the following extensibility enhancements have been made.

- Support for extensions to map: CustVendTrans
- Support for extensions to map: CustVendTransOpen
- Support extensibility for SQL statement: PriceDiscAdmCheckPost.postJournal

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Extensibility changes in Finance and Operations update 8.0.1

2/18/2021 • 2 minutes to read • Edit Online

This is a list of extensibility features that were implemented in Dynamics 365 for Finance and Operations update 8.0.1. For more information about the schedule of changes that support extensibility, see Application extensibility plans.

Refactored methods to support extensibility

These methods have been refactored to support extensibility through chain of command, delegates, or by providing access to members.

METHOD
Class ProjControlPeriod::PeriodInsert
Class ProjInvoiceChoose::doSalesLine
Class CustInvoiceJour::printFreeTextJournal
Class ProjCostControl.createEmptyTransactionType
Class ProjEstimate::autoGenerateEstimateLinesFromTask
Class ProjEstimateDataContract.updateEstimates
Class ProjForecastBudget.Run
Class ProjHierarchyProvider.preDeleteHierarchy
Class ProjPlanVersionsManager::CopyTasks
Class ProjPlanVersionsManager.createDraftFromPublishedVersion
Class ProjPlanVersionsManager.PublishQuotationSubHierarchy
Class ProjPlanVersionsManager.CreateDraftVersion
Class ProjTask.addTask
Class ProjInvoiceProposalCreateLines.performTransTypeSelectionCtrlLookup
Class VendOpenTrans.editMarkTrans
Class CustVendReversePosting.reverseTaxWithholdTrans
Class CustVendSettle.postPennyDiff

METHOD
Class CustVendSettle.processStillOpenTransactions
Class InventTransferOrderOverviewDP.insertTmp
Class LedgerJournalTransCost.LedgerJournalTrans.Create
Class ProjAdjustmentSelect.dialog
Class ProjEstimate.syncEstimateLinesFromTask
Class ProjEstimateDataContract.UpdateEstimates
Class ProjForecastTransferFromWbs.transferToForecast
Class ProjWizardActivityCtrl.insertDBOnServer
Class ProjTaskEstimatesSynchronizer.calcTotalEstimateLineHours
Class ProjTaskEstimatesSynchronizer.countNumberOfHourEstimateLines
Class ProjTaskEstimatesSynchronizer.syncExistingHourEstimatesWithTask
Class ProjTaskEstimatesSynchronizer.syncHourEstimatesWithTaskEffort
Class ProjWbsCostPlanningServerActions.executeDataRetrievalAction
Class ProjWbsCostPlanningServerActions.getProjectCategoryTypes
Class ProjWbsSchedulePlanningServerActions.executeAction
Class ProjWbsSchedulePlanningServerActions.executeDataRetrievalAction
Class ProjStatisticCalc.validate
Class WHSInventReserve.insert
Class WHSInventReserveDelta.insert
Class ProjInvoiceProposalCreateLines.performTransTypeSelectionCtrlLookup
Class ProjJournalTransEmpl - Datasource: ProjJournalTrans.Validate
Class LedgerJournalEngine.initTaxItemGroup
Class LedgerJournalEngine.initValue
Class ProjJournalTrans.mergeResourceDimensionDefault
Class ProjTask.setTaskinfo

METHOD
Class ProjJournalName.standardJournalName
Form ProjInvoiceProposalCreateLines.performTransTypeSelectionCtrlLookup

Other extensibility enhancements

In addition to the refactored methods, the following extensibility enhancements have been made.

- Support variable number of decimals - InventTestLowerLimit
- Support variable number of decimals - InventTestLowerTolerance
- Support variable number of decimals - InventTestStandardValue
- Support variable number of decimals - InventTestUpperLimit
- Support variable number of decimals - InventTestUpperTolerance
- Support to skip prompt on transaction reversal
- Enable extension of PSAProjQuotationApproval workflow

Extensibility changes in the Finance and Operations version 8.0

2/18/2021 • 9 minutes to read • Edit Online

Hard-sealed application models

In Dynamics 365 for Finance and Operations version 8.0, all of Microsoft's application models have been hard-sealed. Overlayered code in these models will now produce compilation errors. The only supported customization model is through extensions. If you cannot customize these models through extension, then you will have to make a request to Microsoft to enable extensibility by changing the standard application.

The following table includes a list of models that are now hard-sealed with this release.

MODULE	MODEL
ApplicationCommon	ApplicationCommon
ApplicationSuite	Electronic Reporting Application Suite Integration
ApplicationSuite	Foundation Upgrade
ApplicationSuite	Foundation
ApplicationSuite	SCMControls
ApplicationSuite	Tax Books Application Suite Integration
ApplicationSuite	Tax Engine Application Suite Integration
CaseManagement	CaseManagement
Currency	Currency
DataImpExpApplication	DataImpExpApplication
DataUpgrade	DataUpgrade
Directory	Directory
Directory	SecurityReports
GeneralLedger	GeneralLedger
Ledger	Ledger
PersonnelManagement	PersonnelManagement
ProcessGuide	ProcessGuide

MODULE	MODEL
Retail	Retail
SourceDocumentation	SourceDocumentation
SourceDocumentationTypes	SourceDocumentationTypes
Subledger	Subledger
Tax	Tax

Enumerations that have been made extensible

The following changes were made to support extending enumerations:

- Many enumerations in the standard application have been made extensible. An enumeration is made extensible by setting two properties on the enumeration. The **IsExtensible** property is set to **Yes**, and the **UseEnumValue** property is set to **No**.
- Some enumerations represent state. New façade methods have been added to help enable adding enumeration values by extension. For information about how to extend an enumeration, see Add values to enums through extension.
- Some application code that uses enumerations was changed to support extensibility. Common changes include:
  - Removing **throw** exception statements in the default case of a switch to allow post-event subscription.
  - Adding **SysExtension** support for extension.
  - Adding explicit delegates.

ENUMERATION
BOMConsumpType
BOMFormula
BOMType
ChequeFormType
CostGroupType
CustAccountStatement
CustMandateScheme
CustVendDisputeStatus
DispositionAction
ItemCalcType
KMCollectionAnswerStatus

ENUMERATION
KanbanEventType
LedgerAccrualPeriod
LogisticsAddressElement
LogisticsLocationEntityType
NoneBeginTransEnd
PSAInvoiceFormats
PdsCumulationPeriod
PdsRebateProgramType
PdsRebateTransaction
PdsUnitType
PriceDiscSystemSource
ProdFlushingPrincipBOM
ProdFlushingPrincipItem
ProdReservation
ProjAccountTypeCost
ProjAccountTypeSales
ProjAccountType
ProjJournalType
RevenueContributionMargin
SMATransactionType
SysPolicyRuleEnum
SysPolicyRuleTypeEnum
SysPolicyTypeEnum
TAMRebateAmtType
TAMVendRebateStatus

ENUMERATION
TMSRecordType
Voided
WMSJointShippingType
WMSReferenceType

Data manipulation methods that do not raise DataEvents or missing insert, update, delete pre- and post-data events

As a general practice, you use data methods on tables to raise events that can be used for extending the application. The code base has not always followed this practice. For example, the **doInsert**, **doUpdate**, and **doDelete** data methods and certain table implementations did not make a call to **super()** in the data method.

The **insert**, **update**, and **delete** methods on the type classes have been refactored. Changes were made so that **super()** is called more consistently in data methods. These changes enable extensions to be added to these methods, so that pre- and post-events are now available for extension. The tables where the **insert**, **update**, and **delete** events were enabled for extension are listed in the following table.

TYPE, NAME, DATA SOURCE, AND METHOD
Form ProjTableCreate.ProjTable.write
Form ReturnTable.ReturnTable.leaveRecord
Form SalesQuotationProjTable.SalesQuotationTable.leaveRecord
Form SalesQuotationTable.SalesQuotationTable.leaveRecord
Form SalesTable.SalesTable.leaveRecord

Refactored methods to support extensibility

These methods have been refactored to support extensibility through chain of command, delegates, or by providing access to members.

TYPE, NAME, AND METHOD
Class AgreementConfirm_Sales.startConfirm
Class AssetChangeGroup.updateAssetGroupInfo
Class AssetPost.createAssetTransForPost
Class AssetSplit.getUpdatedSplitValueModel
Class AssetTableMethod.init
Class AssetTableMethod_SL.calc

TYPE, NAME, AND METHOD
Class AxSalesLine
Class BankPaymCancel.serverRun
Class BomSearch.New
Class BomSearch_BOMCopyType.New
Class Commission.run
Class CostSheetPanel.build
Class CreateInvoiceJournalPost.createFixedAsset
Class CustAccountStatementIntDP.printingAmountMST
Class CustCreditLimit.balanceEstimate
Class CustCreditLimit.calculateBalance
Class CustCreditLimit_SalesTable.New
Class CustInterestCreate
Class CustVoucher.post
Class DimensionDerivationRule.buildDimensionCombination
Class EcoResProductInformation.main
Class EcoResProductReleaseManager.setAndSaveRetailProductProperties
Class EcoResProductValidator.isEssentialFieldValuesSet
Class FormLetterServiceController.newFromContract
Class FormletterJournalPost.postLineDiscount
Class Graphics_WrkCtrCapBooking.insertLoad
Class Graphics_WrkCtrCapBooking.loadGroupReservations
Class Graphics_WrkCtrCapBooking.loadNumReservations
Class InterCompanyPostPurch.construct
Class InterCompanySyncPurchLineType
Class InterCompanySyncPurchTableType.setSalesTableData

TYPE, NAME, AND METHOD
Class InterCompanySyncPurchTableType
Class InventAgeDimDP.createOrMergeInventAgeDimTmp
Class InventAgeDimDP.insertOrMergeInventAgeDimTmp
Class InventAgingCmdAggregateSelected.execute
Class InventCostItemDim.initInventSettlement
Class InventCostReport.newInventCostReport_CostBaseType
Class InventCountCreateItems.run
Class InventDimCtrl_Frm.clearInvisibleRanges
Class InventItemPriceActivationTaskActivateSim.activateOneInventItemPriceSim
Class InventItemPriceSim.moveSimulatedToCurrent
Class InventLedgerPostingDefinitionEntityHelper.inventAccountTypeX2InventAccountType
Class InventMov_SalesQuotation.isQuotationQtyEditable
Class InventProductDimensionLookup.dimEDT2FieldId
Class InventProductDimension
Class InventQualityManagementBlock.actOnAssociations
Class InventQualityManagementCreate.createOnRegistration
Class InventQualityManagementCreate.createQualityOrder
Class InventQualityManagementCreate.generateQualityOrders
Class InventQualityManagementCreateInvent.generateQualityOrdersWithDiscrimination
Class InventQualityMgmtCreateNonInvent.generateQualityOrdersWithDiscrimination
Class InventQualityOrderReopen.main
Class InventQualityOrderReopen.run
Class InventQualityOrderValidate.main
Class InventQualityOrderValidate.run
Class InventQualityReferenceTypeSales.isEligibleForQualityManagement

TYPE, NAME, AND METHOD
Class InventQualityReferenceTypeSales.supportsInventoryBlocking
Class InventQualitymanagementCreate.createPerQualityAssociations
Class InventSumReCalcItem.updateActualInventSum
Class InventTestAssociationTable.checkAccountRelation
Class InventTestAssociationTable.initRecord
Class InventTrackingDimTracingCriteria.initFromArgs
Class InventTransLine.insert
Class InventTransferMulti.run
Class InventTransferMultiReceive::main
Class InventTransferMultiShip.buildParmFromWMSShipment
Class InventTransferMultiShip.runUpdate
Class InventTransferOrderOverviewDP.insertTmpTable
Class InventTransferUpdShip.updateInventTransferLine
Class InventUpd_Physical.updatePhysicalIssue
Class InventUpd_Physical.updatePhysicalReturnedReceipt
Class InventUpd_Picked.updatePickInventTrans
Class InventUpd_Reservation.whsUpdateReserveMore
Class InventUpdate.raiseOnHandChangingOnPhysicalStatusUpd
Class InventUpdate.updateDimReservePhysical
Class InventUpdate.updateTransDimTransferReceipt
Class InventUpdate.writeInventTransAutoDim
Class InventValueReportDP.processInventValueReportTmpLine
Class InventoryMainAccDimensionListProvider.populateMainAccountDimensionList
Class LedgerBalanceQueryGeneralJournal.addToBalanceTotals
Class LedgerBalanceQueryGeneralJournal.createQuery

TYPE, NAME, AND METHOD
Class LedgerJournalCheckPost.checkJournal
Class LedgerJournalCheckPost.postJournal
Class LedgerJournalDP.insertJournalTransForLedgerJournalTable
Class LedgerJournalDP.insertLedgerJournalTmp
Class LedgerJournalGetTrans.createLedgerJournalTrans
Class LedgerVoucherObject.addTrans
Class LedgerVoucherTransObject.check
Class LogisticsLocationSelectForm.construct
Class LogisticsLocationSelectForm.main
Class LogisticsPostalAddressFormHandlerExt.onNewParameters_delegate
Class MCRItemListGeneration.generateItemListLines
Class MCRItemListGeneration.generateItemListLines
Class MCRMarginAlert.skipMarginCalc
Class Markup.mcrDeleteNonUser
Class MarkupAllocationSelectionManager.setQueryRanges
Class PSAProjInvoiceDP.insertPSAProjInvoiceTmp
Class PSAProjInvoiceDP.insertProformaPSAProjInvoiceTmp
Class PdsApprovedVendorListCheck.newBasedOnTableType
Class PlanActivityTimeCalculation.calculatePlanActivityTime
Class PordJournalCreateBOM.createLinesProdBOM
Class PriceDisc.accountRelation
Class PriceDisc.findDiscAgreement
Class PriceDisc.findDisc
Class PriceDisc.findPriceAgreement
Class PriceTypeConverter.priceTypeToPriceGroupType

TYPE, NAME, AND METHOD
Class PrintMgmtReportFormatSubscriber.add
Class PrintMgmtReportFormatSubscriber.populate
Class ProdBOM.prodFlushingPrincipItem2BOM
Class ProdJournalCreateBOM.createLinesInventTrans
Class ProdJournalCreateBOM.createLinesInventTrans
Class ProdJournalCreateBOM.createLinesProdBOM
Class ProdJournalCreateBOM.dialog
Class ProdJournalCreateBOM.validate
Class ProdJournalFormTransBOM.setupCWFormControl
Class ProdPickListController.prePromptModifyContract
Class ProdPicklistDP.insertValues
Class ProdStatusType_Released.checkPostJournal
Class ProdTableListPageInteraction.getEnabledControls
Class ProdUpdReportFinished.updateBomConsumption
Class ProdUpdReportFinished.updateRouteConsumption
Class ProdUpdSplit.createSplitToProduction
Class ProdUpdStartUp.getListOfBOMJournals
Class ProdUpdStartUp.updateBOMConsmption
Class ProjInvoiceDP.insertIntoProjInvoiceTmp
Class ProjInvoiceProposalInsertLines.run
Class ProjInvoiceProposalInsertLines::run()
Class ProjPlanVersionsManager
Class ProjPostItemJournal::projTransCreate
Class ProjProposalTotals.calc
Class PsaProjInvoiceDP::insertProformaPSAProjInvoiceTmp

TYPE, NAME, AND METHOD
Class PsacustomerRetention.createFeeTransactionForProposal
Class PurchAgreementGenerateReleaseOrder.check
Class PurchAgreementGenerateReleaseOrder.validatePurchLinesWithPurchQty
Class PurchAutoCreate.construct
Class PurchAutoCreate.construct
Class PurchAutoCreate_RFQ.construct
Class PurchAutoCreate_SalesProjectItemReq.createLine
Class PurchAutoCreate_SalesProjectItemReq.createPurchLine
Class PurchCancel.parmPurchTable
Class PurchCancel.run
Class PurchCopying.deleteLines
Class PurchCreateFromSalesOrder
Class PurchFormLetterParmData.createParmLine
Class PurchFormLetterParmDataInvoice.createParmLineAndSubLines
Class PurchFormletterParmData.reSelectLines
Class PurchFormletterParmDataApproveJournal.updateQueryBuild
Class PurchFormletterParmDataInvoice
Class PurchInvoiceCreate.createJournalLine
Class PurchInvoiceJournalCreate.checkInvoicePolicies
Class PurchInvoiceJournalCreate.checkMatching
Class PurchInvoiceJournalPost.createFixedAsset
Class PurchInvoiceJournalPost.lateMatchPackingSlip
Class PurchLineType.validateWrite
Class PurchLineVersioningFieldSet.isChangeConfirmationRequired
Class PurchOrderLineSourceDocumentLineItem.calculateSourceDocumentAmountMap

TYPE, NAME, AND METHOD
Class PurchOrderLineSourceDocumentLineItem.calculateSourceDocumentAmountMap
Class PurchOrderLineSourceDocumentLineItem.calculateSourceDocumentAmountMap
Class PurchPackingSlipDP.createProductReceiptLines
Class PurchPackingSlipJournalPost.selectFormletterJournalTrans
Class PurchRFQCaseAutoCreate.newAutoCreate
Class PurchReApprovalPolicyRuleFieldList.addTable2Hierarchy
Class PurchReApprovalPolicyRuleFieldList.addTable2Hierarchy
Class PurchSelectLinesManager.passSets
Class PurchTableInteraction.enableHeaderPurchase
Class PurchTableInteractionHelper.getJournalEnquiryButtons
Class PurchTableInteractionHelper.getUpdateJournalButtons
Class PurchaseOrderResponseConsume.checkIfPurchLinesRequireUpdate
Class PurchaseOrderResponseConsume.checkIfResponseLineCannotBeConsumedAndUpdateConsumptionState
Class PurchaseOrderResponseConsume.consumeFirstPurchaseOrderResponeLineAndInitiateArchivingOnPurchLine
Class PurchaseOrderResponseConsume.consumeRemainingPurchaseOrderResponseLines
Class PurchaseOrderResponseConsumeLine.checkIfSelectedPurchLinesRequireUpdate
Class ReqCalc.covCodeQty
Class ReqCalc.insertItemInventSum
Class ReqCalc.insertItemInventTrans
Class ReqTransFormPo.validateFromInventLocationId
Class ReqTransPoMarkChangeToRFQ.DialogPostRun
Class ReqTransPoMarkFirm.createPurchLine
Class ReqTransPoMarkFirm.setPurchTable
Class RetailAssortmentLookupTask.explodeAssortments
Class RetailCreateLinesFromProductsToAdd.createPeriodicDiscount

TYPE, NAME, AND METHOD
Class RetailCreateLinesFromProductsToAdd.loadLines
Class RetailPackagePurchManagement.createLines
Class RetailProductPropertyManager.saveInventTableAndRelated
Class RetailProductPropertyManager.validateWriteOnInventTable
Class RetailSalesOrderCalculator.saveSalesOrder
Class RetailSalesOrderCalculator.setPriceOnCurrentLine
Class RetailSalesQuotationCalculator.saveSalesQuote
Class RetailSalesQuotationCalculator.setPricesOnCurrentLine
Class ReturnTableInteraction.enableControl
Class RouteCopyToRoute.insertRouteOpr
Class SMAServiceFunctionLine_Transfer.checkJournalType
Class SMAServiceFunctionLine_Transfer.postJournalType
Class SMAServiceFunctionLine_Transfer.sumjournals
Class SMAServiceOrderCreate.createServiceOrderLine
Class SalesAutoCreate_ReleaseFromAgreement.createSalesTable
Class SalesCancelOrder.run
Class SalesCopying.copy
Class SalesCreateOrderFromCutomer.main
Class SalesFormLetter.mainOnServer
Class SalesFormLetterParmData.createParmLine
Class SalesFormLetterReport.construct
Class SalesFormletterParmData.reSelectLines
Class SalesFormletterParmDataInvoice.reSelectInit
Class SalesInvoiceDP.invoiceTxt
Class SalesInvoiceDP.itemId

TYPE, NAME, AND METHOD
Class SalesLineCopyFromSource.updateCopiedLine
Class SalesLineType.setReservation
Class SalesLineType.setSalesStatus
Class SalesLineType.syncPurchLine
Class SalesPackingSlipDP.createSalesPackingSlipLines
Class SalesPackingSlipJournalPost.addToInventReportDimHistory
Class SalesPurchLineInterface.setPriceAgreement
Class SalesQuotationCopying.copyHeader
Class SalesQuotationEditLinesForm_Sales_Confir.createSalesTable
Class SalesQuotationEditLinesForm
Class SalesQuotationLineType_Sales.validateWrite
Class SalesTableListPageInteraction.setButtonInterCompany
Class SalesTableType.checkUpdate
Class SalesTableType.interCompanyMirror
Class SmmCampaignQueries
Class SmmLeadUpdate
Class SmmOpportunityLink
Class SmmUpdateBusRel.updateFromCustTableSFA2
Class TradeCurrencyConversionPrompt.construct
Class TradeLineRenumbering.renumber
Class TradeTotals.calc
Class VendDocumentLineInterface.setPurchaseQty
Class VendInvoicePolicyValidation.policyViolationMessage
Class VendProvisionalBalanceDP.processReport
Class WHSPool.pickFromWorkCenter

TYPE, NAME, AND METHOD
Class WHSShipConfirm.createUOMStructure
Class WHSWorkExecute.pickLicensePlateHandledByLP
Class WhsInventOnHandReserve.changeReservation
Class WhsInventOnHandReserve.setMovement
Class WhsPackForm.buttonPack_clicked
Class WhsPostEngineBase.createLoadFromShipment
Class WhsShipConfirm.createInventTransferParmLineTMS
Class WhsShipConfirm.createInventTransferParmLine
Class WhsWorkExecute
Class WmsBillOfLadingDP::insertIntoTempTable
Class WmsOrderTransType_OutputDontPostTransfer.updateParentMovement
Class WrkCtrCapResHandler.hasNewCapacityReservation
Class WrkCtrCapResHandler.loadCapacityReservations
Class WrkCtrReservedSum.calcReservationSumGroupId
Class WrkCtrReservedSum.calcReservationSumGroupId
Class WrkCtrReservedSum.calcReservationSumWrkCtrId
Class WrkCtrReservedSum.calcReservationSumWrkCtrId
Class WrkCtrScheduler_Prod.saveOperation
Class createParmLinesFromTransferLinesOnLoad
Class smmCampaignQueries.lookupClass
Entity EcoResProductDimensionGroupEntity.dataSourceDimensionFieldId
Entity InventProductDefaultOrderSettingsEntity.insertEntityDataSource
Entity InventProductSiteSpecificOrderSettingsEntity.insertEntityDataSource
Entity PSAActualEntity.createQuery_LaborConsumptionQty
Entity PSAActualEntity.createQuery_LaborConsumption

TYPE, NAME, AND METHOD
Entity PSAActualEntity.createQuery_PlLaborCost
Entity PSAActualEntity.createQuery_PlLaborQty
Entity PSAForecastEntity.createQuery_LaborConsumptionForecastQty
Entity PSAForecastEntity.createQuery_LaborConsumptionForecast
Entity PSAForecastEntity.createQuery_PlLaborForecastCost
Entity PSAForecastEntity.createQuery_PlLaborForecastQty
Form BOMCalcDialog.updateDesign
Form EcoResProductCreate.releaseProductToCompany
Form InventItemOrderSetup.InventItemSetupSupplyType.editOrderType
Form InventLocationIdLookup.InventDim_DS.init
Form InventLocationIdLookup.InventLocation_DS.init
Form InventNonConformanceTable.init
Form InventNonConformanceTableCreate.InventNonConformanceTable.write
Form InventQualityOrderTableCreate.allowEdit
Form InventQualityOrderTableCreate.refreshCaller
Form InventTestAssociationTable.initRecord
Form InventTransPick\TmpInventTransWMS.validateWrite
Form LedgerTransVoucher.updateQueryForProject
Form MarkupAllocation.init
Form MarkupAllocation_VendInvoiceTrans
Form PdsBatchAttributes.PdsBatchAttributes.linkActive
Form PriceDiscAdmTable.init
Form PriceDiscTable.appendInventCriteria
Form PriceDiscTable.buildOrderLineFilter
Form PriceDiscTable.buildSearchFilter

TYPE, NAME, AND METHOD
Form PriceDiscTable.isLineFilterEnabled
Form PriceDiscTable.retrieveRelationType
Form ProdParmStartUp.ProdParmStartUp.active
Form ProjCreditNoteSelect.editMark
Form ProjTableCreate.ProjTable.write
Form PurchCreateFromSalesOrder.initFields
Form PurchUpdateRemain.closeOk
Form ReqTransPoMarkFirm.init
Form RetailAddItems.closeOk
Form RetailColorGroupTable.RetailColorGroupTrans.recordHasChanges
Form RouteLookupOprNum.init
Form VendEditInvoice.invoiceAccountModified
Form VendEditInvoice.run
Form VendOpenTrans.editMarkTrans
Form WrkCtrCapResGraphDialog.setParm
Map BomCalcTransMap.displayUnitId
Table AssetTable.lookupAccountNum
Table AssetTrans
Table CaseDetailBase.validateWrite
Table EcoResProductMasterConfiguration.existWithSameConfigUnit
Table FormletterJournalTrans.getLinePrefix
Table InventItemPriceSim.autoSalesPrice
Table InventQualityOrderLine.adjustInt
Table InventQualityOrderTable.createInventQualityOrderLines
Table InventQualityOrderTable.initFromReference

TYPE, NAME, AND METHOD
Table InventQualityOrderTable.initQtyFromAssocation
Table InventTestAssocationTable.checkAccountRelation
Table InventTestAssocationTable.validateWrite
Table InventTrans.insertReturnTransOrigin
Table InventTransOrigin.createOrigin
Table InventTransferParmLine.createPickLines
Table InventTransferParmLine.createReceiveLines
Table InventTransferParmLine.createShipLines
Table JmgStampjournalTrans
Table JmgTermReg.createJournalSignIn
Table JmgTermReg.update
Table LedgerJournalTrans.delete
Table LedgerJournalTrans.validateWrite_Server
Table PriceDiscAdm.getEntityAutoReportFieldGroupName
Table PriceDiscAdm.getEntityJournalNumberFieldName
Table PriceDiscAdmTrans.CheckAccountRelation
Table PriceDiscAdmTrans.checkItemRelation
Table PurchLine.setPriceDisc
Table RouteVersion.selectRouteVersion
Table SalesLine.checkItemId
Table SalesLine.getSourcingFields
Table SalesLine.setPriceAgreement
Table SalesLine.setPriceDisc
Table SalesLine.setSourcingFields
Table SalesQuotationLine.IsCategoryBased

TYPE, NAME, AND METHOD
Table SalesQuotationLine.mcrCreateFromTmpFrmVirtualFromContract
Table SalesQuotationLine.setPriceAgreement
Table SalesQuotationLine.setPriceDisc
Table Salesline.splitReturnLine
Table SuppItemCreate.createLine
Table TmpInventTransMark.packTmpMark
Table VendInvoiceMatchingLine.initFromPurchLine
Table VendTable.updateOnHold
Table WHSInvent.checkNonPhysicalDims
Table WHSShipmentTable.consolidateShipments
Table WHSShipmentTable.transferShipment
Table WHSTmpPackingLine.addTmpPackLine
Table salesLine.initFromProjTable
Table smmBusRelTable.updateReferences
Table smmLeadTable
Table smmOpportunityTable

Maps enabled for extensibility

New patterns have been introduced for maps implementation that will allow you to add fields and methods by extensions. Details on how this is done is available in the documentation both with maps that are used as interfaces and for versioning implementations.

The following table lists the maps and related tables where changes have been applied for enabling extensibility.

MAPS
CustVendSettlement
JmgStampTransMap
PriceDiscResultFields
SalesPurchLine

Inventory dimensions

This release made minor improvements to the new model for adding inventory dimensions, all targeted at supporting more scenarios through extensions.

CHANGE
BOM hierarchy works only with the config dimension
Form BOMDesigner should use field group for showing dimensions
Form EcoResProductSearchLookup should use field group for showing dimensions
Form FactureJournal_RU should use field group for showing dimensions
Form InventDimParmFixed.InventDimensionXXFlag.Style is incorrect
Form InventItemOrderSetup should use field group for showing dimensions
Form InventTransferParmPick should use field group for showing dimensions
Form InventTransferReleaseOrderPicking should use field group for showing dimensions
Form KanbanCreateScheduled should use field group for showing dimensions
Form KanbanJobPickingListPart should use field group for showing dimensions
Form KanbanRules should use field group for showing dimensions
Form LeanPeggingTree should use field group for showing dimensions
Form MCRItemDisplay should use field group for showing dimensions
Form MCRPriceDiscGroupItem should use field group for showing dimensions
Form PlanActivityServiceWizard should use field group for showing dimensions
Form ProdBOMVendor should use field group for showing dimensions
Form PurchAgreementGenerateReleaseOrder should use field group for showing dimensions
Form PurchAgreementHistory should use field group for showing dimensions
Form PurchComplementaryInvoice should use field group for showing dimensions
Form PurchRFQCompareLineDimensions should use field group for showing dimensions
Form PurchTable.TrackingDimesions has incorrect spelling
Form PurchVendorPortalAllResponse should use field group for showing dimensions
Form PurchVendorPortalConfirmedOrders should use field group for showing dimensions

CHANGE
Form PurchVendorPortalOriginalOrder should use field group for showing dimensions
Form PurchVendorPortalRequests should use field group for showing dimensions
Form PurchVendorPortalResponses should use field group for showing dimensions
Form ReqDemPlanEasyItemAllocator should use field group for showing dimensions
Form ReqOutboundIntercompanyDemand should use field group for showing dimensions
Form ReqSupplyDemandScheduleFilters should use field group for showing dimensions
Form RetailVariantLookup should use field group for showing dimensions
Form RouteVersionFeasibility should use field group for showing dimensions
Form SMAAgreementTable should use field group for showing dimensions
Form SalesAgreementGenerateReleaseOrder should use field group for showing dimensions
Form SalesAgreementHistory should use field group for showing dimensions
Form SalesComplementaryInvoice should use field group for showing dimensions
Form SalesLineDeliveryDetails should use field group for showing dimensions
Form SalesQuotationProjTable should use field group for showing dimensions
Form SalesQuotationTable should use field group for showing dimensions
Form SalesTable should use field group for showing dimensions
Form TAMFundManagement should use field group for showing dimensions
Form TAMTradePromotions should use field group for showing dimensions
Form VendEditInvoice should use field group for showing dimensions
Form VendJournalMatch_PackingSlip should use field group for showing dimensions
Form WHSLoadPlanningWorkBench should use field group for showing dimensions
Form WHSLoadPlanningWorkbench should use field group for showing dimensions
Form WHSLoadTable should use field group for showing dimensions
Form WHSLoadTable should use field group for showing dimensions
Form WHSProdWaveTableManageBOMPool should use field group for showing dimensions

CHANGE
Form WHSWorkTable should use field group for showing dimensions
Form WMSOrderTransUnPick should use field group for showing dimensions
Form WMSPickingRegistration should use field group for showing dimensions
Report InventAging does not support extra dimensions
Table EcoResProductVariantStaging.StagingIdx need extra dimension fields

Other changes

The following table lists additional changes that have been made for extensibility.

CHANGE
Add filter interface: form InventQualityOrderTable
Address management: Adding new address fields
AxMaps - TradePostalAddress - partyTable
Bank Trans Comments - BankReconciliationDataInitializer
Cancellation Log Requirements - Update Sales Deliver Remainder
Extend the grouping mechanisme from purch req line to purch line
Extend the splitting mechanisme from purch req line to purch line
Allow multiple funding sources in conjunction with item requirements
Implementing exchange rate provider framework
Make the PriceDiscPartyCodeType extensible in all usages
Make the PriceDiscProductCodeType extensible in all usages
Table RetailChannelTable does not have ReplacementKey
Table RetailSeasonTable CreateRecIdIndex True
Modify index: table InventTestAssociationTable
Entity UnitOfMeasureEntity switched to public
Entity UnitOfMeasureTranslationEntity switched to public

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Extensibility changes in Finance and Operations, Enterprise edition 7.3

2/18/2021 • 16 minutes to read • Edit Online

This topic lists the extensibility features that were released in Dynamics 365 for Finance and Operations, Enterprise edition 7.3. For more information about the schedule of changes that support extensibility, see Application extensibility plans.

Soft-sealed application models

This release marks the last release before all models will become hard-sealed, and as a step toward this all application models are now soft-sealed. Soft-sealed models still allow for making overlayered code, but warnings will be generated when you compile the overlayered code.

> **NOTE**
> You can still overlayer code, but extension is the recommended approach.

The following table includes a list of the models that are soft-sealed with this release.

MODULE	MODEL
ApplicationCommon	ApplicationCommon
ApplicationSuite	Electronic Reporting Application Suite Integration
ApplicationSuite	Foundation Upgrade
ApplicationSuite	Foundation
ApplicationSuite	SCMControls
ApplicationSuite	Tax Books Application Suite Integration
ApplicationSuite	Tax Engine Application Suite Integration
CaseManagement	CaseManagement
Currency	Currency
DataImpExpApplication	DataImpExpApplication
DataUpgrade	DataUpgrade
Directory	Directory
Directory	SecurityReports

MODULE	MODEL
GeneralLedger	GeneralLedger
Ledger	Ledger
PersonnelManagement	PersonnelManagement
Retail	Retail
SourceDocumentation	SourceDocumentation
SourceDocumentationTypes	SourceDocumentationTypes
Subledger	Subledger
Tax	Tax

Hard-sealed application models

With this release, almost all application core models have been hard-sealed. Overlayered code in these models will now produce compilation errors. The only supported customization model is through extensions. If you cannot customize these models through extension, then you will have to make a request to Microsoft to enable extensibility by changing the standard application.

TThe following table includes a list of models that are now hard-sealed with this release.

MODULE	MODEL
AccountsPayableMobile	AccountsPayableMobile
ApplicationWorkspaces	ApplicationWorkspaces
BankTypes	BankTypes
BusinessProcess	BusinessProcess
Calendar	Calendar
ContactPerson	ContactPerson
CostAccounting	CostAccounting
CostAccountingAX	CostAccountingAX
Dimensions	Dimensions
DirectoryUpgrade	DirectoryUpgrade
DOM	DOM
ElectronicReporting	ElectronicReporting

MODULE	MODEL
ElectronicReportingAppSuiteIntegration	ElectronicReportingAppSuiteIntegration
ElectronicReportingCore	ElectronicReportingCore
ElectronicReportingDotNetUtils	ElectronicReportingDotNetUtils
ElectronicReportingForAx	ElectronicReportingForAx
ElectronicReportingMapping	ElectronicReportingMapping
ExpenseMobile	ExpenseMobile
FinancialReporting	FinancialReporting
FinancialReportingEntityStore	FinancialReportingEntityStore
FiscalBooks	FiscalBooks
InventoryDimensionConversion	InventoryDimensionConversion
Measurement	Measurement
PaymentPredictor	PaymentPredictor
PerformanceTool	PerformanceTool
Personnel	Personnel
PersonnelCore	PersonnelCore
PersonnelMobile	PersonnelMobile
PersonnelUpgrade	PersonnelUpgrade
Policy	Policy
Project	Project
ProjectMobile	ProjectMobile
RegulatoryServices	RegulatoryServices
SCMMobile	SCMMobile
SelfHealing	SelfHealing
SelfHealingRules	SelfHealingRules
SystemHealth	SystemHealth

MODULE	MODEL
TaxEngine	TaxEngine
UnitOfMeasure	UnitOfMeasure
WMSAdvancedMigration	WMSAdvancedMigration

Enumerations that have been made extensible

The following changes were made to support extending enumerations:

- Many enumerations in the standard application have been made extensible. An enumeration is made extensible by setting two properties on the enumeration. The **IsExtensible** property is set to **Yes**, and the **UseEnumValue** property is set to **No**.
- Some enumerations represent state. New façade methods have been added to help enable adding enumeration values by extension. For information about how to extend an enumeration, see Add values to enums through extension.
- Some application code that uses enumerations was changed to support extensibility. Common changes include:
  - Removing **throw** exception statements in the default case of a switch to allow post-event subscription.
  - Adding **SysExtension** support for extension.
  - Adding explicit delegates.

ENUMERATION
AssetCalendarYearType
AssetDepreciationConvention
AssetDepreciationMethod
AssetPeriodMonth
AssetSoldScrap
AssetStatusLVPFilter
AssetTransType
BaseDataProd
BOMRouteVersionSelect
BudgetPlanGenerateSource
BudgetPlanScenarioAccessLevel
BudgetPlanScenarioAttribute
BudgetTransactionColumnType

ENUMERATION
BusinessEvent_ActivityJournal
BusinessEvent_CustomerInvoice
BusinessEventRelievingMethod
CollabSiteEntityType
CollabSiteSharePointType
CostSheetNodeType
CreditCardAddEdit
CreditCardApprovalType
CreditCardDupCheckResult
CreditCardPaymType
CustAssessment
CustCollectionLetterCode
CustInterestFeeType
CustomerTransactionType
CustSettlementPriorityAttribute
CustSettlementTrans
CustTransRefType
CustVendDisputeStatus
CustVendForeignExchRefIndicator_US
CustVendGatewayOperatorOFACIndicator_US
CustVendorBlocked
CustVendOutPaymTrade
CustVendPaymStatus
CustVendSecondaryOFACIndicator_US
DimensionCacheScope

ENUMERATION
DirPartyRoleType
DirPartyRoleView
DistributionProcess
DistributionProcessingState
DistributionStatus
EPProjUpdateSubProjStage
HcmPositionForecastStatusSelection
JournalizingDefinitionLedgerEntryTypeId
LedgerBalanceExportFieldSeparator
LedgerBalanceExportHeaders
LedgerBalanceExportHiddenFields
LedgerBalanceExportInvertSign
LedgerBalanceExportSubcomponents
LedgerBalanceExportSubtotals
LedgerColumnType
LedgerConsDim
LedgerConsolidateAccountSource
LedgerDataExportFormat
LedgerReconciliationStatus
LedgerShowCurrency
LedgerSIEFileType
LedgerTransactionType
LedgerTransEnigneBuildQuery
LogisticsAddressElement
LogisticsAddrZipCodeImportCountryRegion

ENUMERATION
LogisticsLocationEntityType
LogisticsLocationSelectSourceType
MCROrderEventType
PaymManDocType
ProjAccountType
ProjAccountTypeCost
ProjAccountTypeSales
ProjActualVsForecastCategory
ProjActualVsForecastValue
ProjAlertType
ProjAssignConflictStatus
ProjCategoryType
ProjChoose
ProjContractType
ProjCostSales
ProjDimensionStrType
ProjectReportingAnalyticsWorkspaces
ProjEstimateMethod
ProjExportToExcelDimension
ProjFoundMethod
ProjFunctionType
ProjFundingRuleType
ProjInvoiceFrequency
ProjInvoiceProposalsTransSelectionTypes
ProjLevelFilterOption

ENUMERATION
ProjListLedgerTransType
ProjOrigin
ProjOriginOnAcc
projPostedProjectTransactionsListFilter
ProjProdTableListPageFilter
projProjectsListFilter
ProjQuotationTransTypeFilter
ProjResourceCapacityBooking
ProjResourceViewType
ProjSelectTransOnAcc
ProjServerProcessStatus
ProjStatementTypeSub
ProjStatus
ProjTransType
ProjValConnection
ProjValidateType
ProjViewSubProjects
ProjYearEndOptions
PSAActivityDisplayDefault
PSAActivityParent
PSADetailLevel
PSAExpenseProjValCategoryType
PSAInvoiceFormats
PSAProjInvoiceDetailGrouping
PSAProjInvoiceDetailSortBy

ENUMERATION
PSAProjOriginakVsCurrent
PSAPWPAssessment
PSAResAssignView
PSAResSchedStatus
PurchStatus
QuotationProjTransType
ReasonCodeAccountType
ResBasicSearchType
ResRollUpResourceType
ResTransferType
ResUtilizationCategoryEnum
RetailEventNotificationType
SourceDocument_ActivityJournal
SourceDocumentLine_ActivityJournalLine
SpecType
SyncProjTableAddSubProj
SyncProjTableDeleteSubProj
TrvCarRentalChargeType
TrvExpenseFilter
TrvExpenseReportGroupBy
TrvIntermediatePageOnCreateExpenseReport
TrvPayrollQtyOrDayes
TrvPBSTxtType
TrvPolicyViolationAction
TrvPosting

ENUMERATION
TrvPostStatus
TrvTaxType
TrvUDFDisplayOption
TSApprovalLevel
TSDocumentStatusReset
TSTimesheetFilter
TSTimesheetLineFilterType
TSTimesheetListPageFilters
TSVendorPerformanceThreshold
TypeOfCreditmaxCheck
VendInvoiceCloseCommand
VendorInvoiceSearchOptions
VendOutPaymFeeDistribution

Foundation changes were made to improve support for extensible enumerations. The **SysPlugin** framework was enabled for enumerations where **IsExtensible** is set to **Yes**. Views were enabled with new name-based syntax for enumerations.

Data manipulation methods that do not raise DataEvents or missing insert, update, delete pre- and post-data events

As a general practice, you use data methods on tables to raise events that can be used for extending the application. The code base has not always followed this practice. For example, the **doInsert**, **doUpdate**, and **doDelete** data methods and certain table implementations did not make a call to **super()** in the data method.

The **insert**, **update**, and **delete** methods on the type classes have been refactored. Changes were made so that **super()** is called more consistently in data methods. These changes enable extensions to be added to these methods, so that pre- and post-events are now available for extension. The tables where the **insert**, **update**, and **delete** events were enabled for extension are listed in the following table.

TABLE AND METHOD
BOM
BOMTable
BOMVersion

TABLE AND METHOD
ProjTableType.delete
ProjTableType.update
Route
RouteOpr
RouteTable
RouteVersion
SalesLineType::interCompanyResetDeliverNow
Table ForecastSales

Exposing class members

Additional private members are now available for customization as a result of changes to access modifiers and parm methods. The chain of command platform feature enables extension class access to protected methods and members. For more information about chain of command, see Extensible X++: Chain of Command.

MEMBER
BankPaymCancel.custTransToCancel
CustCollectionLetterCancel - method queryBuildUpdate
CustCollectionLetterPost - method queryBuildUpdate
CustCollectionLetterPost - method updateFee
CustCollectionLetterPost - method validateCollectionLetter
CustInterestCancel - method updateQuery
CustInterestHelper - method getFeeLedgerAccount
CustInterestHelper - method getInterestRecord
CustInterestHelper - method getPostingProfile
CustInterestHelper - method getTransLedgerAccount
CustInterestHelper - method getTransLineLedgerAccount
CustInterestHelper - method getVerDetailLedgerDimensionByIntTrans
CustInterestPost - method postVoucher

MEMBER
CustOutPaymControlController
CustVendCreatePaymJournal - method dialogAddInvoiceSelectionCriteriaFields
CustVendPaymProposal - method createProposalLine
CustVendPaymProposal - method parmLedgerJournalId
CustVendPaymProposalLineInsertSetManager - variables
CustVendPaymProposalOrg - variables
CustVendPaymProposalTransferToJournal - method trackSpecTransForUpdate
CustVendPaymProposalTransferToJournal - variables
Form ProjWorkBreakdownStructure
Form/Class CustPaymModeSpec
Form/Class VendPaymModeSpec
InventUpd_ChildReference.initUpdate
InventUpd_ChildReference.parmInventDimId
LogisticsLocationFormHandler.callerGetAddressRecord
ProjAdjustmentSelect.newQuery.addAdditionalHeaderRange
ProjAdjustmentSelect.processProjCostTrans
ProjAdjustmentSplit.deleteTransaction
ProjAdjustmentSplit.splitTransaction
ProjInvoiceChoose.parmprojInvoiceProjId
ProjProposalTotals.projInvoiceExchRate
SalesInvoiceJournalCreateBase
smmActivityCreate.createOrPrompt
Table SalesQuotationTable.canSubmitToWorkflow
VendorInvoiceLineSourceDocLineItem.initializeProjectFields
WHSWorkUserSession.WorkExecuteMode

Construct methods with throw statements

Some **construct** methods were implemented with **throw** statements if there was a missing implementation for a given type. This doesn't work well with extensibility, so to mitigate this, **construct** methods were changed so that they do not throw exceptions. These methods are now to open for extensibility through class augmentation or by post-event subscription.

OBJECT
AddressZipCodeImport
CaseCategoryHierarchyTree
CustInterestCancel
CustInterestHelper
CustInterestPost
CustOutPaymControlController
CustTransQueryBuild
CustVendCreatePaymJournal_Cust
CustVendFindSettlements
CustVendOpenTransBalances.new
CustVendOpenTransManager.initFromCaller
CustVendPaymProposalTransferToJournal
CustVendTransQueryBuild
Form PdsApprovedVendorList
Form WhsContainerTable.init
FormletterJournalCreate.newSalesJournalCreate
FormLetterJournalPost.newPostSales
InventUpd_Physical::newInventMovement
InventUpd_Physical::newProdReleaseLossProfit_RU
LogisticsLocationSelectForm
PdsApprovedVendorListCheck.newBasedOnTableType
ProjInvoiceChoose

OBJECT
ProjTrans
PurchReqAutoCreate.newAutoCreate
PurchTableForm_Project
SalesQuantity
SalesTotals
WHSReservation

Find methods with throw statements

Some **find** methods were implemented with **throw** statements if there was a missing implementation for a given type. This does not work well with extensibility, so to mitigate this, **find** methods were changed so that they do not throw exceptions. These methods are now to open for extensibility through class augmentation or by post-event subscription.

METHODS
TradePostalAddress.partyTable

Extracted method to open for class extensions

The **Chain of Command** feature lets you create extension classes. Extensions classes offer a stronger way of extending than other options because they allow access to both protected and public methods and members. This provides more flexibility than extending through delegates or by pre or post event.

Within this group of changes, longer methods are extracted into smaller methods. The new methods have a more specific focus and you have more control over the scope of your extensions.

After the introduction of the **Chain of Command** feature, we suggest using extensibility by extracting methods instead of adding delegates because this approach provides a more versatile solution.

The following table lists the new methods that have been extracted and opened for building extension classes.

METHOD
AssetPost
BankPrintTestCheque
CustCreditLimit.showErrorMsg
CustVendCheque
CustVendChequeSlipTextCalculator
Form CustBankAccounts

METHOD
Form DirPartyQuickCreateForm.init
Form HierarchyDetail.contextChanged
Form HierarchyDetail: smmActiviate: initValue
Form HierarchyNameLookoup: Hierarchy: init
Form LedgerJournalTransDimension.init
Form ProjInvoiceProposalDetail.editInvoiceFormat
Form SalesCopying.upDateRemainderCache
Form SalesQuotationProjLinkWizard-> changeType
Form smmActivities: ResponsibleWorker_Overview: lookupReference
Form smmActivities: smmActivities::initValue
FormletterJournalPrint
HierarchyTemplateCopying.run
HierarchyTemplateCopying_CRM.copyActivity
HierarchyTemplateCopyingDialog.main
HierarchyTree
HierarchyTree.buildSubTree
InventDimCtrl_Frm_Mov_QualityOrder.mustEnableField
InventDistinctProductValidator.checkProductNotStopped
InventMovement.createProjLedgerForUpdateLedgerAdjust
InventTransferUpdShip::populateIssueReceiptDimensions
JournalFormTable
JournalizingDefinitionManager.newJournalizingDefinitionManagerCustomer
LedgerJournalCheckPost.checkJournal
LedgerJournalCheckPost.postJournal
LedgerJournalEngine.parmLedgerJournalTrans_Project

METHOD
LedgerJournalEngine_Server.addVoucher
LedgerJournalEngine_VendApprove.cancelVoucher
LedgerJournalizeReportDP_DE.processReport
LedgerJournalTransUpdateCust.checkAccountBlocked
LedgerJournalTransUpdateVend.checkVendorBlocked
LogisticsAddressFormatProcess.run
ProjAdjustmentUpdate.journalTableInsert
ProjAdjustmentUpdate_Post
projCategoryLookup.buildQuery_PSA_impl
ProjEstimate.add
ProjFormLetter_invoice.projPrintFormLetter
ProjIntercompanyVendorInvoiceCreator.createVendorInvoiceLine
ProjListTransDP.insertTmpProjTransList
ProjPostRevenueProposal.projTransCreate
projUnpostedTransactionsListPage.populateMenuFunction
PSAProjAndContractInvoiceController.preRunModifyContract
PSARetenetionRelease.run
PurchAutoCreate_SalesLine.setPurchTable
PurchCreateFromSalesOrder.run
PurchFormletterParmDataInvoice.createParmLinesAndTable
PurchTableForm_DlvScheduleSyncEnabled.syncDeliveryScheduleCommercialAttributes
ReqCalc.deleteItemRequirement
ReturnAcknowledgmentAndDocumentDP.insertIntoTempTable
ReturnAcknowledgmentAndDocumentDP.setReturnAckAndDocumentTemplate
SalesAgreementFormDatasourceManager.transferCustAccount

METHOD
SalesCopying.copy
SalesFormLetterParmData.createParmSubTable
SalesFormLetterParmDataInvoice.reSelectInit
SalesQuotationConfirmationDP.processReport
SalesQuotationConfirmationDP.setSalesQuotationDetailsTmp
SalesQuotationEditLinesForm.createParmLine
SalesQuotationEditLinesForm_Sales_Confir.createSalesLines
SalesQuotationTableForm_Sales.syncDeliveryScheduleCommercialAttributes
SalesQuotationTableType.validateField
SalesTable2LineUpdate.update
SalesTableForm.interCompanySetLineAccess
SalesTableForm_DlvScheduleSyncEnabled.syncDeliveryScheduleCommercialAttributes
SalesUpdateRemain.updateICDeliverRemainder
SmmProcessInstance.openForm
SubledgerJournalizerProjectExtension.createProjectActualHeader
Table CustTable.createOneTimeAccount
Table CustTable.lookupCustomer
Table InventPosting.salesAccount2AccountType
Table InventTable.lookupItem
Table ReqPO.update
Table SalesLine -> createFromSalesQuotationLine
Table SalesTable::initFromCustTableIL
Table smmBusRelTable.relation2Customer
Table smmBusRelTable.updateCustTable
Table TSTimesheetTrans.updateCommentsFromLineWeekUpdateTSTimesheetTrans

METHOD
Table TSTimesheetTrans.updateFromTimesheetLineWeekUpdateTSTimesheetTrans
Table VendTable.lookupVendor
Table WHSWorkTable -> deleteAndCleanupWorkLines
Table WHSWorkTable -> SetBlankFields
Table WHSWorkTable -> SetFields
Table WrkCtrActivity.getCompanyContext
Tax.insertLineInInternal
TransactionReversal_Cust.fld900_1_modified
whsLoadTemplateAssignmentForm: WHSLoadTable::clicked
WhsWorkExecuteDisplay.getNextFormState
WHSWorkExecuteDisplay.setBatchDetails
WhsWorkExecuteDisplayReturnOrder.buildReturnOrder
WhsWorkExecuteDisplayReturnOrder.displayForm

Changes using other methods to support extensibility

The group of changes in this section includes several different approaches to extensibility and represents the extensibilty changes made before **Chain of Command** was introduced. Some of the approaches used are extracting methods, adding "stub" methods, adding delegates, changing access modifiers on methods, and using the SysExtension framework. Please consult the implementation in places required for your customization to determine if the approach taken will work for your customization. In future releases, this group will be small, because we will primarily be using **Chain of Command**.

METHOD
AccDistRuleSaleOfProductExtendedPrice.parmLedgerDimensionAllocList
AssetPost.createInventorySoldTransaction
AssetSplit.validate
AxSalesLine.doSave
AxSalesLine.setLineAmount
AxSalesQuotationLine.setLineAmount
BankPaymCancel.main

METHOD
BankPaymCancel.run
BankPaymCancel.serverRun
BomCalcJob.main
CustCollectionLetterCancel.main
CustCollectionLetterCancel.run
CustCollectionLetterCreate.checkCustTransOpen
CustCollectionLetterCreate.createJournal
CustCollectionLetterPost.updateFee
CustCollectionLetterPost.validate
CustCollectionsExcelStatement.setTransactionWorksheetRow
CustInterestCancel.run
CustInterestCreate
CustInterestCreate.dialog
CustInterestCreate.runOnce
CustInterestHelper.getFeeLedgerAccount
CustInterestHelper.getVerDetailLedgerDimensionByIntTrans
CustInterestPost.postVoucher
CustInterestPost.run
CustInterestPost.updateFee
CustInterestPost.validateInterestTrans
CustInvoiceSpecDP::insertIntoTempTable
CustOutPaymControlController.init
CustPostInvoiceJob.custPostInvoiceUpdate()
CustSettlementPriorityProcessing
CustSettlementPriorityProcessing.markAllSelected

METHOD
CustSettlementPriorityProcessing.markTransByCreditNoteOnBillingClasses
CustVendCreatePaymJournal.initBalances
CustVendOpenTransManager::updateOriginatorForMarkedTrans
CustVendPaymProposalCalcPaym.calcPaymDueDate
CustVendReversePosting.updateNow
CustVendSettle.mustOffsetOriginalSummaryDistributions
CustVendVoucher.initLedgerPosting
DimensionDerivationRule.buildDimensionCombination
DimensionDerivationRule.initialize
EcoResProductReleaseManager.release
EcoResProductReleaseSessionBatch.runJob
EcoResProductReleaseSessionManager.executeOnServer
EcoResProductVariantCreationMgr.buildVariantSuggestions()
Form BankPaymCancel.closeOK
Form BOMChangeLine.init
Form BOMConsistOf: BOMCreate
Form ConfigPartOf: EcoResConfiguration
Form CustBankAccounts.write
Form CustCollections.showAgingIndicator
Form CustOpenTrans.editMarkTrans
Form CustOpenTrans.updateDesignStatic
Form CustPaymEntry.editIsMarkedForSettlement
Form CustPaymEntry.write
Form CustSettlementPrioritySetup.active
Form CustTable.PrintManagement.clicked

METHOD
Form EcoResProductImage.init
Form EcoResProductImage.setProductRecId
Form HRMAbsenceRequest.init
Form LedgerJournalTransAccrual.enableFields
Form LedgerJournalTransAccrual.LedgerJournalTransAccrual.clicked
Form LedgerJournalTransCustPaym: LedgerJournalTrans.active
Form LedgerJournalTransDaily: SettlementButton.clicked
Form LedgerJournalTransDimension.init
Form MarkupTable.init
Form MCRItemListCopying.copyLines
Form PCProductModelVersion
Form ProjInvoiceProposalCreateLines.modifiedTransFilter
Form ProjTransItem: ProjItemTrans.salesAmount
Form PurchCreateFromSalesOrder.SalesLine.included
Form ReqItemTable.init
Form SalesCopying.canClose
Form SalesCreateQuotation.setFieldsActive
Form SalesQuotationTable.init
Form SalesTable: SalesLine.write
Form SalesTable: SalesLine_DS.ItemId.lookup
Form: VendEditInvoice
FormletterJournalPos::newPostSales
FormLetterJournalPost.post
FormletterService.run
From ProjTable.init

METHOD
HierarchyTemplateCopying.copyHierarchy
HrmAbsenceRequestAction.run
InterCompanyUpdateRemPhys_PurchLine::synchronizeExternal
InterCompanyUpdateRemPhys_PurchLine::synchronizeInternal
InterCompanyUpdateStatus_PurchLine::synchronizeExternal
InterCompanyUpdateStatus_PurchLine::synchronizeInternal
IntrastatTransfer::calcCustVendInvoiceTransQty
InventDim::dimReportStr
InventMov_Transfer::checkUpdateEstimated
InventMov_Transfer::defaultDimension
InventMovement::checkNotSubDelivery
InventQualityManagementCreate.createQualityOrder
InventTransferParmLine::createShipLines
InventTransferParmLine::initFromInventTransferParmTable
InventTransferUpdShip::updateInventTransferLine
InventUpd_Estimated::createEstimatedInventTrans
InventUpd_Financial::newInventTransferLineReceive
InventUpd_Financial::newInventTransferLineShip
InventUpd_Financial::updateFinancialIssue
InventUpd_Financial::updateFinancialReceipt
InventUpd_Physical::newInventMovement
InventUpd_Reservation::updateReserveBuffer
InventUpd_Reservation::updateReserveFromForm
InventUpdate::whsUpdateDimReservePhysical
InventUpdate::whsUpdateWorkTransDimIssue

METHOD
LedgerJournalCheckPost.postJournal
LedgerJournalEngine - method preDelete
LedgerJournalEngine::findSettledAmount
LedgerJournalEngine_CustPayment.allowEditTrans
LedgerJournalEngine_CustPayment.initDefaultDimension
LedgerJournalEngine_CustPayment.write
LedgerJournalFormTable.verifyCanDelete
LedgerVoucherTransObject.newTransLedgerJournal
LogisticsPostalAddressFormHandler.main
Map SalesPurchLine.calcPrice2LineAmount
Map SalesPurchLine.setPriceAgreement
Markup.calc
MarkupAdjustment::main
McrPriceHistoryForm.insertPotentialTradeAgreements
OffsetVoucherCust.updateNow
PcGenerateBOMTableAndVersion.generate
PriceDisc.findDisc
PriceDisc::findItemLineDiscAgreement
PriceDisc::findItemPriceAgreement
PriceDisc::findMultiLineDiscServer
PriceDisc::newFromPriceDiscHeading
PriceDisc_LineDisc::findLineDiscAgreement
PriceDisc_Price::findPriceAgreement
PriceDiscAdmCheckPost.checkJournal
PrintMgmtHierarchy_Project.getParentImplementation

METHOD
ProjAdjustmentSplit.createNewTrans.getNewTotalCostAmount
ProjInvoiceJournalPost.createProjInvoiceRevenue
ProjInvoiceProposalInsertLines.doSalesLine
ProjPost.postCost
ProjPostCostJournal.projTransCreate
ProjPostCostTrans_AdjNeg.projTransCreate
PurchAutoCreate_PurchReq.prepareSort
PurchCalcItem.initListBOM
PurchFormLetter.mainOnServer
PurchFormLetterParmData::newChooseLines
PurchFormletterParmDataInvoice.createParmLineAndSubLines
PurchInvoiceJournalPost.checkSourceLine
PurchInvoiceJournalPost.postCustVend
PurchInvoiceJournalPost.postInventory
PurchLineType.initDimensionsSpecificDefaulting
PurchLineType.interCompanyMirror
ReqCalcExplodeSales.run
SalesAutoCreate_ReleaseOrder.createSalesLine
SalesConfirmDP::createData
SalesConfirmDP::printDimHistory
SalesConfirmDP::setSalesConfirmDetailsTmp
SalesCopying.copy
SalesFormLetter.mainOnServer
SalesFormletterParmData.calcAutomaticTotalDiscount
SalesFormletterParmDataPickingList.insertParmLine

METHOD
SalesInvoiceController::main
SalesInvoiceDP.insertIntoSalesInvoiceTmp
SalesInvoiceDP::insertIntoSalesInvoiceTmp
SalesInvoiceDP::loadCustPackingSlipTrans
SalesInvoiceDPBase::createData
SalesInvoiceJournalCreate::initInvoiceLineFromSourceLine
SalesInvoiceJournalPost::postCustVend
SalesLine::initReleasedProductSpecificDefaulting
SalesLineType.initDimensionsSpecificDefaulting
SalesLineType.interCompanyMirror
SalesLineType::checkDelete
SalesLineType::delete
SalesLineType::insert
SalesLineType::interCompanyMirror
SalesLineType::setSalesStatus
SalesLineType::update
SalesLineType::validateWrite
SalesPickingListJournalCreate::createJournalLine
SalesQuotationConfirmationDP::processReport
SalesQuotationCopying.copy
SalesQuotationDP::processReport
SalesQuotationLine.modifySalesQty
SalesQuotationLineType.initReleasedProductSpecificDefaulting
SalesQuotationToLineField.getFieldDescription
SalesTable2LineUpdate.update

METHOD
SalesTableListPageInteraction.setButtonInterCompany
SalesTableListPageInteraction.setButtonInvoice
SalesTableListPageInteraction.setButtonPickAndPack
SalesUpdateRemain
SalesUpdateRemain::updateDeliveryRemainder
smmActivityCreate.setup
smmAttendeeTable.insert
smmSalesCustItemStatisticsDP::processReport
SpecTransManager.updateFullSettlement
SubledgerJournalizer.loadAccDistTmpRelieveAccrual
SubledgerJournalizer.loadaccountingDistributionTmp
SubledgerJournalizer.recordSubledgerJourAccEntriesForRounding
SubledgerJournalizer.recordSubledgerJournalAccountEntries
SubLedgerJournalTransferUIBuilder::build
SuppItemCreate_SalesQuotation::createLine
Table - PurchLine.Insert
Table - PurchLine.Update
Table CostingVersion.validateField
Table CustBankAccount.lookupBankAccount
Table CustCollectionLetterJour.cancelCollectionLetterCodeCustTrans
Table CustInterestJour.feeLedgerDimension
Table CustInvoiceTable - method validateWrite
Table CustTable.blocked
Table CustTrans.reverseTransact
Table InventNonConformanceTable.setEditableFields

METHOD
Table InventPosting.accountItemLedgerDimension
Table InventTable.updateAutoSalesPrice
Table InventTestAssociationTable.checkDocumentType
Table InventTrans.accountLossProfitLedgerDimension
Table LedgerJournalTrans.checkVATNumJournal
Table MarkupTrans.checkMarkCode
Table PurchLine.convertCurrencyCode
Table ReqPO.validateWrite
Table SalesLine.checkAndUpdateLoadLines
Table SalesLine.setPriceDisc
Table SalesQuotationLine.setPriceDisc
Table SalesTable.createMarkupTrans
Table TmpInventTransMark.updateTmpMark
Table TMSAppointment.validateWrite
Table WHSLoadLine::inventTransferLine
Table WHSLoadLine::purchLine
Table WHSLoadLine::salesLine
Table WHSRFMenuItemTable.validateWrite
Tax.distributeTotalTax
TradeInterCompany::insertInterCompanyInventDim
TransactionReversal_Asset.checkStatusApplicable
TransactionReversal_Cust.main
TransactionReversal_Cust.reversal
TransactionReversal_CustVend.createCustVendTrans
TSTimesheetLineWeek.loadFromLine

METHOD
VendInvoiceInfoListPageMultiSelect.determineSelectState
WHSInventReserveDeltaLevelsEnumerator::moveNext
WhsPostEngineBase::createLoadFromShipment
WHSWorkCreateProdPut.insertProdParmforProdItem
WmsArrivalCreateJournal.createWMSJournalTransFromTmp
WMSPickingList_OrderPick.RunPrintMgmt
WrkCtrlScheduler_Prod.loadJobsDetail

Methods made hookable

Extensibility support has been extended for some methods that were not public and were not hookable. The following methods have been explicitly decorated with hookable behavior.

METHOD
Bank.checkBankIBAN
BankDepositCreateCancelJour.initValues
BankDepositCreateCancelJour.newDepositCreateCancelJour
BankPaymCancel.initParms
BankPaymCancel.updateCollectionsStatusAutomation
CustAccountStatementExtController
CustAccountStatementIntDP.insertCustAccountStatementIntTmp()
CustCollectionLetterPost.updateFee
CustInterestCreate.construct
CustProvisionalBalanceDP.insertCustProvisionalBalanceTmp()
CustSettlementPriorityProcessing
CustVendCreatePaymJournal.dialogAddDateSelectionFields
CustVendPaymReconciliationSetStatus
CustVendReversePosting.updateCustVendTrans
DataEntity EcoResTrackingDimensionGroupEntity.dataSourceDimensionFieldId

METHOD
EcoResProductCrossTableManager.saveValuesToProduct
EcoResProductImage.getImageFrom2Records
EUSalesListTransfer - 3 methods
Form EcoResProductCreate.applyTemplate
Form EcoResProductCreate.createData2Controls
Form PriceDiscTable.initFromCallerTable
Form ProjCostControl.setButtonVisibility
Form projPostedTransRelInfoFormPart: ProjPostTransView: costPrice
Form ProjTable.lookup Reference
Form ProjWorkBreakdownStructure
Form WHSPack.updateSummaryFields
FormletterJournalPost
FreeTextInvoiceDP.bankGroupIdName_CH
FreeTextInvoiceDP.bankZipCode_CH
FreeTextInvoiceDP.insertGiroInformation
FreeTextInvoiceDP.insertIntoFreeTextInvoiceHeaderFooterTmp
FreeTextInvoiceDP.insertIntoFreeTextInvoiceLocalizationTmp
FreeTextInvoiceDP.insertIntoFreeTextInvoiceTmp
HierarchyCreate_CRM.initHierarchy
HierarchyTemplateCopying_Proj.copyEstimates
InventDimGroupSetup.combineInventDimParms
InventLookupItemIdByDefaultOrder.initializeQuery
InventStorageDimMap.modifiedInventSiteFromParent
InventUpd_Physical.updatePhysicalReceiptTrans
JournalFormTable.initJournalTypeFromCaller

METHOD
LedgerJournalCheckPost.runInternal
Map SalesPurchLine.setPriceAgreement
Maps VendDocumentLineMap.setPurchaseInventReceiveNow
OffsetVoucherCust.getAutoSettlementQuery
ProjAdjustmentSelect.doTransCost
ProjAdjustmentSelect.doTransSale
ProjAdjustmentSelect.processProjEmplTrans
ProjAdjustmentSelect.validate
ProjAdjustmentSplit
ProjAdjustmentSplit.createNewTrans
ProjAdjustmentSplit.run
ProjAdjustmentUpdate.newPostAdjustment
ProjBegBalJournalTrans_CostSales.createProjTransPosting
ProjBegBalJournalTrans_Fee.createProjTransPosting
ProjBegBalJournalTrans_OnAcc.createProjTransPosting
projCostControl.progressUpdate
ProjEstimatesDataContract.setRevenueSalesPrice
ProjForecastBudget.forecastCopy
ProjForecastBudget.forecastDelete
ProjForecastPostItemFixedInvest.checkEnterCost
ProjForecastTransferFromWBS.transferToForecast
ProjFormLetter. mainOnServer
ProjFormLetter.printPreview
ProjInvoiceDP.insertIntoProjInvoiceLocalizationTmp
ProjInvoiceDP.insertIntoProjInvoiceTmp

METHOD
ProjInvoiceJournalCreate.creditMaxOk
ProjInvoiceJournalPost.initProposalUpdate
ProjLedger.newInventCost
ProjPlanVersionManager.copyActivityData
ProjPlanVersionsMananger.createDraftVersion
ProjProjectTransListPageInteraction.linkActive
Projtask.getCorrespondingTaskElementNumber
ProjValCheckTrans.validateMandatory
projWbsUpdateController.getNodesMapSortedByPath
PSAProjInvoiceDP.processLinesFromInvoiceJournal
psaProjQuotationSubmitSend.validateProjectDates
PSAQuotationsDP.insertPSAQuotationsTmp
PurchaseOrderResponseCreate.createPurchaseOrderResponseLines
PurchCancel.cancelMarkup
PurchCreateFromSalesOrder.preMatchIncludedLinesWithAgreements
PurchPackingSlipDP.setPurchPackingSlipDetailsTmp
PurchPackingSlipDP.setPurchPackingSlipHeaderTmp
PurchReceiptsListDP.setPurchReceiptsListDetailsTmp
PurchReceiptsListDP.setPurchReceiptsListHeaderTmp
PurchSummary.checkFormLetterId
ReqPlanCopy.insertLog
ResRollupActivityWriter::updateRollupTableWithLockedCapacityForActivityResource()
ResRollupAvailabilityWriter.updateRollupTableWithLockedCapacityForNamedResource()
SalesConfirmDP.setSalesConfirmDetailsTmp
SalesConfirmDP.setSalesConfirmHeaderTmp

METHOD
SalesInvoiceController::main
SalesInvoiceDP.bankGroupIdName_CH
SalesInvoiceDP.bankZipCode_CH
SalesInvoiceDP.insertIntoSalesInvoiceHeaderFooterTmp
SalesInvoiceDP.insertIntoSalesInvoiceLocalizationTmp
SalesInvoiceDP.insertIntoSalesInvoiceTmp
SalesPackingSlipDP.setSalesPackingSlipDetailsTmp
SalesPackingSlipDP.setSalesPackingSlipHeaderTmp
SalesQuotationLineType.validateDelete
SalesQuotationLineType.validateWrite
SalesQuotationTableForm.CreateABSFromTemplate
salesQuotationTransferToProject.initParameters
SalesTable.initFromCustTableMandatoryFields
SalesTableListPageInteraction.setButtonSell
smmActivityCreate.createActivity
smmActivityCreate.new
smmActivityParentLinkTablee.insert
SubledgerJournalizerProjectExtension.createLedgerUpdate
Table CustBankAccount.validatePreNote
Table InventItemGTIN.formatGTIN
Table PriceDiscAdmTrans.checkItemRelation
Table PriceDiscAdmTrans.checkproductDimensions
Table ProjCategory.lookupProjCategoryType
Table ProjTable.validateWriteServer
Table PSAActivityEstimates.checkUpdateQuotationLine

METHOD
Table PSAActivityEstimates.setSalesPriceFromCostPrice
Table PurchLine.setPriceDisc
Table PurchTable.internalTableIdToTableId_W
Table SalesLine.setPriceAgreement
Table Salesline.setPriceDisc
Table SalesQuotationLine.setPriceAgreement
Table SalesQuotationLine.setPriceDisc
Table SalesTable.setSalesOrderReleaseStatus
Table TmpCustVendTrans.createLineCreditLimit
Table TmpCustVendTrans.createLineCreditRemain
Table TmpCustVendTrans.createLineOrdered
Table TmpCustVendTrans.createLinePackingSlip
Table TmpCustVendTrans.createLineTotal
Table TmpCustVendTrans.insertTmpCustVendTransForCustBalance
Table TSTimeSheetLine.checkActivity
TSTimesheetLine::buildQuerySmmActivities
TsTimesheetPost.validatePost
VendProvisionalBalanceDP.insertVendProvisionalBalanceTmp()
VendTransQueryBuild::construct
VersioningPurchaseOrderResponse.archiveResponseLines
VersioningPurchaseOrderResponse.restoreLines
WhsCycleCountCreateLocation.run
WhsLoadReplenishment.calculateReplenishQty
WHSLoadTable::initPurchOriginDestination
WhsReplenishment.calculateReplenishQty

METHOD
WhsRFControlData.validateAndUpdateWorkClusterLPScan
WhsShipConfirm.tmsRouteConfirmation
WhsWarehouseRelease.buildReleaseQuery
WhsWarehouseRelease.createLoadLines
WHSWaveTable.createWaveTableFromTemplate
WHSWorkExecute.createAdjustmentWork
WHSWorkExecute.createCountingJournal
WHSWorkExecute.createInventLine
WHSWorkExecute.executeShortPick
WHSWorkExecute.shortPickAdjustOut
WHSWorkExecuteDisplayPOReceiving.createWork
WorkTimeTable.lookupTime

Inline delegates

Inline delegates are now available. The most common way to use inline delegates is to split the method into more granular methods and enable extensibility events in the smaller methods.

METHOD
AssetCopy.run
AssetPost.post
AssetSplit.createTrans
AssetSplit.run
BankDepositCreateCancelJour.createDepositCancelJournal
BankPaymCancel.createCancellingCustTrans
BankPaymCancel.reverseSettlement
BankPaymCancel.run
BankPositivePayExport.initPositivePayQuery
BankPrintTestCheque.createTestCheque

METHOD
BOMCalcItem.initListBOM
BOMCalcJob.runBOMCalculation
BOMRouteCopyJob.checkTo
BOMRouteCopyJob.main
BomRouteCopyJob::main
BomSearch.init
bomVersionActivate.run
CaseDetailForm.lookupParentCase
CaseDetailFormCreate.main
CaseUpdateStatus.changeStatus
CaseUpdateStatus_Close.updateStatus
ChequeDP.insertChequeTmp
Class SalesLineType.intercompanyMirror
Commission.run
CostControlPostingSourceDocumentLine.createCommittedCost
CostSheetPanel.build
CustAccountStatementExtController.insertCustAccountStatementExtTmp
CustAgingReportController.getReportName
CustAgingReportDP.insertCustAgingReportTmp
CustCollectionJourDP.collectionLetterTitle
CustCollectionJourDP.insertCustCollectionJourTmp
CustCollectionLetterCancel.main
CustCollectionLetterCreate.createJournal
CustCollectionLetterCreate.updateCreatedCollectionLetter
CustCollectionLetterPost.run

METHOD
CustCollectionLetterPost.updateFee
CustInterestCancel.run
CustInterestCreate.createJournal
CustInterestCreate.insertCustInterestTrans
CustInterestCreate.insertCustInterestTransLine
CustInterestPost.updateCustInterestTransVoucherRef
CustInterestPost.updateFee
CustInvoiceDP::insertCustInvoiceTmp
CustInvoiceSpecDP::insertIntoTempTable
CustNsf.createFeeJournalTrans
CustOutPaymControlController.insert
CustPostInvoice.createJournalHeader
CustPostInvoice::createJournalHeader
CustSettlementPriorityProcessing.createTempData
CustSettlementPriorityProcessing.markTransByCreditNoteOnBillingClasses
CustTransOpenPerDateDP.insertCustTransOpenPerDateTmp
CustVendCreatePaymJournal.checkBlocked
CustVendCreatePaymJournal.dialogAddInvoiceSelectionCriteriaFields
CustVendCreatePaymJournal.runPaymentProposalGenerationProcess
CustVendCreatePaymJournal_Vend.UpdateQuery
CustVendFindSettlements.findSettledSettlements
CustVendOpenTransBalances.initAccountNumCurrencies
CustVendOpenTransBalances.new
CustVendOpenTransManager.initFromCaller
CustVendOpenTransManager.updateOriginatorForMarkedTrans

METHOD
CustVendPaymProposal.createProposalLine
CustVendPaymProposalTransferToJournal.initLedgerJournalTransFromPaymLine
CustVendPaymProposalTransferToJournal.run
CustVendPaymProposalTransferToJournal.transferProposal
CustVendReversePosting.updateCustVendTrans
CustVendSettle.createSettlementForDebitOrCreditTrans
CustVendSumForPaym::Validate
CustVendVoucher.post
CustVoucher.createInvoiceJournal
DataEntityView FreeTextInvoiceEntity.insertFreeTextInvoiceLines
DataEntityView FreeTextInvoiceEntity.preTargetProcessSetBased
DimensionHierarchyHelper::getHierarchyTypeByAccountType
EcoResProductMasterManager.addProductDimensionValue
EcoResProductReleaseManager.createInventITable
EcoResProductReleaseManager.createInventItemSetupSupplyType
EcoResProductReleaseManager.setInventTableFields
EcoResProductTemplateManager.getBufferByDataSourceName
EcoResProductVariantManager.createProductVariant
Extend delegatestr(DirPartyPostalAddressFormHandler, defaultLocationRoles_delegate)
Form BankReconciliation: BankAccountReconcile::clicked
Form CustCreditLimitCreditPart.totalAgingByCompany
Form CustDirectDebitMandate.run
Form CustFormletterParameters.PrintMgMt.clicked
Form CustOpenTrans.init
Form CustOpenTrans.updateDesignStatic

METHOD
Form CustOpenTrans: Button UpdateNow::clicked
Form CustOpenTrans::doesCallerAllowEdit
Form CustTable: CustTable::write
Form EcoResProductCreate.writeMoreFields
Form EcoResProductVariantsPerCompany: InventDimCombination::write
Form HierarchyTemplateCopying_Proj.copyEstimates
Form InventDimParmFixed: InventDimParm::create
Form InventOnhandReserve: InventSum::reserveNow
Form InventOnhandReserve: InventTransOriginMovement::movementOnOrderUnit
Form InventOnhandReserve: InventTransOriginMovement::movementReservOrderedUnit
Form InventOnhandReserve: InventTransOriginMovement::movementReservPhysicalUnit
Form InventTransRegister: TmpInventTransWMS::setEnabled
Form LedgerJournalTransCustPaym.accountNumModifiedPost
Form LedgerJournalTransCustPaym: Button ButtonSettlement::clicked
Form LedgerJournalTransVendPaym: buttonPaymReconciliation::Clicked
Form LedgerJournalTransVendPaym: PaymReconciliationReject::Clicked
Form MarkupTrans.MarkupTrans_DS.active()
Form MCRSalesQuickQuote.init
Form MCRSalesQuickQuote.prepareSearch
Form MCRSalesQuickQuote.tmpFrmVirtualInventDimId
Form MRCSalesQuickQuote.createLines
Form PriceDiscActual::init
Form ProcCategoryHierarchyManagement.init
Form ProjAdjustment.init
Form ProjAdjustment.selectAdjRecords

METHOD
Form ProjCreditNoteSelect.canClose
Form ProjCreditNoteSelect.writeTmpFrmVirtual
Form ProjInvoiceProposalCreateLines.TransTypeSelectionCtrl.lookup
Form PurchTable: PurchTable::enableJournalButtons
Form SalesATP.SalesATP
Form SalesQuickQuote: InventDimCombination::getSetQuantyties
Form SalesQuickQuote: InventDimCombination::salesQty
Form SalesQuotationProjTable::SalesQuotationLine::ItemId::modified
Form SalesQuotationTable: SalesQuotationTable::write
Form SalesTable.modified
Form SalesTable.SalesTable_DS.linkActive
Form SalesTable.write
Form SalesTable: SalesLine::write
Form SalesTable: SalesTable::write
Form TMSRateRouteWorkbench.updateRoutes
Form VendEditInvoice: VendInvoiceInfoTable.write
Form WhsWorkTable.setFilter
FormLetterJournalPost.post
FormLetterService.run
Forms WHSLoadPlanningWorkbench.init
Forms WHSLoadPlanningWorkbench.restoreQuery
FreeTextInvoiceDP::insertIntoFreeTextInvoiceHeaderFooterTmp
From ProjTableCreate.init
HierarchyCreate.run
HierarchyTemplateCopyingDialog_proj.main

METHOD
InterCompanyPost.formLetterCollect
InventAgeDimDP.calcAllDim
InventAgeDimDP.insertInventAgeDimTmp
InventAgeDimDP.insertOrMergeInventAgeDimTmp
InventCountCreate.dialog
InventDimCtrl_Frm_Lookup.initDisplayOrderDataSource
InventDimPhysDP.processReport
InventDimViewContract
InventMovement.updateSerialNumIssue
InventMovement.updateSerialNumReceipt
InventMovement::updateLedgerPhysical
InventOnhandReserve.updateReserveNow
InventSumDateEngine.clearNotSelectedDimensions
InventTransferMulti.insert
InventUpd_Picked.updatePickMore
InventUpd_Reservation.updateReserveLess
InventUpdateOnhand.checkOnHand
InventValueReportContract
InventValueReportController
InventValueReportPopulateResource.initReportLines
JmgPostStandardSystem.postProjTime
JournalFormTable.designLookupJournalName
JournalFormTable.initAllOpenPostedFromCaller
LedgerBalancesBase.CalculateBalance
LedgerInAccountStatement.main

METHOD
LedgerJournalCheckPost.createReverseEntryJournalLine
LedgerJournalCheckPost.postJournal
LedgerJournalCheckPost.runInternal
LedgerJournalCheckPost::updateSystemBlockCheckedPostedJournal
LedgerJournalMultiPost.multiSelectPost
LedgerJournalTrans table.checkBankAccounts
LedgerJournalTransUpdateVend::postNewVendorVoucher
Map ProjTableWizardCtrl::insertDB
Map SalesPurchLine.calcPrice2LineAmount
Map SalesPurchLine.resetPriceAgreement
Map SalesPurchLine.setPriceAgreement
MarkupAllocation.sumValue
MCRInventSearch.executeSearch
MCROrderEventTable.Insert
McrPriceHistoryForm.insertPriceHistory
PmfFormCtrl.initPost
PriceDiscAdmCheckPost.checkForOverlapsAndGaps
PriceDiscAdmCopy.updateNow
ProdJournalCheckPostProd::checkTrans
ProdJournalCheckPostProd::postTransLedger
ProdJournalCheckPostProd::postVoucher
ProdJournalCheckPostRoute.updateProdRouteScheduling
ProdJournalCreateProd.createLines
ProdJournalCreateRoute.createLinesProdRoute
ProdJournalFormTable.datasourceExecuteQueryPre

METHOD
ProdMultiBOMCalc.run
ProdMultiCostEstimation.run
ProdMultiHistoricalCost.run
ProdMultiRelease.insert
ProdMultiRelease.run
ProdMultiReportFinished.main
ProdMultiReportFinished.run
ProdMultiSchedulingJob.run
ProdMultiSchedulingOperation.run
ProdMultiStartUp.run
ProdPurch.createPurchTable
prodTableChangeQtySched.performActionFromDefaultValues
prodTableChangeQtySched.performActionFromPrompt
ProdUpdCostEstimation.costEstimateOperations
ProdUpdCostEstimation.createPurchLine
ProdUpdReportFinished.run
ProdUpdReportFinished.updateBOMConsumption
ProdUpdStartUp.updateBOMConsumption
ProdUpdStartUp.updateRouteConsumption
ProjAdjustmentSelect.doTrans
ProjAdjustmentSelect.newQuery
ProjAdjustmentSelect.Run
ProjAdjustmentUpdate.checkTransChanged
ProjBudgetTransactionsManager.adjustBudget
ProjCopyForecastItem.copyToSalesLine

METHOD
ProjCOstControl.createActualCosts
ProjCOstControl.createActuals
ProjCostControl.createAverageForRemaining
ProjCostControl.createCommittedCosts
ProjCostControl.createForecastCosts
ProjCostControl.queryCommittedCosts
ProjCostControl.queryProjTransPosting
ProjCostControl.run
ProjCostControl.validate
ProjForecastBudgetCopy.do_cost
ProjForecastBudgetCopy.do_empl
ProjForecastBudgetCopy.do_OnaCC
ProjForecastBudgetCopy.do_Revenue
ProjForecastBudgetCopy.do_Sales
ProjForecastBudgetCopy.initQuery
ProjForecastBudgetCopy.validate
ProjForecastBudgetDelete.initQuery
ProjForecastTransferFromWbs. transferItemToForecast
ProjFundingEngine.allocate
ProjFundingEngine.isAmountWithinFundingLimits
ProjFundingEngine.updateFundingLimits
ProjInvoiceChooseNormal.dialog
ProjInvoiceJournalCreate.initTotals
ProjInvoiceJournalPost.createProjInvoiceCost
ProjInvoiceJournalPost.createProjInvoiceEmpl

METHOD
ProjInvoiceJournalPost.createProjInvoiceItem
ProjInvoiceJournalPost.createProjInvoiceOnAcc
ProjInvoiceJournalPost.createProjInvoiceRevenue
ProjInvoiceJournalPost.postCustVend
ProjInvoiceProposalCreateLines.runSalesLineQuery
ProjInvoiceProposalCreateLines.runTransactions
ProjInvoiceProposalInsertLines.run
ProjInvoiceProposalNormalPeriodic.createParameters
ProjInvoiceProposalPeriodic.dialog
ProjJournalCheckPost.processHourJournalResourceRateCost
ProjLedger.initFromProjectPostingTransaction
ProjLedgerUpdate.insert
ProjPlanVersionsManager.copyTasks
ProjProposalTotals.calc
ProjSplitBill.buildRuleQR
ProjSplitBill.split
ProjStatisticCalc.mapPSAEntityToTmpProjStatistic
ProjValCheckTrans.setVariablesFromBuffer
PsaCustomerRetention.createFeeTransaction
PsaGenerateQuotationLines.createSalesQuotationLines
PsaProjInvoiceDP::insertPSAProjInvoiceHeaderTmp
PsaProjInvoiceDP::insertPSAProjInvoiceTmp
PSARetentionRelease.insertLineRecords
PurchAgreementGenerateReleaseOrder.check
PurchAutoCreate_RFQ.createPurchLine

METHOD
PurchAutoCreate_Sales.createLine
PurchCancel.cancelMarkup
PurchCancel.run
PurchCopying.copyLine
PurchCreateFromSalesOrder.mcrDropChipCreateTmpFrmVirtual
PurchCreateFromSalesOrder.preMatchIncludedLinesWithAgreements
PurchFormLetter.PrePromptInit
PurchformLetter::Main
PurchFormLetter::MainOnServer
PurchFormletterParmData.createParmTable
PurchFormletterParmDataInvoice.chooseLinesFromPurchSelectLinesManager
PurchLineType.intercompanyMirror
PurchLineType_Project.initFromInventTable
PurchLineType_WithMultipleDeliveries.recalculateDeliveryScheduleOrderLine
PurchPackingSlipDP::setPurchPackingSlipDetailsTmp
PurchPackingSlipDP::setPurchPackingSlipHeaderTmp
PurchPurchaseOrderDP.createData
PurchPurchaseOrderDP.initializePurchPurchaseOrderHeader
PurchPurchaseOrderDP.processReport
PurchPurchaseOrderDP.setPurchPurchaseOrderDetails
PurchPurchaseOrderDP::setPurchPurchaseOrderDetails
PurchPurchaseOrderDP::setPurchPurchaseOrderHeader
PurchReceiptsListDP::setPurchReceiptsListDetailsTmp
PurchReceiptsListDP::setPurchReceiptsListHeaderTmp
PurchReqTable2LineField.lineUpdateDescription

METHOD
PurchRFQCaseAutoCreate.newAutoCreate
PurchRFQCompare.BuildReplyLineList
PurchRFQSendDP::processReport
PurchRFQSendJournalCreate.createOrUpdateRFQ
PurchTable2LineField.getFieldDescription
PurchTableType.intercompanyMirror
ReqActionApplyPurchaseOrder.applyActionToReferencedOrder
ReqBOMCreate.createBOM
ReqCalc.mcrInsertItemContinuitySales
ReqCalcScheduleItemTable.run
ReqSetupDim.setReqItemTableGrouped
ReqSupplyDemandScheduleModel.executeQuery
ReqSupplyDemandScheduleModel.insertPeriodValue
ReqTransPoMarkChangeToRFQ.change2RFQ
ReqTransPoMarkFirm.create
ReqTransPoMarkFirm.createInventTransferLine
ReqTransPoMarkFirm.createPurchLine
ReqTransPoMarkFirm.createPurchTable
ReqTransPoMarkFirm.firmSelectedPlannedOrders
RouteCopyToProd.copyTo
SalesAgreementGenerateReleaseOrder.check
SalesAgreementGenerateReleaseOrder.main
SalesAutoCreate_ReleaseFromAgreement.createSalesLine
SalesAutoCreate_ReleaseFromAgreement.createSalesTable
SalesAutoCreate_ReleaseOrder.createSalesTable

METHOD
SalesConfirmDP.setSalesConfirmDetailsTmp
SalesConfirmDP.setSalesConfirmHeaderTmp
SalesConfirmDP::setSalesConfirmDetailsTmp
SalesConfirmDP::setSalesConfirmHeaderTmp
SalesCopying.copy
SalesCopying.copyHeader
SalesCopying.deleteLines
SalesCopying_CreditNote.copy
SalesCopying_CreditNote.copyHeader
SalesFormLetter.mainOnServer
SalesFormLetter.reselect
SalesFormletterParmData.createParmLine
SalesFormletterParmData::createParmTable
SalesInvoiceController::outputReport
SalesInvoiceDP.useExistingReportData
SalesInvoiceDP::insertIntoSalesInvoiceHeaderFooterTmp
SalesInvoiceDP::insertIntoSalesInvoiceTmp
SalesLineExplodeBOM.explode
SalesLineType.canPickingListBeRegistered
SalesLineType.delete
SalesLineType.initDimensionsSpecificDefaulting
SalesLineType.initFromSalesLine
SalesLineType.interCompanyMirror
SalesLineType.setSalesStatus
SalesLineType.validateField field ShippingDateRequested and ShippingDateConfirmed

METHOD
SalesPackingSlipDP.printDimHistory
SalesPackingSlipDP::setSalesPackingSlipDetailsTmp
SalesPackingSlipDP::setSalesPackingSlipHeaderTmp
SalesPurchTableToLineUpdate.update
SalesQuantity_PackingSlip.calcQtySales
SalesQuantity_PickingList.calcQtySales
SalesQuotationConfirmationDP::setSalesQuotationDetailsTmp
SalesQuotationConfirmationDP::setSalesQuotationHeaderTmp
SalesQuotationCopying.buildTreeControl
SalesQuotationCopying.Copy
SalesQuotationDP::setSalesQuotationDetailsTmp
SalesQuotationDP::setSalesQuotationHeaderTmp
SalesQuotationEditLinesForm.mainOnServer
SalesQuotationEditLinesForm_Proj_Confirm.queryBuildSalesQuotationTable
SalesQuotationEditLinesForm_Proj_Send.queryBuildSalesQuotationTable
SalesQuotationEditLinesForm_Sales_Confir.updateNow
SalesQuotationEditLinesForm_Sales_Confirm.createSalesLine
SalesQuotationEditLinesForm_Sales_Send.checkLines
SalesQuotationJumpRef.main
SalesQuotationLineType.initFromSalesQuotationLine
SalesQuotationLineType_Proj.validateWrite
SalesQuotationProjLinkWizard.transferForecastToProject
SalesQuotationProjLinkWizard.transferItemReq
SalesQuotationTransferToProject.transferItemsToForecast
SalesQuotationTransferToProject.transferItemsToItemReq

METHOD
SalesQuotationUpdate.getCallerModuleFromParm
SalesTableForm.initValues
SalesTableForm_DeliverySchedule.updateSalesLineTable
SalesTableType.intercompanyMirror
SalesTableType.update
SalesTableType.validateDelete
smmSalesCustItemStatisticsDP::processReport
Table CaseDetailBase.validateWrite
Table CustCollectionLetterJour.updateCollectionLetterCodeCustTrans()
Table EcoResProductTranslation.queryAddCompanyLanguage
Table InventLocation::lookupBySiteIdAllTypes
Table InventPosting.accountGroup
Table InventPosting.accountItemLedgerDimension
Table InventPosting.deleteFromCust
Table InventPosting.deleteFromVend
Table InventTable.defaultProductDescription
Table InventTable.defaultProductName
Table InventTable.lookupBOMId
Table InventTrans.updateMarkReqTransCov
Table PaymTerm.due
Table ProdBOM.updateStartUp
Table ProdBOM.updateSubPurch
Table ProdJournalBOM.insertJournalCreate
Table ProdJournalBOM.lookupTransId
Table ProdTable.validateRouteId

METHOD
Table ProjBegBalJournalTrans_CostSales.postProjTransactionCost
Table ProjBegBalJournalTrans_CostSales.postProjTransactionHour
Table ProjBegBalJournalTrans_CostSales.postProjTransactionItem
Table ProjBegBalJournalTrans_Fee.postProjTransaction
Table ProjBegBalJournalTrans_OnAcc.postProjTransactionCost
Table PurchLine.initBarCode
Table PurchLine.priceDateDelegate
Table PurchLine.setPriceDisc
Table PurchTable.updateFromPurchReqLineMap
Table ReqPO.updateBOMRoute
Table ReqTrans.bulkInitFromInventTransOrigin
Table RouteOpr.validateFieldValue
Table RouteVersion.checkExistInventSiteId
Table SalesLine.checkPriceDate
Table Salesline.convertCurrencyCode
Table SalesLine.convertToDeliverySchedule
Table SalesLine.createFromTmpFrmVirtualIL
Table SalesLine.createReplacement
Table SalesLine.createSalesLine
Table SalesLine.expandBOM
Table Salesline.modifyInventDimSet
Table SalesLine.priceDateDelegate
Table salesLine.setPriceAgreement
Table Salesline.splitReturnLine
Table SalesLine::createFromSalesQuotationLine

METHOD
Table SalesQuotationLine.createFromTmpFrmVirtual
Table SalesQuotationLine.modifiedField
Table SalesQuotationLine.modifyInventDim
Table SalesQuotationTable.copyAddressToLine
Table SalesQuotationTable.lookupTemplateName
Table SalesTable.copyAddressToLine
Table SalesTable.copyRMALines
Table SalesTable.copyThirdPartyBillingAddressToLine
Table SalesTable.existingJournals
Table SalesTable.initFromCustTableIL
Table SalesTable.initFromProjTable
Table SalesTable.initFromSalesQuotationTable
Table SalesTable.unlinkAgreement
Table WHSAccountItemStatusDefault.checkModuleAccountNum
Table WHSLoadLine.delete
Table WHSLoadLine.updateReleaseQty
Table WhsLoadLine.validateQty
Table WHSLoadTable.assignOriginInfo
Table WHSProdTable.pickMore
Table WhsWorkTabke.lockUnLockWork
Table WMSBillOfLading.constructFromInvoice
Table WMSBillOfLading.constructFromPackingSlip
Table WMSBillOfLading.constructFromShipment
Table WMSOrderTrans.loopWMSOrderTransMulti
Table WrkCtrActivityRequirementSet.copyRequirements

METHOD
Table WrkCtrActivityRequirementSet.schedulingProperties
Tables SalesLine/Methods/setPriceDisc
TamDeductionUpdate_Deny.update
TmsProcessXML_Container.readShipContainer
TmsProcessXML_Shipment.readShipContainer
TradeInterCompany.insertInterCompanyInventDim
TradeInterCompanyConv.axPurchItemId
TradeInterCompanyConv.axSalesItemId
TransactionReversal_Asset.reversalBook
TransactionReversal_Cust::reversal
TransactionReversal_CustVend.createCustVendTrans
VendInvoiceDocumentDP::insertVendInvoiceDocumentTmp
VendInvoiceTableToLineUpdate.convertPurchTableFieldToVendInvoice
VendorInvoiceLineSourceDocLineItem.calculateSourceDocumentAmountMap
VersioningDocument.change
VersioningPurchaseOrder.createChangeRequest
WhsCycleCountCreateThreshold.processCycleCountThresholdItem
WHSInventOnHandReserve.changeReservation
WHSLaborStandards.findLaborStandardByItem
WHSLoadLine.update
WHSLoadTable.tmsLoadConfirmation
WHSLocationBuild
WHSLocationDirective.findLocation
WHSLocationDirective.findPickPutLocation
WHSLocationDirectiveActionQuery.modifyPickLocDirActionQuery

METHOD
WHSPool.pickFromWorkCenter
WHSPostEngineBase.prodCreateWork
WHSPostEngineBase.prodPickQty
WhsRfControlData.getClusterPickQty
WHSRFControlData.processControl
WhsShipConfirm.canShipConfirm
WHSShipConfirm.createInventTransferParmLineFromContainerTable
WHSShipConfirm.runTransferShip
WhsShipConfirn.validateAllAllowedForOverOrUnderdeliveryWorkQtyHasBeenPicked
WHSSplitWork.splitWork
WHSWorkClusterTable.cleanupCluster
WHSWorkCreateProdPut.insertProdParmforCoByProduct
WHSWorkCreateProdPut.insertProdParmforProdItem
WhsWorkExecute.getFirstOpenLineSystemDirected
WHSWorkExecute.overPickByItem
WHSWorkExecute.putAwayToLocation
WHSWorkExecute.scanLicensePlate
WHSWorkExecuteDisplay.buildPORecTrackingDimensions
WhsWorkExecuteDisplayCycleCount.findOrCreateCycleCountWorkLines
WhsWorkExecuteDisplayLPReceiving.displayForm
WHSWorkExecuteDisplayMixedLPReceiving.displayForm
WHSWorkLine.cancelLineMultiPick
WHSWorkLine.cancelLinePartial
WMSArrivalCreateJournal.createWMSJournalTrans
WMSArrivalCreateJournal.createWMSJournalTransFromTmp

METHOD
WMSArrivalOverviewGeneration.buildReturnOrderFromSalesLine
WMSJournalCheckPostReception.checkReference
WMSJournalTransUpdateSerialId.dialog
WmsPickingList_OrderPickDP::insertIntoTempTable
WrkCrtScheduler.writeJobCapacityReservations
WrkCtrApplicableResourceQuery.query
WrkCtrlScheduler_Prod.loadJobsDetail
WrkCtrlScheduler_Prod.saveOrder
WrkCtrlScheduler_Proj.saveOrder
WrkCtrlScheduler_Proj.writeJobData
WrkCtrReservedSum.find
WrkCtrScheduler.computeJobTimes
WrkCtrScheduler.insertWrkCtrCapResUsingInsertList
WrkCtrScheduler.writeJobCapacityReservations
WrkCtrScheduler.writeJobData

SQL operations made extensible

Application code with embedded SQL statements cannot be modified through extensions. Changes have been made to the standard application to enable extensibility in the methods listed in the following table. This has commonly been enabled by transforming embedded SQL statements into query objects that support extending how SQL statements are built in these methods.

METHOD
CustCollectionLetterCreate.updateAllExisting
CustCollectionLetterCreate.updateExisting
CustProvisionalBalanceDP.insertCustProvisionalBalanceTmp
CustProvisionalBalanceDP.processReport
CustSettlementPriorityProcessing.createTempData
DirPartyTable.getLocationFromRole

METHOD
InventQualityOrderTable.createInventQualityOrderLines
InventTransIdSum::calcSum
InventTransIdSum_InventLocation::calcSum
InventTransReference::setRefTrans
Markup.insertMarkupTrans
Markup.mcrCopyForReturn
PriceDisc.findDisc
PriceDisc.findPriceAgreement
PurchFormLetterParmDataInvoice.createLineProject
ReqPlanCopy.copyReqTransAndReqTransCov
ReqPlanCopy.copyReqTransKeep
ReqPlanCopy.copyWrkCtrCapRes
ReqPlanCopy.copyWrkCtrCapResForReqPO
SalesCopying.deleteLines
SalesLineType::deliveredInTotal
SalesTableType.parmPickingListRegistrationEnumerable
SalesTableType_Sales.canPickingListBeUpdated
SalesUpdateRemain.canclRemainderOnOpenSalesLines
SubledgerJournalizer.createSummaryFromJournalAccountEntry
SubledgerJournalizer.loadAccountingDistributionTmpJournalize
SubledgerJournalizer.loadFinalizeSubledgerJournalTmpDetail
SubledgerJournalizer.loadStandardSubledgerLedgerJournalTmpDetail
SubledgerJournalizer.loadSubledgerJourTmpDetailWithRelieving
SubledgerJournalizer.recordSubledgerJourAccEntriesForRounding
SubledgerJournalizer.summarizeJourAccountEntryDetailForRound

METHOD
SubledgerJournalTransferCommand.insertGeneralJournalAccountEntryRelatedDetail
TmsProcessXML_Base.writeShipDeliveryAccessorials
VendProvisionalBalanceDP.insertVendProvisionalBalanceTmp
VersioningPurchaseOrder.archivePurchLine
WhsLoadPostEngineBase::createShipments
WHSPackForm.getShipmentId
WHSPostEngineBase.executeWaveSteps
WhsPostPackingSlip.preparePackingSlipPosting
WhsWarehouseRelease.createShipments
WhsWarehouseRelease::createOrConsolidateShipmentForSalesOrder

Maps enabled for extensibility

New patterns have been introduced for maps implementation that will allow you to add field and methods by extensions. Details on how this is done is available in the documentation both with maps that are used as interfaces and for versioning implementations.

The following table lists the maps and related tables where changes have been applied for enabling extensibility.

MAPS AND TABLES
Map CustVendInvoiceTrans
Map SalesPurchLine extensions or inheritance
Map SalesPurchTable
Map VendDocumentLineMap
Table PurchLine mappings
Table PurchLineHistory mappings

Inventory dimensions

This release introduces a new model for adding inventory dimensions. In previous releases it was not practical to support customization for new inventory dimensions if that required extending every SQL statement that included inventory dimensions. Instead, we have added 10 inventory dimensions without any specific designated usage. Partner solutions will code through indirection models that hold their code, and other models are made for individual implementation projects that deploy one or more of the prefabricated inventory dimensions toward use in a partner solution. Documentation will be available on how to implement with

inventory dimensions under this model, and release of a sample app with a Flavor dimension will help you learn about the new model. The new inventory dimension can be freely deployed and used as either product dimensions or tracking dimensions.

The changes have led to changing multiple places across the application, including what is shown in the following list.

CHANGE
Extensible Product Dimensions
Form WHSInventOnHandReserve.updateInventDimFixedControls
InterCompanyInventDim: Condition with throw
InventDimFieldsMap - Added field
Inventory dimension InventUpdateOnhand::checkOnHand
Inventory Dimensions - Table InventDim.create
InventTransferUpdShip.populateIssueReceiptDimensions
Map InventInventoryDimensionEntityFieldsMapping::resolveInventDim()
Rename InventDimFieldsMap::getFieldIdForDimensionOnMappedTable to inventoryDimensionFieldIdOnMappedTable()
Table BOMConsistOfTmp mappings
Table BOMPartOfTmp mappings
Table EcoResTrackingDimensionGroup.isDimFieldTrackingDimension
Table InterCompanyInventDim mappings
Table InventAgeGroupDimTmp mappings
Table InventCheckRecieptCostPricePcsTmp mappings
Table InventCostTmpTransBreakdown mappings
Table InventCountStatisticsTmp mappings
Table InventDim mappings
Table InventOnhandTmp mappings
Table InventPhysicalPerWarehouseTransTmp_IT mappings
Table InventPriceOverviewTmp mappings
Table InventSumCriticalTmp mappings

CHANGE
Table InventSumDateTransReport mappings
Table InventSumDeltaDim mappings
Table InventTable mappings
Table InventTransferOrderOverviewTmp mappings
Table InventValueReportTmpLine mappings
Table ProdPickList mappings
Table SalesInvoiceTmp mappings
Table WHSPurchLine.registerPurchaseLine
Table WHSTmpCompleteWorkLine.lookupBatch
Table WHSTmpCompleteWorkLine.lookupTargetLicensePlateId
Table WMSCheckABCZonesTmp mappings
Table WMSPickingList_OrderPickTmp mappings
Table WMSPickingListReportTmp mappings

Metadata changes to enable extensibility

The following table lists changes made for enabling extensibility for specific metadata on these objects. These changes vary from instance to instance, you can consult the specific implementation to review the changes.

CHANGE
CountryRegionCodes property
CustCustomerEntity
EcoResProductCategoryAssignmentEntity made public
Form AssetSplit : FormControls
Form CustCollections.Cases
Form CustGroup
Form LedgerJournalTransCustPaym - menu item button auto declaration
Form LedgerJournalTransVendPaym - menu item button auto declaration
Table DimensionAttributeValueSetItem

CHANGE
Table EcoResReleasedProductCreationStaging missing ReplacementKey like other staging tables.
Tables SubledgerJournalAccountEntry(Tmp...)
View SubLedgerJournalAccountEntryView

Other changes

The following table lists additional changes that have been made for extensibility.

CHANGE
CustCollectionLetterCreate
CustCollectionLetterPost
Extensibility approach for number of decimal places for currency
Extensible edt decimal places: AssetDepreciationAmountUnit
Form Extension - DirPartyTable - registerOverrideMethod jumpRef
Form ProjCategoryLookup
Method signature changed: InventPostingSetupCache
Method signature changed: Table ProjCostPriceExpense.find
Method signature changed: Table ProjCostPriceExpense.findCostPrice
Method signature changed: Table ProjCostSalesPrice.find
Method signature changed: Table ProjCostSalesPrice.findCostSalesPrice
Method signature changed: Table ProjHourCostPrice.Find
Method signature changed: Table ProjHourCostPrice.FindCostPrice
Method signature changed: Table ProjHourSalesPrice.find
Method signature changed: Table ProjHourSalesPrice.findHourSalesPrice
New Quantity EDT added to ApplicationCommon
Other: New base enum for Price & Discount framework
Set Alternative Key = Yes to enable reference group lookups
Use of interface EcoResIProductCrossTableData

Bugs

The following table lists changes that were requested for extensibility but were acknowledged as bugs and fixed in the standard application.

CHANGE
class CaseUpdateStatus_Close, method changeStatus
Incorrect relation on CustCollectionLetterJour
Other: Bug fix on CompanyHelper.testCreateParameter
Table CustCollectionLetterJour - class cancelCollectionLetterCodeCustTrans

Extensibility changes in Finance and Operations, Enterprise edition (July 2017)

2/18/2021 • 9 minutes to read • Edit Online

This is a list of extensibility features that were implemented in the Dynamics 365 for Finance and Operations, Enterprise edition (July 2017). This version was released in July 2017 and has a build number of 7.2.11792.56024. For more information about the schedule of changes that support extensibility, see Application extensibility roadmap.

Soft-sealed application models

The following application middle-tier models were soft-sealed in this release. Overlayered code in these models will generate warnings on compilation.

CATEGORY	MODEL
Application Frameworks	CaseManagement
Application Frameworks	Dimensions
Application Frameworks	Directory
Application Frameworks	Organization
Application Frameworks	Currency
Application Frameworks	ApplicationCommon
HCM Core Models	3817938
Tax Models	Tax
Tax Models	Tax Books
Tax Models	Tax Books Application Suite Integration
Tax Models	Tax Engine Application Suite Integration
Tax Models	Tax Engine Configuration
Tax Models	Tax Engine Interface
Tax Models	Tax Engine Runtime Generation
Tax Models	TaxEngine
Source Document Models	SourceDocumentation

CATEGORY	MODEL
Source Document Models	SourceDocumentationTypes
Ledger Models	GeneralLedger
Ledger Models	Ledger
Ledger Models	Subledger
Middle Tier SCM Models	CostAccounting
Middle Tier SCM Models	CostAccountingAX
Middle Tier SCM Models	SCMControls
Middle Tier SCM Models	SCMMobile
Middle Tier SCM Models	UnitOfMeasure
Middle Tier SCM Models	WMSAdvancedMigration
Middle Tier SCM Models	InventoryDimensionConversion
Workspaces	ApplicationWorkspaces
Finance	Fiscalbooks
Management tools	DataUpgrade

Hard-sealed application models

The following application middle-tier models were hard-sealed in this release. Overlayered code in these models will generate errors on compilation.

CATEGORY	MODEL
Accounts Payable	AccountsPayableMobile
Finance	FinancialReportingEntityStore
Tools	PerformanceTool
Expenses	ExpenseMobile
GER	ElectronicReporting
GER	ElectronicReportingCore
GER	Electronic Reporting Application Suite Integration

Enumerations that are now extensible

The following changes were made to support extending enumerations:

- Many enumerations in the standard application have been made extensible. An enumeration is made extensible by setting two properties on the enumeration. The **IsExtensible** property is set to **Yes**, and the **UseEnumValue** property is set to **No**.
- Some enumerations represent state. New façade methods have been added to help enable adding enumeration values by extension. For information about how to extend an enumeration, see Add values to enums through extension.
- Some application code that uses enumerations was changed to support extensibility. Common changes include:
  - Removing **throw** exception statements in the default case of a switch to allow post-event subscription.
  - Adding **SysExtension** support for extension.
  - Adding explicit delegates.

ENUMERATION
AgreementState
AssetAccountType
AssetTransTypeJournal
BankAccountType
BankFormat
BarcodeContentType
BarcodeCoverPageEntityType
BarcodeType
BOMCalcCostingVersionUpdate
BOMCalcCostPriceUsed
BOMCalcSalesPriceUsed
BOMCalcType
BOMCheckLevel
BOMCopyContext
BOMCopyMethod
BOMCopyType
BOMCostCalculationMethod

ENUMERATION
BOMExplode
BOMRouteCopyDataType
BOMVersionFilter
BudgetReservation_BusinessEvent_PSN
BudgetReservation_SourceDocument_PSN
BudgetReservation_SourceDocumentLine_PSN
CatCallMethod
CatContentType
CatImportStatus
CatMaintenanceRequestWfStatus
CatProcurementErrorCode
CatPurchaseStatus
CatUserReviewApprovalStatus
CatVendorCatalogFileUploadType
CatVendorCatalogTemplateCategory
CatVendorCategoryHierarchyType
CatVendorConfigurationForImport
CatVendorLegalEntityStatus
CatVendorSiteType
ConsignmentReplenishmentOrderLineStatus
ConsignmentReplenishmentOrderStatus
CostBreakdown
CostCalculationCompareProductType
CostCalculationRole
CostCalculationState

ENUMERATION
CostCalculationSurchargeSubtype
CostingActivationType
CostingVersionCompareTo
CostingVersionPriceType
CostPriceBase
CostProfitSet
CostSalesPriceDisplay
CostSheetNodeListType
CostSheetPanelView
CostSheetProdFlowMode
CostStatementCacheAggregationAfter
CostWIPStatementCategory
CustPaymentValidate
CustSpecTransOverviewFormMode
CustTransRefType
CustVendPaymentStatus
CustVendTransportPointTypeTransfer
DlvScheduleMarkupConversionMode
EcoResAttributeModifier
EcoResCategoryAttributeModifier
EcoResCategoryChangeStatus
EcoResCategoryHierarchyModifier
EcoResCategoryNamedHierarchyRole
EcoResProductImageUsage
EcoResProductListPage

ENUMERATION
EcoResProductPerCompanyListPageType
EcoResProductTemplateType
EcoResReleaseProductToCompany
EcoResVariantConfigurationTechnologyType
ECPsalesOrdersViewType
EPCSSProductViewType
EUSalesTransMethod
FormLetterType
GanttCallerWrkCtr
GanttSetupType
GanttTimeUnit
GanttWrkCtrDisplayColumnsType
IntercompanyGoodsInTransitLineType
InterCompanyGoodsInTransitOrigin
InventAccountType
InventAccountTypeStdCostVariance
InventAdjustmentBy
InventAgingView
InventBatchJournalType
InventCostBundleState
InventCostCostDistribution
InventCostTransactionCategory
InventCostTransRefType
InventCountCode
InventItemCostingType

ENUMERATION
InventItemLookupDefaultTab
InventItemOrderSetupCallerType
InventItemOrderSetupType
InventItemPriceCompareLevel
InventItemPriceFilterType
InventItemPriceType
InventJournalOwnershipChangeLineCreateQueryStatusIssue
InventJournalType
InventLedgerConflictModule
InventLocationType
InventMovSubType
InventNonConformanceApproval
InventNonConformanceHistoryType
InventNonConformanceType
InventParameters
InventPhysicalReduction
InventRefType
InventReleaseOrderPickingType
InventReportDimHistoryLogType
InventStdCostConvItemStatus
InventStdCostPeriodType
InventSumFields
InventSupplyDlvModeSelectCust
InventSupplyDlvModeSelectSupply
InventSupplyLeadTimeSource

ENUMERATION
InventSupplyTmpLeadtimeType
InventTestActionOnFailure
InventTestBlockProcess
InventTestCorrectionPriority
InventTestCorrectionStatus
InventTestDocumentStatus
InventTestOrderStatusDisplay
InventTestOutcomeStatus
InventTestQtySpecification
InventTestQuarantineType
InventTestReferenceType
InventTestReport
InventTestType
InventTrackingDimNodeType
InventTrackingProductType
InventTrackingRegisterTransRegStatus
InventTransChildType
InventTransferRemainStatus
InventTransferStatus
InventTransferUpdateType
InventTransPickRegisterLineStatus
InventTransType
InventUpdType
InventValueReportLedgerAccountCategory
InventValueReportLedgerLineType

ENUMERATION
InventValueReportResourceType
ItemGroupLedgerDimensionGroup
ItemReservation
JmgAbsenceColumnLayout
JmgAbsenceMethodEnum
JmgAttendanceRegistrationType
JmgAttendanceReportType
JmgBarCodeType
JmgBreakDropEnum
JmgClockStyle
JmgControlType
JmgDaysTotalWorkflowStatus
JmgEmployeeSignInStatus
JmgFeedbackButtonFunction
JmgFeedbackStyle
JmgFieldName
JmgGetRegistrationTimeFrom
JmgGridAppearance
JmgJobTableSynchronizationMode
JmgJournalRegWorkflowStatus
JmgMark
JmgMessageType
JmgPayAdjustType
JmgPayEventsExportType
JmgPaySpecTypeEnum

ENUMERATION
JmgPaySpecTypeEnumPick
JmgPostAutomatically
JmgProdStatusUpdate
JmgProdStatusUpdateReportFinished
JmgProfileSpecTypeEnum
JmgProfileStartCodeBlankPrev
JmgProjStatusUpdate
JmgRegistrationTouchJobStatus
JmgSecondPresentationEnum
JmgShopFloorServiceStatus
JmgSignInButtonFunction
JmgStoppedCompletedStatus
JmgTermBaudeRate
JmgTermComPort
JmgTermDataBit
JmgTerminalInsertMode
JmgTermStopBit
KanbanBoardRefreshCycle
KanbanBoardType
KanbanCardAssignmentType
KanbanControlActionState
KanbanControlLegendFormat
KanbanControlSelectionChanged
KanbanDemandOriginType
KanbanJobPeggingType

ENUMERATION
KanbanJobPickingListLineType
KanbanLineEventType
KanbanMultiMode
KanbanPrintInstructions
KanbanProdBOMLineEventType
KanbanQuantityCalculationStatus
KanbanSalesLineEventType
KanbanStockReplenishmentEventType
LeanBOMLineReservationMethod
LeanCostingUnusedQtyType
LeanHandlingUnitEmptyPolicy
LeanInventoryControl
LeanPeggedEventType
LeanPlanJobReferenceTypes
LeanProductionFlowCostingStatus
LeanProductionFlowVisualizationViewMode
LeanProductTypes
LeanTaktStatus
LedgerPostingType
LedgerTransTxt
MarkupAllocateAfter
MarkupCategory
MCRBrokerContractStatus
MCRCustSearchType
MCRFullTextSearchType

ENUMERATION
MCRInstallPlanApplyMiscCharge
MCRItemListGenerationType
MCRPickingPrompt
MCRPickingSessionStatus
MCRPickingWaveStatus
MCRPriceHistoryType
MCRRoyaltyLineBreakType
MCRRoyaltyTakenFrom
MCRRoyaltyTransactionType
MCRRoyaltyUnitType
MCRRoyaltyUOMOption
MCRSalesOrderDetailStatus
ModuleInventCustVend
ModuleInventPurchSales
OriginalDocument
PaymDocumentType
PCExpressionEditorSymbolType
PCLookupMethod
PCNewSelectComponent
PCRequirement
PCTableConstraintType
PDSAdjustmentPrinciple
PdsBatchAttribToleranceAction
PdsBatchAttribUpdateType
PDSCalcElementBase

ENUMERATION
PDSCompensationPrincipleEnum
PDSElementTypeEnum
PDSIngredientTypeEnum
PdsMRCDocumentStatus
PdsMRCEffectiveDateBasis
PdsMRCEventModule
PdsMRCEventType
PdsMRCListType
PdsPaymtType
PDSPotencyAttribRecordingEnum
PdsRebateCalcDateType
PdsRebateTakenFrom
PdsRebateUOMOption
PdsSameLotError
pdsTMAJournalPosting
PdsUpdateBatchDate
PdsUpdateDispositionStatus_Quality
PlanActivityCreateRelationType
PlanActivityProductionFlowActivityType
PlanTypes
PmfCostAllocationMethod
PmfOrderType
PmfOrderTypeFilter
PmfProdType
PMFSeqCalendarPeriod

ENUMERATION
PriceBase
PriceDiscPurchasePromptSystemSource
PriceDiscSalesPromptSystemSource
PriceGroupType
PriceSalesPurch
PriceType
ProcCategoryAdministrationActivity
ProdBOMConsumpProposal
ProdBOMJournalQty
ProdBOMJournalSplit
ProdErrorCause
ProdGanttJobColorType
ProdGanttLoad
ProdGanttRouteColorType
ProdJournalCleanUpMode
ProdMode
ProdNotificationLevel
ProdParamInventDimLookup
ProdParmType
ProdRefLookUp
ProdRouteJobCurrentFormTabId
ProdSchedDirection
ProdScrapMethod
ProdStandardCostVariance
ProdStatusAll

ENUMERATION
ProdStatusType
ProductionTransType
ProdUpdateJour
ProdWHSReleasePolicy
ProdWIPType_NA
PurchaseOrderResponseAction
PurchaseType
PurchasingTransactionType
PurchCORReceivingMethod
PurchCORRejectStatus
PurchCovRef
PurchDlvAddr
PurchLineBackOrderViews
PurchLineDeliveryFulfillment
PurchLineDeliveryPrecision
PurchMatchingPolicyOverrideOption
PurchPrepayApplicationPolicy
PurchPriceDateType
PurchPurchaseOrderCreationMethod
PurchReApprovalPolicyRuleViewType
PurchReqAuthorizationSpecificReporting
PurchReqAutoCreatePurch
PurchReqCatalogAllNon
PurchReqConsolidationActiveStatus
PurchReqConsolidationPriority

ENUMERATION
PurchReqConsolidationStatus
PurchReqCreationStatus
PurchReqItemDescriptionTransfer
PurchReqItemFilterType
PurchReqOnBehalfReports
PurchReqOriginationAuthorizationView
PurchReqProcessingState
PurchReqQuestionnaireAggregateStatus
PurchReqQuestionnaireStatus
PurchReqReportSortOrder
PurchReqReportStatus
PurchReqReviewStatus
PurchReqRFQRequirement
PurchReqRFQType
PurchReqSaveChanges
PurchReqStatus
PurchReqType
PurchReqWorkflowState
PurchRFQQuestionnaireStatus
PurchRFQStatusVendor
PurchRFQType
PurchTableFormId
PurchTableListPage
PurchTableMode
PurchTotalsCachingMethod

ENUMERATION
PurchUpdate
PurchVendorPortalShowResponseType
QuotationType
ReqBOMRouteCreated
ReqCurrentDaySchedFrom
ReqDemPlanDataSourceType
ReqDemPlanDemandCategory
ReqDemPlanForecastAttributeType
ReqDemPlanForecastingStrategy
ReqDemPlanForecastType
ReqDisplayDelay
ReqForecastReducedBy
ReqGanttColorType
ReqGanttShow
ReqItemJournalType
ReqItemTableWizardPurpose
ReqMarkUpdate
ReqPeggingAssignmentType
ReqPeggingType
ReqPlannedOrderLeveling
ReqPlanType
ReqPOStatus
ReqQtyAmount
ReqRefType
ReqRefTypeShort

ENUMERATION
ReqRefTypeTrunc
ReqTraceMessageType
ReqTransFuturesActionPartType
ReturnCodeType
ReturnCycleTimeScope
ReturnReasonCodeDispExtended
ReturnReasonDispCode
ReturnUpdateAction
RouteFormula
RouteOprPriority
SalesBasketType
SalesBatch
SalesCheckForPickup
SalesCheckQtyCachKey
SalesDeliveryTimeState
SalesDocumentTimezonePreference
SalesPriceDateType
SalesPriceModelBasic
SalesPurchCopy
SalesPurchCycleAction
SalesPurchGroup
SalesPurchParmCleanUpMode
SalesQuotationFilter
SalesQuotationLinkToProject
SalesQuotationListPage

ENUMERATION
SalesQuotationPriceConversion
SalesQuotationPriceSimResult
SalesQuotationTypeListPage
SalesShipping
SalesSourcingOrigin
SalesStatus
SalesTableFormId
SalesTableListPage
SalesTableMode
SalesType
SalesUpdate
ShipCarrierDlvType
ShipCarrierFreightApplied
ShipCarrierMkUpFreight
SMAInvoiceProjectSelection
SMAItemSetupType
SMAProjectSelection
SMAReasonType
SMAServiceBOMChangeAction
SMAServiceFunctionType
SMAServiceLevelAgreementLogType
SMAServiceOrderActionType
SMAServiceOrderFilter
SMAServiceOrderOrigin
SMAServiceOrderProgress

ENUMERATION
SMAServiceTaskTitleOption
SMASubscriptionPeriodType
SMAWizardCreateType
smmAccountNumToCreate
smmActivityParentType
smmAppointmentNThInstance
smmBusinessRelationsListFilter
smmBusRelTypeSourceTable
smmCampaignBroadcastType
smmCampaignProjectJournalType
smmCampaignResponse
smmCampaignsListFilter
smmContactsListFilter
smmCreateOpportunityOptions
smmDisplayEMailInOutlook
smmDragDropObjectType
smmDupMethods
smmEMailSMS
smmEntityToCreate
smmFieldDelimiters
smmLeadsListFilter
smmLogType
smmOpportunitiesListFilter
smmOpportunityAssociation
smmOutlookContactDeleteAction

ENUMERATION
smmOutlookRecurrenceType
smmOutlookSyncPrinciple
smmOutlookUpdateAction
smmProjectNewExisting
smmQuotationAccountType
smmQuotationStatus
smmRecordDelimiters
smmSalesUnitMemberRelation
SmmSourceTypeList
smmSwotType
smmTransLogUpdateAction
smmUpdateOpportunityOptions
smmWarningError
SMAActiveAll
SMAAgreementFilter
SMAAgreementTableListPageType
TAMCustRebateApprovalStatus
TAMFundStatus
TAMFundType
TAMPromoCustomerType
TAMPromoMerchEvent
TAMPromoMgmtApprovalStatus
TAMPromotionDate
TAMPromotionMode
TAMRebateCustInclusive

ENUMERATION
TAMRebateLineBreakType
TAMRebateStatus
TAMRebateUnitType
TAMRebateUOMOption
TAMVendRebateApprovalStatus
TAMVendRebateCalcDateType
TAMVendRebateTakenFrom
TAMVendRebateTransactionType
TaxModuleType
TaxSourceType
TMSAccessorialAssignmentLevel
TMSAccessorialAssignmentTarget
TMSAccessorialDeliveryType
TMSAccessorialType
TMSAppointmentAlert
TMSDiscountResultType
TMSDiscountType
TMSFeeType
TMSFreightBillMatchStatus
TMSFreightBillReconcileType
TMSFwkErrorType
TMSHubPosition
TMSInvoiceAccountType
TMSLineType
TMSLoadBuildSessionState

ENUMERATION
TMSLoadTender
TMSLookupType
TMSNumberSequenceType
TMSOverrideLocationType
TMSRecurrenceDays
TMSRecurrenceType
TMSRecurrenceWeeks
TMSResponsibleForPayment
TMSRouteStatus
TMSSalesPurchTransfer
TMSTableRef
TMSTransportationType
TMSTransportRefType
TMSTransportTypeFilter
TMSUOM
TMSZoneType
TradeCurencyConversion
TradeLineDlvType
TradeNonStockedConversionChangeType
TradeNonStockedConversionIssue
TradeNonStockedConversionResolveUndo
TradePrintType
TradeTable2LineUpdate
VendNotificationCategorySelection
VendNotificationStatus

ENUMERATION
VendPackingSlipTransTimeStatus
VendRequestCompanyType
VendRequestOriginatedByType
VendRequestQuestionnairesCompleted
VendRequestRoleType
VendReviewRatingScore
VersioningAction
WHSAllowMaterialOverPick
WHSApplicableDemand
WHSAutoReleaseContainerAtContainerClose
WHSAutoReleaseOrderType
WHSBreakCluster
WHSContainerizationQueryType
WHSContainerPackingStrategy
WHSContainerTableFormViewType
WHSCrossDockFulfillmentStrategy
WHSCustJourType
WHSCycleCountPlanStatus
WHSDefaultDataField
WHSFilterModule
WHSHistoryEvent
WHSLoadPlanning
WHSLoadPostMethodsBase
WhsLoadReplenishment
WHSLoadTable

ENUMERATION
WHSLocDirStrategy
WHSLPAssignment
WHSLPWFilterType
WHSManifestAt
WHSManifestRequirementContainerGroup
WHSMenuItemDirectedBy
WHSMixedLPReceivingMode
WHSMixingLogicTables
WHSMobileAppPagePattern
WHSOriginType
WHSPickOldestBatch
WHSPostMethodBaseKanban
WHSPostMethodBaseKanbanOptional
WHSPostMethodBaseOptional
WHSPostMethodBaseProd
WHSPostMethodBaseProdOptional
WHSPostMethodsBase
WHSQtyPct
WHSReleaseQuantitySpecification
WHSReplenishmentDependentWorkBlockingPolicy
WHSReservationHierarchyLevelStrategyType
WHSReservationStatus
WHSUseFixedLocations
WHSWorkActivity
WHSWorkClusterStatus

ENUMERATION
WHSWorkCreationProcess
WHSWorkExceptionLogStatus
WHSWorkExecuteMode
WHSWorkListPageFilter
WHSWorkPrintOption
WHSWorkPutFlow
WHSWorkTransType
WHSWorkType
WHSWorkTypePickPut
WMSAutoAddStop
WMSFreightChargeTerms
WMSFreightCounted
WMSHandlingType
WMSLocationType
WMSPackageType
WMSPalletMovementProcessing
WMSPhysicalUpdateStatus
WMSReceiptStatus
WMSReservationMethod
WMSReservationMethodInternal
WMSShipmentStatus
WMSShipmentType
WMSSpaceUtilInconsistencyGroup
WMSSpaceUtilShowBy
WMSSpaceUtilStorageLoadUnitType

ENUMERATION
WMSStoreAreaType
WMSTrailerLoaded
WrkCtrActivityType
WrkCtrBulkResReqSearchType
WrkCtrCapacityType
WrkCtrCapRefType
WrkCtrCommitState
WrkCtrGroupWrkCtr
WrkCtrSchedulerCommand
WrkCtrSchedulerConstraintType
WrkCtrSchedulerLoggerMode
WrkCtrType
WrkCtrTypeFilter

These enumerations were removed, and not made extensible.

ENUMERATION REMOVED
BackorderLinesListPageMode
BackorderPurchLinesListPageMode
EcoResProductPerCompanyListPageType
ReturnTableListPageType
SMAAgreementTableListPageType

Foundation changes were made to improve support for extensible enumerations. The **SysPlugin** framework was enabled for enumerations where **IsExtensible** is set to **Yes**. Views were enabled with new name-based syntax for enumerations.

Data manipulation methods that do not raise DataEvents or missing insert, update, delete pre- and post-data events

As a general practice, you use data methods on tables to raise events that can be used for extending the application. The code base has not always followed this practice. For example, the **doInsert**, **doUpdate**, and **doDelete** data methods and certain table implementations did not make a call to **super()** in the data method.

The **insert**, **update**, and **delete** methods on the type classes have been refactored. Changes were made so that **super()** is called more consistently in data methods. These changes enable extensions to be added to these methods, so that pre- and post-events are now available for extension. The tables where the **insert**, **update**, and **delete** events were enabled for extension are listed in the following table.

TABLE
InventBlocking
InventTransferLine
Kanban
KanbanJob
KanbanJobPickingList
MCRRoyaltyVendTable
PdsRebateTable
PmfProdCoBy
ProdBOM
ProdRoute
ProdTable
PurchLine
PurchRFQCaseLine
PurchRFQCaseTable
PurchRFQLine
PurchTable
SalesLine
SalesQuotationLine
SalesQuotationTable
SalesTable
TAMVendRebateTable
WMSOrder
WMSOrderTrans

Exposing class members

Additional private members are now available for customization as a result of the changes to the access modifier or new parm methods. The chain of command platform feature enables extension class access to protected methods and members. For more information about chain of command, see Extensible X++: Chain of Command.

MEMBER
AssetPost.ledgerJournalTrans
Class DimensionDerivationRule.ledgerDimensionAllocationList
Class PurchInvoiceJournalCreate.purchTable
Class PurchTableType.purchTable
Class SalesInvoiceJournalPost.salesLine
Class SalesQuotationLineType
Class SalesQuotationTableType
Class VendorInvoiceLineSourceDocLineItem.purchLine
CustCreditLimit.balanceTotalsCalculated
CustCreditLimit_SalesTable.salesTable
Form LedgerJournalTransCustPaym.ledgerJournalEngine
PurchLineType.purchLine
PurchLineType.purchLine_orig
SalesLineType.salesLine
SalesLineType.salesLine_orig
SalesTableType.checkSalesQty
SalesTableType.SalesTable_orig
WHSControl.data
WHSLocationDirective.targetLicensePlateId

Construct methods with throw statements

Some **construct** methods were implemented with **throw** statements if there was a missing implementation for a given type. This doesn't work well with extensibility, so to mitigate this, **construct** methods were changed so that they do not throw exceptions. These methods are now to open for extensibility through class augmentation or by post-event subscription.

OBJECT
BarcodeEAN128.string()
CustCreditLimit.Construct
FormLetterReport
JournalStatic
MarkupAllocation
PurchTable2LineField
SalesCalcTax
SalesEditLinesForm
SalesFormLetter
SalesFormLetterContract.newFromPackedVersion
SalesFormletterParmData.newData
SalesQuantity
SalesQuotationEditLinesForm.construct
SalesSummaryFields
SalesTable2LineField
VendInvoiceTableToLineUpdate

Find methods with throw statements

Some **find** methods were implemented with **throw** statements if there was a missing implementation for a given type. This does not work well with extensibility, so to mitigate this, **find** methods were changed so that they do not throw exceptions. These methods are now to open for extensibility through class augmentation or by post-event subscription.

METHODS
JournalStatic.findJournalTableFromTrans
JournalStatic.findJournalTableId

Methods made hookable

Extensibility support has been extended for some methods that were not public and were not hookable. The following methods have been explicitly decorated with hookable behavior.

METHOD
Class CustVendReversePosting.updateCustVendTrans
Class JournalTableData.construct
Class PriceDisc.makeKey
Class PurchInvoiceJournalCreate.initJournalHeader
Class SalesInvoiceJournalPost.checkSourceLine
Class SalesInvoiceJournalPost.postCustVend
Form LedgerTransVoucher.addDynaLink
ReqCalc.deleteItemRequirement
ReqCalc.initTransFromInventTrans
ReqCalcForecastItemTable.deleteRequirement
Table TmpCustVendTrans.createLineCreditLimit
Table TmpCustVendTrans.createLineCreditRemain
Table TmpCustVendTrans.createLineOrdered
Table TmpCustVendTrans.createLinePackingSlip
WHSLocationDirective.addRangeByTransType
WhsWorkCreate.createWorkLine

Inline delegates

Inline delegates are now available. The most common way to use inline delegates is to split the method into more granular methods and enable extensibility events in the smaller methods.

METHOD
AxClass - ChequeDP - Method - insertChequeTmp
AxClass - VendInvoiceTableToLineUpdate - Method - convertPurchTableFieldToVendInvoice
class JournalStatic.findJournalTableFromTrans
class JournalStatic.findJournalTableId
Class WhsLocationDirective.findLocationMultiSKU
EcoResReleasedProductVariantEntity.insertEntityDataSource

METHOD
ForecastSales.ForecastSales_ds.updateForecastSalesFields
Form SalesTable - method updateDesign
ReqTransPoMarkFirm.firmSelectedPlannedOrders
SalesLineType.insert
SalesLineType.update
SalesTable2LineUpdatePrompt.dialog
table ExtCodeTable
table InventItemGroup.getGroupForAccountType
table InventItemGroup.ledgerDimensionDescription
table InventTestAssociationTable
Table LedgerJournalName - method validateWrite
Table PaymTerm - method due
TaxCalculation.newForSourceType
TaxCalculation.newForSourceTypeWithTaxUncommitted
WHSLoadLine.getOrderCommonFromLoadLine
WhsLocationDirective.findLocation
WHSRFControlData.populateData
WHSRFControLData.processControl
WHSRFMenuItemTable.getWHSWorkExecuteMode

Other changes

The following table lists additional changes that have been made for extensibility.

CHANGE
Add indirection for existing product dimensions
Class FormLetterParmDataOutputContract is not extensible
Create an instantiation strategy for the SysExtensionFramework that supports one or more arguments

CHANGE
Customization: TableField: Extension Model: Change EDT type of on a table field
CustVendOpenTransBalances - initAccountNumCurrencies() switch statement
CustVendOpenTransBalances - new() switch statement
CustVendOutPaym (Class) needs extensibility improvement
CustVendPaymReconciliationSetStatus (Class) needs extensibility improvement
CustVendSumForPaym (Class) needs extensibility improvement
Decouple AddressCountyId and AddressStateId EDTs from SysGroup
Document Management event handling needs improved extensibility support
Exchange rate provider framework requires custom built providers to be placed in the Currency Model
Extending GS-128
Extension model: Allow customizations on the CountryRegionCodes property.
Form extension - Extension of "extended" form elements with new controls are not working
InventDim: Condition with throw
InventDimRenameDimValue
Method overlayering - Class VendInvoiceTableToLineUpdate.convertPurchTableFieldToVendInvoice
Method overlayering: class Markup - method delete
Method overlayering: class McrPriceHistoryForm.insertPotentialTradeAgreements
Method overlayering: Class OffsetVoucher - method markTransaction
Method overlayering: Form LedgerJournalTransDimension.init
Method overlayering: Form LedgerTransVoucher - method init
Method overlayering: Table CustInvoiceTable - method validateWrite
Method signature changed: RetailCreateLinesFromProductsToAdd.parmCallerCommon
Method signature changed: WHSInvent.getCommonFromWorkTransType
Method signature changed: WHSPoolProdBom.movementBuffer
Missing construct method: class SMAServiceOrderTableButtonStateProvider

CHANGE
Number sequence scope extensibility needed
Runbase needs a way for class extensions to pack/unpack their members
String EDT size extension issues
Support opening Inventory on-hand form based on custom InventDim
SysExtension framework: SysExtensionIInstantiationStrategy and SysExtensionIAttribute are not compatible
Variations of EventHandlerResult are requested to ensure that delegates used in Request/response scenarios are more robust
WHS Mobile Framework: passes
WhsLocationDirectiveLine To/FromQty not extensible
WHSMobileApp Extensibility
WHSMobileAppAttachedImageDetails.removeLabelFromDimValue() is not generic enough about Product dimensions
WHSMobileAppAttachedImageDetails.removeLabelFromDimValue() is not generic enough about Product dimensions
WhsRFControlData.processControl must reference WhsControl.data instead of _data in the switch block

Intrusive customizations

2/18/2021 • 3 minutes to read • Edit Online

This topic defines the characteristics of an intrusive customization. Intrusive customizations are the major obstacle to keeping continuous upgrade costs close to zero. Some types of intrusive customizations can be prevented by tooling, whereas other types remain the responsibility of the author of the extension. The X++ compiler and Microsoft Visual Studio designers will prevent some types of intrusive customizations. However, a subset of intrusive customizations can't be detected by tooling but might still prevent continuous upgrades. Ultimately, the developer is responsible for avoiding intrusive customizations.

A customization that violates any of the following principles is intrusive.

NOTE

When extending other solutions you are expected to be responsible. This includes:

- Accepting the responsibility of your extension. You are introducing new behavior and therefore own the full responsibility of the change.
- Allowing other extensions to co-exist. Recognize that you are not the only consumer of an extension point. For example, only respond when needed.
- Only adding extensions that are coherent with the extension point. The longevity of a solution is defined by its resilience to change. Recognize that extension points may be exercised in more or fewer scenarios in future versions. For example, only add relevant validation logic to a validate() method.
- Avoiding dependencies on implementation details. Implementation details are likely to change in future versions. Make your solution resilient by avoiding dependencies on local variables, call-stacks and, calling sequences and avoid using reflection.

Don't change type definitions

Types are referenced by their definition. A change to a type's definition is a breaking change and requires that all references be updated. It's impossible to ensure that future references will be implemented correctly (for example, in the model that hosts the type). There are several implications:

- Don't change a method signature. The method signature includes the return type, the name (which includes casing), and the parameters (which include optional parameters).
- Don't change requirements for implementers of interfaces and table maps. For example, don't add a new method to an interface or a new field to a table map.
- Don't change requirements for classes that are derived from abstract classes. For example, don't add a new abstract method to a class.
- Don't reduce access modifiers for types or members. For example, don't change classes, tables, or methods from public to private.
- Don't change constraints that are defined on a table or a data entity. Constraints include allowing editing, mandatory constraints, uniqueness constraints, and referential constraints.

Don't break encapsulation

The author of a model must be able improve the product by remaining in control of encapsulated code and types. Model owners must be able to change and delete encapsulated code and types at will, without prior notice, and without risk of downstream impact on extensions and customizations. Encapsulation is broken if, for example, a private method is deleted. Here are some of the implications:

- Don't increase access modifiers for types or members. For example, don't change classes, tables, or methods from private to public.
- If something should be closed for modification, don't open it for customizing behavior. Extension capabilities must be designed as open for extensibility but closed for modification.

Be additive in nature

New behaviors are enabled through added functionality as part of extensions. Extension capabilities must be designed in and open for extensions, and they must support multiple extensions that exist side by side in the same installation. There are several implications:

- Don't overlayer. Overlayering replaces the default implementation and prevents multiple solutions from changing the same element.
- Don't significantly change the characteristics of the standard functionality. These characteristics include the user experience and performance. For example, performance must not be adversely affected if an extension is installed but isn't used.
- Don't unconditionally set results in **EventHandlerResult** classes, and don't unconditionally set return values in **XppPrePostArgs** classes.
- Don't replace existing behavior by default, unless the replacing logic conforms to the Liskov Substitution Principle.

> **NOTE**
>
> The author of the extension is responsible for not violating these principles.

Class extension model in X++

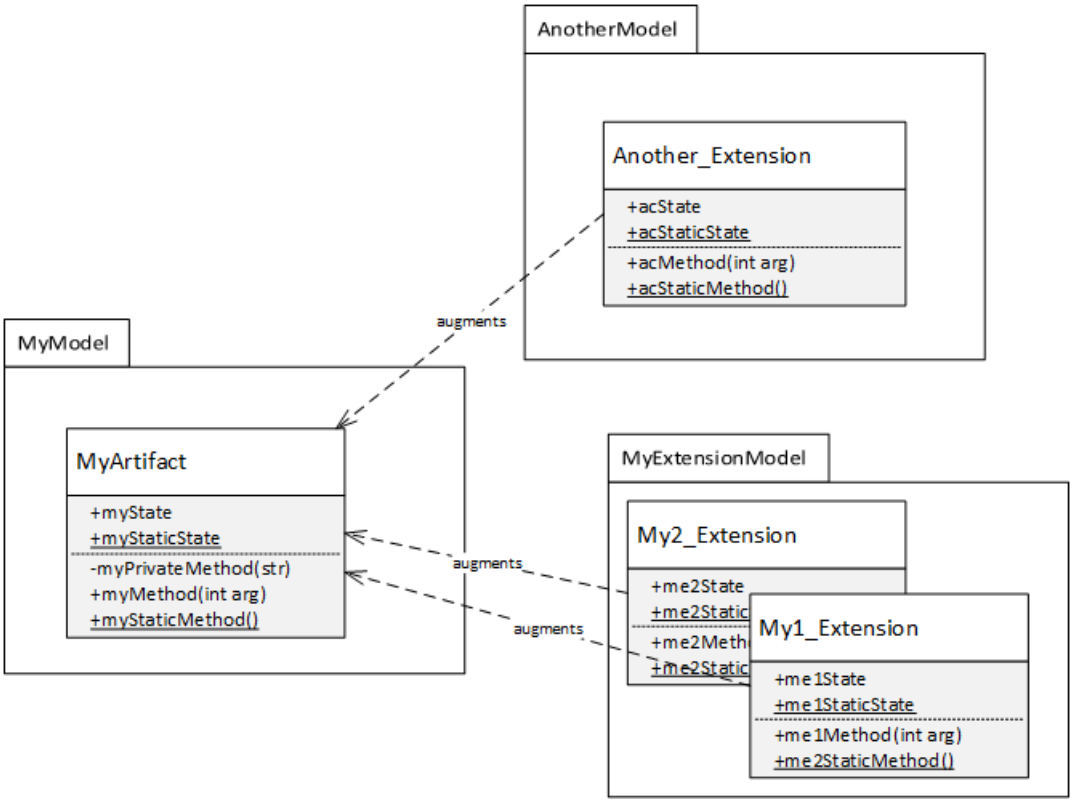
2/18/2021 • 7 minutes to read • Edit Online

This article describes the new class extension model in X++.

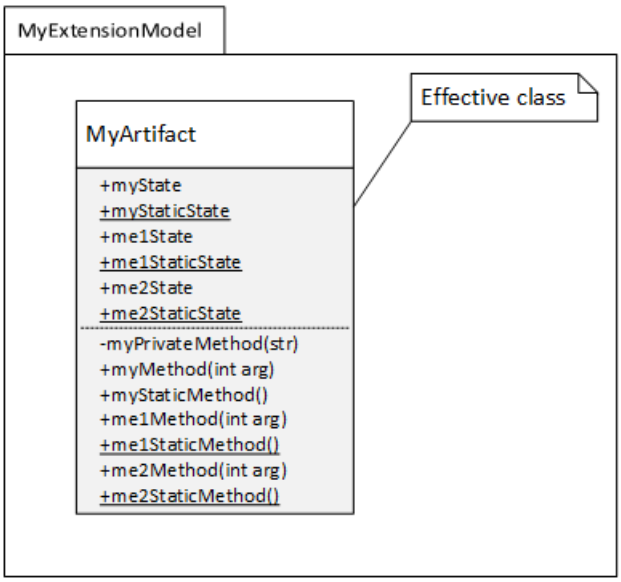
Because over-layering is a very intrusive feature, we recommend that you not use it. The alternative to over-layering is extension. Extension lets you extend existing artifacts in a new model. Extensions are easier to maintain, but the amount of extension that can be done during customization is limited. There are rich ways to extend the metadata. For example, you can add new fields to a table. This article describes how X++ code can be extended, so that you can add methods and state to artifacts that are defined in other models without recompiling those models. A similar code extension mechanism already exists for X++ and is modeled after the corresponding feature in C#. Under this mechanism, a class can be designated as an extension class through a naming convention and by hosting public static methods. In the existing feature, the type of the first argument that is passed to the extension method is the type to extend. What this article describes is the next step in that direction, which offers a more capable and natural extension story. In object-oriented programming, the term *extend* has a well-defined meaning. If we say, "class B extends class A," we mean that B inherits from A, that A is B's parent class, and the usual object-oriented rules are implied. In fact, this term is even used in the X++ syntax that is used in class declarations to express this relationship. At the same time, we use the term *extension* to talk about metadata that has contributions from several models. To avoid further overloading the term *extend*, we will instead use the term *class augmentation* to designate the relationship between a class A in a base model and a class B in a model that depends on it, where B provides additional functionality to class A in the context of that model. Nevertheless, we will also continue to use the term *extension class*, because it's prevalent.

The effective class concept

It's useful to have a term for a class that consists of the public members of the augmented artifact and all the public members of all the class extensions that augment that artifact. This class is called the effective class in a given model. The following illustration shows an artifact, **MyArtifact**, that is defined in a base model, **MyModel**, and two dependent models that have extension classes for **MyArtifact**.



In this example, the effective class is the class in the extension models that contains all the original methods and all the public artifacts from the extension classes. The effective class isn't the same in every model because it includes only the class extensions that are defined in a given model. The following illustration shows the effective class of **MyArtifact** in the **MyExtensionModel** model.



We will describe class extensions by using a class that is named **MyClass** in a model that is named **MyModel**.

```
class MyClass
{
    public int mycState;
    public str mycMethod(int _arg)
    {
        // ...
    }
}
```

We can add new methods and state to **MyClass** by introducing an extension class in the extension model (**MyExtensionModel**) that builds on top of (that is, has a dependency on) **MyModel**.

Extension class declarations

Extension classes are final classes that are adorned with the **ExtensionOf** attribute and that also have a name that has the **_Extension** suffix. (This restriction on the naming might be removed later.) The name of the extension class is otherwise unimportant. The class augments the artifact that is specified in the parameter of the **ExtensionOf** attribute, as shown in the following example.

```
[ExtensionOf(classStr(MyClass))]
final class MyClass_Extension
{
    private void new()
    {
    }
}
```

Because the classes are instantiated by the runtime system, it's not meaningful to derive from the extension class. Therefore, the extension class must be marked as **final**. The extension class **MyClass_Extension** does not extend the designated class (**MyClass**). Therefore, you cannot override methods from **MyClass** in **MyClass_Extension**. The **classStr** compile-time function must be used to designate the augmented class, and it serves two purposes:

- It produces a compilation error if the **MyClass** class doesn't exist.
- The compile-time function that is used tells the compiler what kind of artifact is augmented. Artifact names by themselves don't uniquely identify a given artifact to augment. For example, forms can have the same names as tables, classes, and enums.

Any number of extension classes can augment a given artifact in a particular model. Extension classes are never referenced directly by the programmer, only by the runtime system.

Extension class inheritance

Any class that inherits from an augmented class also inherits the effective class. In other words, the classes that inherit from a class that has extensions inherit the methods that are defined in the extension classes.

Constructors

X++ supports both instance constructors and static constructors.

Instance constructors

The instance constructor is the method that is named **new**. Constructors are useful for initializing the state of the extension objects. The instance constructor that is defined in an extension class can't have parameters. Instances of the extension classes are created, and the runtime system calls their constructors as required by the usage scenario. These constructors are never explicitly called by your code. It's guaranteed that the constructor that is provided in an extension class will be called once before any instance method or the instance state on the extension class is accessed. However, if no such references are made, the constructor isn't called.

Static constructors

Static constructors are the parameter-less static methods that are named **typenew**. Static constructors can be defined on extension classes. It's guaranteed that the runtime system will call the constructor exactly once before the first reference to the extension type. You can't assume any particular order of invocation for static construction among a set of extensions. This means that you should be careful about referencing static data from other classes in static constructors.

Methods

The public methods that are defined in extension classes provide additional functionality to the augmented class in the context of the model where the extension class is defined. Only public methods are exposed in this way. You can define private methods to help implement the public methods, but those private methods aren't part of the effective class. Because extension classes are final, methods should not be marked as **protected**.

Instance methods

The following example defines an extension method named **ExtensionMethod** in a class that augments **MyClass**.

```
[ExtensionOf(classStr(MyClass))]
final class MyClass_Extension
{
    private void new()
    {
    }
    public int ExtensionMethod(int arg)
    {
    }
}
```

The public instance method (**ExtensionMethod**) is defined in the extension class. Therefore, it's available just as if it were defined in **MyClass** in the context of the model where the extension class is defined. The following example shows how to call the method in the model.

```
MyClass c = new MyClass();
print c.ExtensionMethod(32);
```

Note that the instance method that is defined in the extension class is used as an instance method on the augmented artifact. An extension method can access public and protected members only from the artifact that it augments. This behavior is by design. No artifact should be able to interact directly with state and methods that are explicitly hidden through the **private**, or **internal** keywords. Otherwise, direct interaction with explicitly hidden state and methods could cause malfunction by invalidating key implementation assumptions in those artifacts. Methods and statements in the method body can use the **this** keyword. In this context, the type of **this** is the effective class of the augmented artifact.

Static methods

Methods that are defined as public and static in the extension class are available as static methods on the artifact that is augmented.

```
[ExtensionOf(classStr(MyClass))]
final class MyClass_Extension
{
    private void new()
    {
    }
    public int method1(int arg)
    {
    }
    public static real CelsiusToFahrenheit(real celsius)
    {
        return (celsius * 9.0 / 5.0) + 32.0;
    }
}
```

The following example shows how to call the method in the model.

```
var temp = MyClass::CelsiusToFahrenheit(20.0);
```

A static method can access the public static methods and state in the effective class of the augmented artifact. As an interesting side effect, static extension methods on the **Global** class become available in the language as functions, which are available without any prefix.

State

In addition to providing static and instance methods to an artifact, you can add instance state and static state.

Instance state

Instance state, which is state that pertains to a particular instance of an artifact, can be specified on extension classes. The following example defines a state that is named **state**.

```
[ExtensionOf(classStr(MyClass))]
final class MyClass_Extension
{
    public int state;
    private void new()
    {
    }
}
```

The following example shows how to use **state** in your code.

```
MyClass c = new MyClass();
c.state = 12;
```

Static state

Static state applies to the type, not instances of the type. The following example defines a static member called **staticState** in the Extension class augmenting the **MyClass** class.

```
[ExtensionOf(classStr(MyClass))]
final class MyClass_Extension
{
    public int state;
    public static int staticState;
    static void TypeNew()
    {
        staticState = 77;
    }
}
```

Class extension - Method wrapping and Chain of Command

2/18/2021 • 12 minutes to read • Edit Online

The functionality for class extension, or class augmentation, has been improved. You can now wrap logic around methods that are defined in the base class that you're augmenting. You can extend the logic of public and protected methods without having to use event handlers. When you wrap a method, you can also access public and protected methods, and variables of the base class. In this way, you can start transactions and easily manage state variables that are associated with your class.

For example, a model contains the following code.

```
class BusinessLogic1
{
    str doSomething(int arg)
    {
        // ...
    }
}
```

You can now augment the functionality of the **doSomething** method inside an extension class by reusing the same method name. An extension class must belong to a package that references the model where the augmented class is defined.

```
[ExtensionOf(classStr(BusinessLogic1))]
final class BusinessLogic1_Extension
{
    str doSomething(int arg)
    {
        // Part 1
        var s = next doSomething(arg + 4);
        // Part 2
        return s;
    }
}
```

In this example, the wrapper around **doSomething** and the required use of the **next** keyword create a Chain of Command (CoC) for the method. CoC is a design pattern where a request is handled by a series of receivers. The pattern supports loose coupling of the sender and the receivers.

We now run the following code.

```
BusinessLogic1 object = new BusinessLogic1();
info(object.doSomething(33));
```

When this code is run, the system finds any method that wraps the **doSomething** method. The system randomly runs one of these methods, such as the **doSomething** method of the **BusinessLogic1_Extension** class. When the call to the next **doSomething** method occurs, the system randomly picks another method in the CoC. If no more wrapped methods exist, the system calls the original implementation.

Supported versions

IMPORTANT

The functionality that is described in this topic (CoC and access to protected methods and variables) is available in Platform update 9. However, the class that is being augmented must also be compiled on Platform update 9 or later. As of August 2017, all current releases of the applications for Finance and Operations have been compiled on Platform update 8 or earlier. Therefore, to wrap a method that is defined in a base package (such as Application Suite), you must recompile that base package on Platform update 9 or later. As an example: If you create your own extension model that is augmenting a class that exists in the Application Suite model, and if you are using CoC or accessing protected methods/variables, you will need to build both Application Suite and your extension model. You will also need to create a deployable package that includes both models in order to deploy this functionality on a runtime environment.

Capabilities

The following sections give more details about the capabilities of method wrapping and CoC.

Wrapping public and protected methods

Protected or public methods of classes, tables, data entities, or forms can be wrapped by using an extension class. The wrapper method must have the same signature as the base method.

- When you augment form classes, only root-level methods can be wrapped. You can't wrap methods that are defined in nested classes.
- Currently, only methods that are defined in regular classes can be wrapped. Methods that are defined in extension classes can't be wrapped by augmenting the extension classes. This capability is planned for a future update.

What about default parameters?

Methods that have default parameters can be wrapped by extension classes. However, the method signature in the wrapper method must not include the default value of the parameter.

For example, the following simple class has a method that has a default parameter.

```
class Person
{
    public void salute(str message = "Hi") {...}
}
```

In this case, the wrapper method must resemble the following example.

```
[ExtensionOf(classStr(Person))]
final class APerson_Extension
{
    public void salute(str message)
    {
        // ...
    }
}
```

In the **APerson_Extension** extension class, notice that the **salute** method doesn't include the default value of the **message** parameter.

Wrapping instance and static methods

Instance and static methods can be wrapped by extension classes. If a static method is the target that will be wrapped, the method in the extension must be qualified by using the **static** keyword.

For example, we have the following **A** class.

```
class A
{
    public static void aStaticMethod(int parameter1)
    {
        // ...
    }
}
```

In this case, the wrapper method must resemble the following example.

```
[ExtensionOf(classStr(A))]
final class An_Extension
{
    public static void aStaticMethod(int parameter1)
    {
        // ...
        next aStaticMethod(parameter1);
    }
}
```

> **IMPORTANT**
> The ability to wrap static methods doesn't apply to forms. In X++, a form class isn't a new class, and can't be instantiated or referenced as a normal class. Static methods in forms don't have any semantics.

Wrapper methods must always call next

Wrapper methods in an extension class must always call **next**, so that the next method in the chain and, finally, the original implementation are always called. This restriction helps guarantee that every method in the chain contributes to the result.

In the current implementation of this restriction, the call to **next** must be in the first-level statements in the method body.

Here are some important rules:

- Calls to **next** can't be done conditionally inside an **if** statement.
- Calls to **next** can't be done in **while**, **do-while**, or **for** loop statements.
- A **next** statement can't be preceded by a **return** statement.
- Because logical expressions are optimized, calls to **next** can't occur in logical expressions. At runtime, the execution of the complete expression isn't guaranteed.

> **NOTE**
> If a method is replaceable, extenders don't have to unconditionally call **next** when wrapping the method by using chain of command. Although extenders can break the chain, the expectation is that they will only conditionally break it. The compiler doesn't enforce calls to **next** for methods with the attribute, **Replaceable**.

Wrapping a base method in an extension of a derived class

The following example shows how to wrap a base method in an extension of a derived class. For this example, the following class hierarchy is used.

```
class A
{
    public void salute(str message)
    {
        info(message);
    }
}

class B extends A {}
class C extends A {}
```

Therefore, there is one base class, **A**. Two classes, **B** and **C**, are derived from **A**. We will augment or create an extension class of one of the derived classes (in this case, **B**), as shown here.

```
[ExtensionOf(classStr(B))]
final class B_Extension
{
    public void salute(str message)
    {
        next salute(message);
        info("B extension");
    }
}
```

Although the **B_Extension** class is an extension of **B**, and **B** doesn't have a method definition for the **salute** method, you can wrap the **salute** method that is defined in the base class, **A**. Therefore, only instances of the **B** class will include the wrapping of the **salute** method. Instances of the **A** and **C** classes will never call the wrapper method that is defined in the extension of the **B** class.

This behavior becomes clearer if we implement a method that uses these three classes.

```
class ProgramTest
{
    public static void main(Args args)
    {
        var a = new A();
        var b = new B();
        var c = new C();

        a.salute("Hi");
        b.salute("Hi");
        c.salute("Hi");
    }
}
```

For calls to **a.salute("Hi")** and **c.salute("Hi")**, the Infolog shows only the message "Hi." However, when **b.salute("Hi")** is called, the Infolog shows "Hi" followed by "B extension."

By using this mechanism, you can wrap the original method only for specific derived classes.

Accessing protected members from extension classes

As of Platform update 9, you can access protected members from extension classes. These protected members include fields and methods. Note that this support isn't specific to wrapping methods but applies all the methods in the class extension. Therefore, class extensions are more powerful than they were before.

The Hookable attribute

If a method is explicitly marked as **[Hookable(false)]**, the method can't be wrapped in an extension class. In the following example, **anyMethod** can't be wrapped in a class that augments **AnyClass1**.

```
class AnyClass1
{
    [Hookable(false)]
    public void anyMethod() {...}
}
```

> **NOTE**
>
> For compatibility reasons, **[Hookable(false)]** overrides the behavior of chain of command in addition to pre- and post-handlers. However, **[Hookable(true)]** only applies to pre- and post-handlers and does not influence chain of command wrapping.

Final methods and the Wrappable attribute

Public and protected methods that are marked as **final** can't be wrapped in extension classes. You can override this restriction by using the **Wrappable** attribute and setting the attribute parameter to **true** (**[Wrappable(true)]**). Similarly, to override the default capability for (non-final) public or protected methods, you can mark those methods as non-wrappable (**[Wrappable(false)]**).

In the following example, the **doSomething** method is explicitly marked as non-wrappable, even though it's a public method. The **doSomethingElse** method is explicitly marked as wrappable, even though it's a final method.

```
class AnyClass2
{
    [Wrappable(false)]
    public void doSomething(str message) {...}

    [Wrappable(true)]
    final public void doSomethingElse(str message) {...}
}
```

Extensions of form-nested concepts such as data sources, data fields, and controls

In order to implement CoC methods for form-nested concepts, such as data sources, data fields, and controls, an extension class is required for each nested concept.

Form data sources

In this example, **FormToExtend** is the form, **DataSource1** is a valid existing data source in the form, and **init** and **validateWrite** are methods that can be wrapped in the data source.

```
[ExtensionOf(formdatasourcestr(FormToExtend, DataSource1))]
final class FormDataSource1_Extension
{
    public void init()
    {
        next init();
        //...
        //use element.FormToExtendVariable to access form's variables and datasources
        //element.FormToExtendMethod() to call form methods
    }

    public boolean validateWrite()
    {
        boolean ret;
        //...
        ret = next validateWrite();
        //...
        return ret;
    }
}
```

Form data fields

In this example, a data field is extended. **FormToExtend** is the form, **DataSource1** is a data source in the form, **Field1** is a field in the data source, and **validate** is one of many methods that can be wrapped in this nested concept.

```
[ExtensionOf(formdatafieldstr(FormToExtend, DataSource1, Field1))]
final class FormDataField1_Extension
{
    public boolean validate()
    {
        boolean ret
        //...
        ret = next validate();
        //...
        return ret;
    }
}
```

Controls

In this example, **FormToExtend** is the form, **Button1** is the button control in the form, and **clicked** is a method that can be wrapped on the button control.

```
[ExtensionOf(formControlStr(FormToExtend, Button1))]
final class FormButton1_Extension
{
    public void clicked()
    {
        next clicked();
        //...
    }
}
```

Requirements and considerations when you write CoC methods on extensions for form-nested concepts

- Like other CoC methods, these methods must always call **next** to invoke the next method in the chain, so that the chain can go all the way to the kernel or native implementation in the runtime behavior. The call to next is equivalent to a call to **super()** from the form itself to help guarantee that the base behavior in the runtime is always run as expected.
- Currently, the X++ editor in Microsoft Visual Studio doesn't support discovery of methods that can be

wrapped. Therefore, you must refer to the system documentation for each nested concept to identify the correct method to wrap and its exact signature.

- You **cannot** add CoC to wrap methods that aren't defined in the original base behavior of the nested control type. For example, you can't add **methodInButton1** CoC on an extension. However, from the control extension, you can make a call into this method if the method has been defined as public or protected. Here is an example where the **Button1** control is defined in the **FormToExtend** form in such a way that it has the **methodInButton1** method.

```
[Form]
public class FormToExtend extends FormRun
{
    [Control("Button")]
    class Button1
    {
        public void methodInButton1(str param1)
        {
            info("Hi from methodInButton1");
            //...
```

- You do **not** have to recompile the module where the original form is defined to support CoC methods on nested concepts on that form from an extension. For example, if the **FormToExtend** form from the previous examples is in the **ApplicationSuite** module, you don't have to recompile **ApplicationSuite** to extend it with CoC for nested concepts on that form from a different module.

Extensions of tables and data entities

An extension class is required for each concept.

Tables

In this example, **TableToExtend** is the table and **delete**, **canSubmitToWorkflow**, and **caption** are methods that can be wrapped in the table.

```
[ExtensionOf(tablestr(TableToExtend))]
final class TableToExtend_Extension
{
    public void delete()
    {
        next delete();
        //...
    }

     public boolean canSubmitToWorkflow(str _workflowType)
    {
        boolean ret;
        //...
        ret = next canSubmitToWorkflow(_workflowType);
        //...
        return ret;
    }

    public str caption()
    {
        str ret;
        //...
        ret = next caption();
        //...
        return ret;
    }
}
```

Data entities

In this example, **DataEntityToExtend** is the data entity and **validateDelete** and **validateWrite** are methods that can be wrapped in the data entity.

```
[ExtensionOf(tableStr(DataEntityToExtend))]
final class DataEntityToExtend_Extension
{
    public boolean validateDelete()
    {
        boolean ret;
        //...
        ret = next validateDelete();
        //...
        return ret;
    }

    public boolean validateWrite()
    {
        boolean ret;
        //...
        ret = next validateWrite();
        //...
        return ret;
    }
}
```

Restrictions on wrapper methods

The following sections describe restrictions on the use of CoC and method wrapping.

X++ classes that are compiled by using Platform update 8 or earlier

The method wrapping feature requires specific functionality that is emitted by an X++ compiler that is part of Platform update 9 or later. Methods that are compiled by using earlier versions don't have the infrastructure to support this feature.

Methods on types nested within forms can be wrapped in Platform update 16 and later

The ability to wrap methods on types nested within forms (data sources and controls) by using class extensions was added in Platform update 16. This means that Chain of Command can be used to provide overrides for data source methods and form control methods.

However, wrapping (extension) of purely X++ methods on those nested types (form controls and form data sources) is not yet supported like it is on other types (forms, tables, data entities). Currently, if a developer uses Chain of Command on purely X++ methods on types inside forms, then it compiles, but the extension methods are not invoked at runtime. This capability is planned for a future update.

Unimplemented system methods on tables and data entities can be wrapped in Platform update 22 and later

The ability to wrap methods in nested classes by using class extensions was added in Platform update 16. The concept of nested classes in X++ applies to forms for overriding data source methods and form control methods.

Next calls can be put inside try/catch/finally in Platform update 21 and later

In a CoC extension method, the next call must not be called conditionally. However, in Platform update 21 and later next calls can be placed inside a try/catch/finally to allow for standard handling of exceptions and resource cleanup.

```
public void someMethod()
{
    try
    {
        //...
        next updateBalances();
        //...
    }
    catch(Exception::Error)
    {
        //...
    }
}
```

Extensions of extensions are not yet supported

Currently, only methods that are defined in regular classes can be wrapped. Methods that are defined in extension classes can't be wrapped by augmenting the extension classes. This capability is planned for a future release.

Extensions of constructors

Constructors cannot be extended. A **new** method that is defined on an extension class will define a constructor for the extension class itself. Additionally, the **new** method has to be public, and it can't have any arguments. For more information, see Constructors.

Tooling

For the features that are described in this topic, the Microsoft Visual Studio X++ editor doesn't yet offer complete support for cross-references and Microsoft IntelliSense.

Naming guidelines for extensions

2/18/2021 • 5 minutes to read • Edit Online

High level guidance: use prefixes to reduce conflicts and improve identification

Naming model elements

Every element in a model must have a name that is unique across all models at installation time. However, at installation time, you don't know the names of all the models that your model might be installed together with. To accommodate this situation, every element name should include a prefix that is specific to your solution. By including this prefix when you name elements in your model, you significantly reduce the risk of naming conflicts.

- If a model contains multiple solutions, each solution in the model can be identified by a different prefix.
- You must carefully choose the prefix to minimize the risk that other models from other parties use the same prefix for their elements.

When you extend functionality in other models, elements that are being extended already contain a prefix. However, you should not add your prefix to the extension elements, so that the names include multiple successive prefixes. Instead, you should include your prefix or another term or abbreviation as an infix when you name extension elements.

Naming extensions

An extension element, such as a table extension, view extension, or form extension, must have a unique name that minimizes the risk of conflicts with extensions in other models. To minimize the risk of conflicts, the name should include a term, abbreviation, or infix that distinguishes the extension from other extensions to the same element in other models.

- Include either the name of the model where the extension element resides or the prefix that the extension is associated with. For example, a Warehousing module extends the HCMWorker table and uses the **WHS** prefix in the name of all other elements. In this case, the extension might be named **HCMWorker.WHSExtension**. Notice that the prefix that is used to name other elements in the module is inserted as an infix in the name. As another example, an extension of the ContactPerson table in the **ContosoCustomizations** model might be named **ContactPerson.ContosoCustomizations** if the extension is intended to contain all extensions to the ContactPerson table from the **ContosoCustomizations** model. The developer tools will default to using the model's name as the extension name, since the model name is already required to be unique.
- Don't name the extension just **<Element that is being extended>.Extension**. For example, an extension of the InventLocation table must not be named **InventLocation.Extension**, because the risk of conflicts is too high.

Naming extension classes

Extension classes that are used to augment the logic on tables, classes, or other elements must have a name that is unique across all types in all models. Preferably, the extension class should include the name of the type that is being extended. However, the name must also include a term, abbreviation, or prefix that distinguishes the class from other types.

- Start the name of the extension class with the name of the type that is being augmented, and end the name

with the term **_Extension**. Therefore, an extension class that augments the ContactPerson table should start with the name **ContactPerson** and end with **_Extension**. For example, one extension class might be named **ContactPersonWHS_Extension**.

- Include either the name of the model where the extension element resides or the prefix that the extension is associated with. For example, a Warehousing module uses an extension class to augment the ContactPerson table and uses the **WHS** prefix in the name of all other elements. In this case, the extension class might be named **ContactPersonWHS_Extension**. Notice that the prefix that is used to name other elements in the module is inserted as an infix in the name. As another example, an extension class that augments the ContactPerson table in the ApplicationSuite model might be named **ContactPersonApplicationSuite_Extension** if the extension class is intended to contain all extensions to the ContactPerson table in the ApplicationSuite model.
- Consider adding additional element type information in case you create a class extension for elements that can't be declared in the code (like Forms, DataSources, or FormControls). For example, **CustTableFormWHS_Extension** is the extension for the **CustTable** form.
- Don't name the extension just **<Element that is being extended>_Extension**. For example, an extension class that augments the InventLocation table must not be named **InventLocation_Extension**, because the risk of conflicts is too high.

Naming fields, field groups, indexes, relations, and metadata elements added in extensions

Fields, field groups, indexes, relations, and metadata elements added in extensions must have a name that is unique across both the element that is being extended and other extension elements. Therefore, **these artifacts should include a prefix** that minimizes the risk of conflicts across models. In addition, these artifacts should have clear terms and abbreviations so that they can be easily understood.

- Include a prefix, term, or abbreviation at the beginning of the name of the metadata node. For example, an approving worker foreign key field is added as part of a table extension, and **WHS** is one of prefixes that are dedicated to other elements in the hosting model. In this case, the field might be named **WHSApprovingWorker**.

Naming variables and methods added in extension classes

Variables and methods added in extension classes must have a name that is unique across both the type that is being extended and all other extension classes that extend the same type. You should take care to create unique and readable names for variables and methods. When you can't create a unique name, then you should apply a prefixto minimize the risk of conflicts across models.

- Create unique and readable names for variables and methods. For example, an approving worker class-level member variable might be named **approvingWorkerForLocalWarehouse** if the area of functionality is related to supporting some local warehouse extension functionality. An **approveWork** method for the same area of functionality might be named **approveWorkForLocalWarehouse**.

- When you cannot create a unique and readable name, then add a prefix, term, or an abbreviation at the beginning of the member variable or method name. For example, an approving worker class-level member variable might be named **whsApprovingWorker** if **WHS** is one of the prefixes that is used by other elements in the model. An **approveWork** method might be named **whsApproveWork** if **WHS** is a prefix that is used by other elements in the hosting model.

- Avoid generic names, because the risk is high that multiple extensions could be using the same term or that the base functionality would be enhanced with an identical name in a future release. Some example of names likely to collide are **Approver**, **Delay**, **Group**, **Lookup**, and **Process**.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Relax model restrictions to refactor overlayering into extensions

2/18/2021 • 2 minutes to read • Edit Online

Development tools for Finance and Operations apps, starting with version 8.0, do not allow Microsoft code to be customized by using over-layering. Instead, extension capabilities should be used to modify and add behavior. The "no over-layering" restriction is a key part of the evolution of the product toward providing customers with a cloud service that is simple to update and always running the most recent version possible to allow all customers to receive the benefits of the latest features and fixes.

After you upgrade code from Dynamics AX 2012 or from Dynamics 365 for Finance and Operations version 7, any customizations that still use over-layering will cause errors when you compile your code. To refactor the code, the over-layering restriction can be temporarily relaxed in the model descriptor file of the model that is being over-layered. This temporary relaxation only works on development and demo environments and cannot be deployed on runtime environments like Standard Acceptance Test (or higher) sandbox or production environments. Relaxing the descriptor restriction will enable the code to be gradually refactored to extensions, compiled, run, and then tested.

Detailed process

Complete the following steps to relax model restrictions. This procedure can be completed on a cloud environment or a local virtual machine (VM).

1. Deploy the development environment for the Finance and Operations app.
2. Run the Lifecycle Services (LCS) code upgrade service to upgrade the solution.
3. Temporarily allow over-layering in Microsoft models as needed to enable compilation.

   a. Locate the desired model within the C:\AOSService\PackagesLocalDirectory folder.

   b. Navigate to the descriptor folder. For example, \Currency\Descriptor.

   c. Open the XML file. For example, Currency.xml.

   d. Add a **Customization** metadata element inside the **AxModelInfo** metadata element to indicate **AllowAndWarn** so that the start of the file now looks like this.

   ```
   <AxModelInfo xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
   <AppliedUpdates xmlns:d2p1="http://schemas.microsoft.com/2003/10/Serialization/Arrays" />
   <Customization>AllowAndWarn</Customization>
   ```

4. Refactor over-layering to extensions and test. Make use of extension capabilities to eliminate over-layering. If needed, make extensibility requests.
5. Revert the temporary changes to Microsoft models. The deployment of a model that uses over-layering will not be possible, so it's important to ensure that the descriptor file is updated.

Prototyping extensibility requests

As a solution gradually migrates toward extensions, there will be places where an extensibility request is required to unblock the solution. To fully understand what is needed to unblock a solution and smooth the

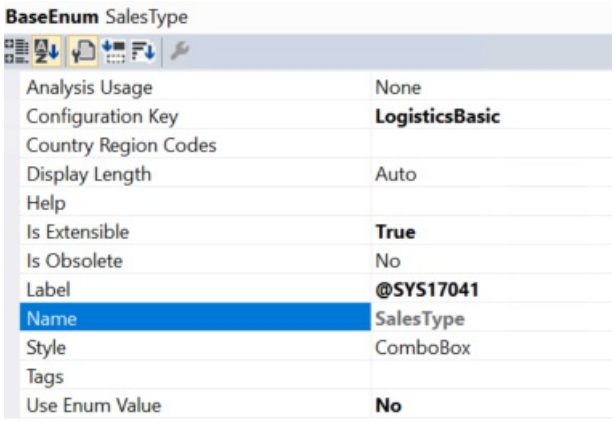
migration process toward extensions, extension capabilities can be prototyped in a separate model.

1. Identify the need for an extension capability to refactor some over-layering.
2. Add a prototype of extension capabilities into an extension prototype model.
    - For enumeration extensibility, put a copy of the enumeration into the extension prototype model and mark it as extensible.
    - For extract method extensibility, prototype the extracted method and the overlayered original in the extension prototype model.
3. Use the prototype extension capability.
4. If the prototype extension capability is sufficient, create a matching request.
5. When the extension capability has been implemented in the platform or application, the prototype extension capability and any associated over-layering should be removed from the extension prototype model.
6. After the solution has all over-layering refactored to extensions and the extension prototype model is empty, the solution refactoring is complete.

Add values to enums through extension

2/18/2021 • 2 minutes to read • Edit Online

To add new values to an enum, you should extend the enum. Any enum that is marked as **Extensible** (**IsExtensible = true**) can be extended. You can find the extensibility information in the **Properties** window in Microsoft Visual Studio, as shown in the following illustration.



When enum values of extensible enums are synchronized, the integer values of the baseline enum are deterministic, whereas the integer values of the extension enum values are non-deterministic. The values are generated during synchronization. Therefore, you can't have logic that depends on the integer value of the enum values. Here are some examples:

- Range comparisons, such as `<`, `>`, and `..`
- Modeled ranges in views and queries
- Query ranges that are created from code

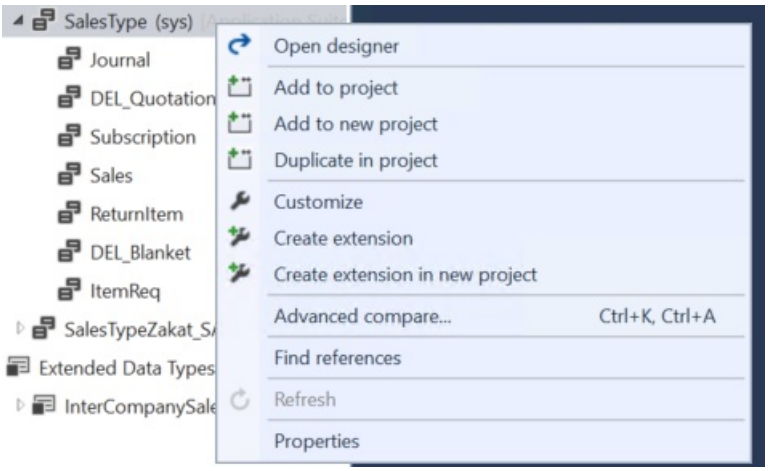
Usually, an extended enum must have its own implementation wherever it's used. Look for all uses of the enum, and uptake the implementation where it's required. Here are some significant places to look for:

- Switch blocks:
  - If the switch block doesn't have a default case block or a default case block that doesn't throw an exception, handle the extended enum value by subscribing to a delegate, if a delegate is provided. Otherwise, add a post-event handler to the method.
  - If the enum is used in a switch that has a default case block that throws an exception, contact Microsoft to request a delegate.
- If the enum has an associated class hierarchy that handles the enum, create a subclass for the extended enum–specific implementation, and uptake the construct on the base class as required. For more information, see Register subclasses for factory methods.

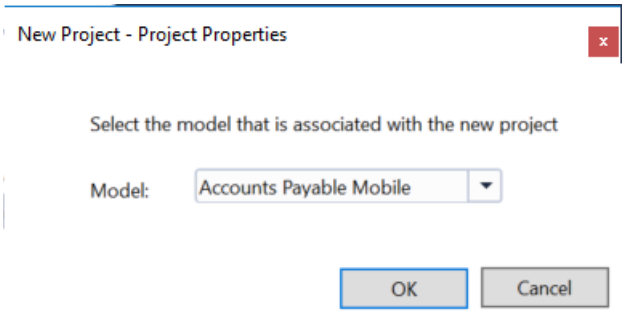
Extend an enum

There are two ways to extend an enum:

- Create a project that has a model reference where you want the new enum extension. Right-click the enum to extend, and then select **Create extension**.



- Right-click the enum to extend, and then select **Create extension in new project**. You're prompted to select the model that the extension enum should be created in.



The enum extension is created in the selected model. You can add new enum values to this extension.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

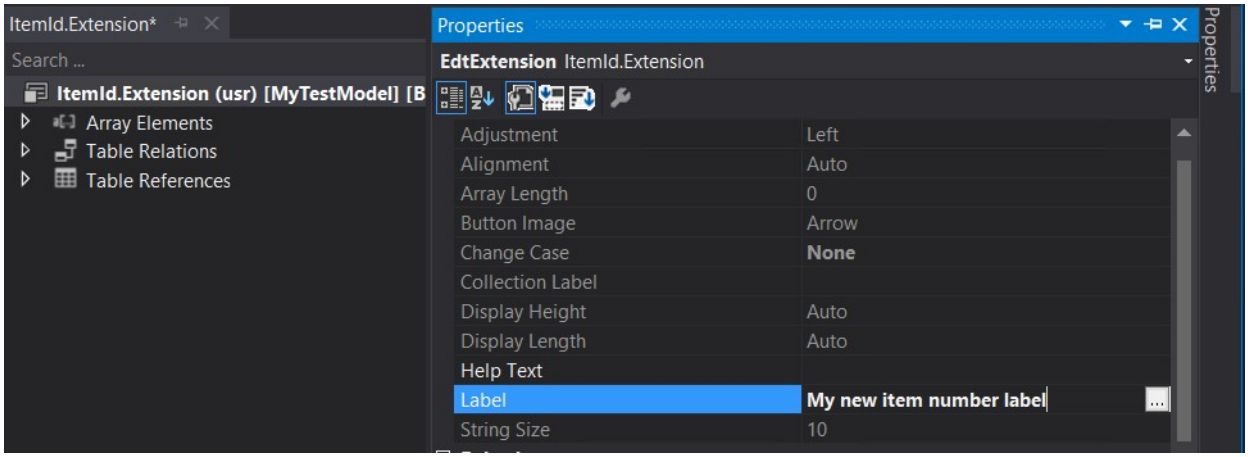
Modify extended data types (EDTs) through extension

2/18/2021 • 2 minutes to read • Edit Online

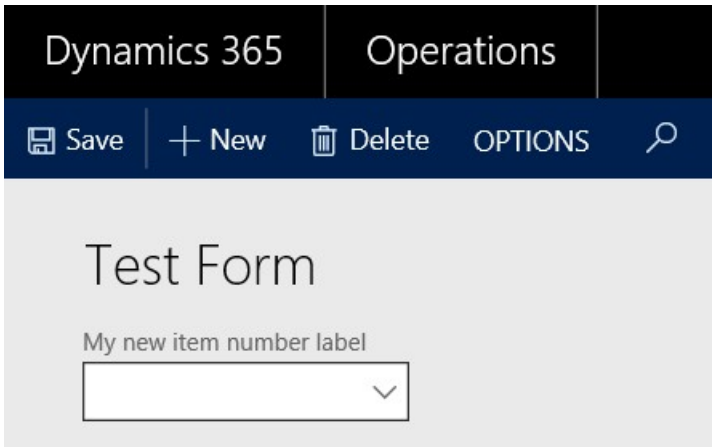
There are several properties that can be customized on existing extended data types (EDTs) through extension:

- Label
- Help text
- Form help
- Country region codes
- String size
  - You can only modify the value if the EDT does not extend from another EDT.
  - You can only set the new String size to a value equal to or larger than the base EDT value.
- Decimals (NoOfDecimals property)
  - For more information, see Extending decimal point precision for selected data types.

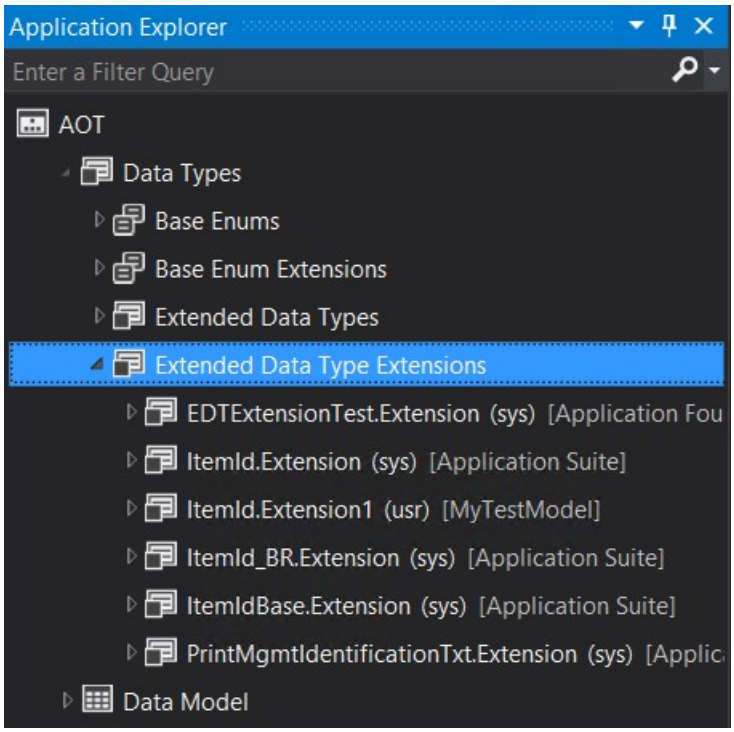
You modify the properties as you would for newly added elements, using the property sheet.



After compiling the code, you can see the changes in the application.



You can view the created extensions in the Application Explorer in Visual Studio.



If the EDT is modified in more than one model

If multiple ISVs have extended the same extended data type, the properties of the EDT from the model with the highest Model ID (closest to USR) will be used. If there are multiple models with changes in the same layer, changes from the model with the highest Model ID will be used. For example, if ISV 1 modified the label of ItemId to "Awesome item number" in model AwesomeModel (USR layer) with ID 15, while ISV 2 modified the label of ItemId to "Super item number" in model SuperModel (USR layer) with ID 12, the end user would see "Awesome item number" in the user interface instead of "Item number".

> **NOTE**
>
> Instead of extending an existing EDT, you can create a new one, deriving it from the existing EDT. This allows you to edit more properties than you could edit using the extension approach. This means that you would need to modify the fields using this EDT to use your new EDT.

Register subclasses for factory methods

2/18/2021 • 2 minutes to read • Edit Online

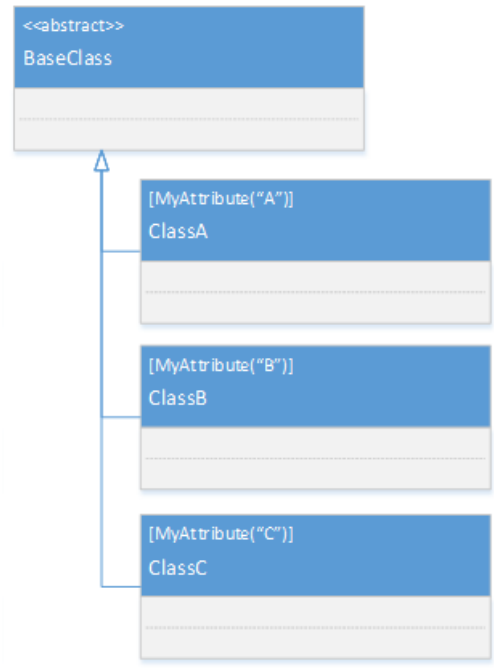
Class inheritance is a central concept in X++, as in other object-oriented languages. The object-oriented strategy pattern is used throughout the X++ business logic. In this pattern, variations in behavior can be encapsulated by subclasses, and the business process uses an abstract base class or interface. A factory method determines the variation that is used, by creating an instance of a specific subclass.

This topic describes how to register your own variations for the factories.

In X++, the factories use reflection to perform the following tasks:

- Find the correct subclass. The factory uses an extension framework to search all subclasses in a hierarchy for a specific set of attributes. If the attributes that decorate a subclass match the parameters that were passed to the factory, that specific class is used.
- Create an instance. After the right type is identified, reflection is used to create an instance of the class.

The following illustrations shows a typical decorated hierarchy.



In X++, two extension frameworks serve the same purpose. The implementer of the factory method determines which extension framework should be used:

- SysExtension
  - This extension framework uses custom attributes that make it easier to consume.
  - It supports singletons that save a little performance when the same instance is created repeatedly. This extension framework is especially useful for stateless subclasses.
  - It seamlessly supports extensible enums that are often used to determine the variant.
- SysPlugin
  - This extension framework is based on the Managed Extension Framework. The Managed Extension Framework makes the SysPlugin extension framework available to non-X++ code.
  - It uses the **ExportMetadataAttribute** attribute to decorate the variants and is string-based.
  - It uses the **ExportAttribute** attribute to make the variant discoverable.

Introduce a new variant

1. Identify the base class (or interface) of the variant that you must implement.
2. Create a new subclass of the base class, and implement your variation.
3. Identify which attribute is required in order to register your class. There are two approaches:
   - Look for attributes that are defined on other subclasses in the hierarchy.
   - Look at the implementation of the factory method. That implementation will contain the attributes that the factory method is searching for.
4. Decorate your subclass with the attribute that was used to match your variation.

SysExtension example

```
[WHSWorkExecuteMode(WHSWorkExecuteMode::About)]
class WHSWorkExecuteDisplayAbout extends WHSWorkExecuteDisplay
{
    // Your code here.
}
```

SysPlugin example

```
[ExportMetadataAttribute('CaseIAssociation', 'Lead'),
ExportAttribute('Dynamics.AX.Application.CaseIAssociation')]
class smmLeadCaseAssociationProvider implements CaseIAssociation
{
    // Your code here.
}
```

Respond by using EventHandlerResult

2/18/2021 • 4 minutes to read • Edit Online

Some delegate methods are implemented so that they can request a response from subscribing delegate handler methods. The delegate calling logic then uses the response from a potential subscriber when it continues execution after the response has been received. These delegate methods usually have a signature that has an **EventHandlerResult** parameter as the last parameter. However, because of the support for the **EventHandlerAcceptResult** and **EventHandlerRejectResult** types, the parameter can be of any type that implements the **IEventHandlerResult** interface.

- In general, the logic that is implemented in the delegate handler method should contain a condition that verifies that the subscribing logic is responsible for providing a response. It should also include logic to provide the response in the form of a result.
- When the delegate handler method must provide the response to an **EventHandlerResult** object parameter, the subscribing logic might also contain logic to calculate or retrieve the result.
- When the condition and the response logic are implemented, the calculation of the result must occur only when the condition is evaluated to **true**.
- All the subscribing delegate handler methods are run when a delegate is called. Therefore, you should make sure that the overhead of running your method is as low as possible when the method isn't responsible for providing a response. Therefore, make sure that the condition is evaluated to **false** as quickly as possible when your delegate handler method isn't responsible for providing a result.

Examples

The following example shows a delegate handler that has a condition in the form of a **switch** statement. The delegate handler also has logic to provide a response in the form of the result. The responding logic is run only when the condition is evaluated to **true**.

```
[SubscribesTo(tableStr(InventWarehouseEntity), delegateStr(InventWarehouseEntity,
validateWarehouseTypeDelegate))]
public static void validateWarehouseTypeIsSupportedStandardDelegateHandler(InventLocationType
_inventLocationType, EventHandlerResult _result)
{
    switch (_inventLocationType)
    {
        case InventLocationType::Standard:
        case InventLocationType::Quarantine:
        case InventLocationType::Transit:
            _result.result(true);
            break;
    }
}
```

When the delegate method requests a response by using an **EventHandlerAcceptResult** or an **EventHandlerRejectResult** object parameter, the subscriber is expected to respond only with an accept or a reject. The subscribing logic might also add messages to the Infolog.

The following example resembles the previous example. However, the delegate method now requests a response by using an **EventHandlerAcceptResult** object and by calling the **accept** method.

```
[SubscribesTo(tableStr(InventWarehouseEntity), delegateStr(InventWarehouseEntity,
validateWarehouseTypeDelegate))]
public static void validateWarehouseTypeIsSupportedStandardDelegateHandler(InventLocationType
_inventLocationType, EventHandlerAcceptResult _result)
{
    switch (_inventLocationType)
    {
        case InventLocationType::Standard:
        case InventLocationType::Quarantine:
        case InventLocationType::Transit:
            _result.accept();
            break;
    }
}
```

The following example shows a delegate handler method that responds by using an **EventHandlerRejectResult** object. To respond by using an **EventHandlerRejectResult** object, you can call the **reject** method or the **checkFailed** extension method. If you use the **checkFailed** method, you can add a warning message to the Infolog. Internally, the **checkFailed** method calls the **reject** method.

```
[SubscribesTo(classStr(ProdTableType), delegateStr(ProdTableType,
validateWriteProdTableInventRefTypeDelegate))]
public static void validateWriteProdTableInventRefTypeDelegateHandler(ProdTable _prodTable,
EventHandlerRejectResult _result)
{
    if (_prodTable.InventRefType == InventRefType::ProdLine)
    {
        if (! _prodTable.InventRefId || !_prodTable.InventRefTransId)
        {
            _result.checkFailed("@SYS19558");
        }
        ProdBOM prodBOM;
        select prodBOM
            where prodBOM.InventTransId  == _prodTable.InventRefTransId;
        if (! _prodTable.checkRefProdBOM(prodBOM))
        {
            _result.reject();
        }
    }
}
```

Guidelines

In addition to the previously described practices, the following general guidelines apply:

- Respond only when the subscribing logic is responsible for responding. The delegate handler methods were implemented to provide a response when a specific condition is met. Therefore, the subscribing logic must provide a result when a specific condition is met. Before the subscribing logic responds, it should not evaluate whether the result object parameter already contains a result. For example, a delegate handler method should not contain logic that resembles the logic in the following example. This logic evaluates whether the **EventHandlerResult** object parameter already contains a result when the method is run.

  > **WARNING**
  > This example is an example of code that you should **not** write.

```
[SubscribesTo(tableStr(InventWarehouseEntity), delegateStr(InventWarehouseEntity,
validateWarehouseTypeDelegate))]
  public static void validateWarehouseTypeIsSupportedStandardDelegateHandler(InventLocationType
_inventLocationType, EventHandlerResult _result)
  {
      // this if statement is an example of the bad practice.
      if (_result.hasResult())
      {
           return;
      }
      switch (_inventLocationType)
      {
          case InventLocationType::Standard:
          case InventLocationType::Quarantine:
          case InventLocationType::Transit:
               _result.result(true);
              break;
      }
  }
```

- Don't provide a response on behalf of other subscribers. If the delegate handler method isn't responsible for providing a response, the method must not provide a response. If the method provides a response when the condition isn't met, it provides a response on behalf of other subscribers. The requesting logic must be responsible for handling situations where no subscribers have responded. The delegate handler method must not contain logic that resembles the logic in the following example. This logic which provides a result when the condition is evaluated to **false**.

  WARNING

  This example is an example of code that you should **not** write.

```
  [SubscribesTo(tableStr(InventWarehouseEntity), delegateStr(InventWarehouseEntity,
validateWarehouseTypeDelegate))]
  public static void validateWarehouseTypeIsSupportedStandardDelegateHandler(InventLocationType
_inventLocationType, EventHandlerResult _result)
  {
      switch (_inventLocationType)
      {
          case InventLocationType::Standard:
          case InventLocationType::Quarantine:
          case InventLocationType::Transit:
              _result.result(true);
              break;
       // this default block is an example of the bad practice
          default:
              _result.result(false);
              break;
      }
  }
```

Extend the RunBase class

2/18/2021 • 2 minutes to read • Edit Online

When you extend functionality of the application suite, you will encounter classes that extend the **RunBase** class. This topic shows how a **RunBase** class can be augmented end to end.

For example, you want to extend the SysUserLogCleanup class. Out of the box, this class can delete records from the SysUserLog table. However, you want to archive these records to a different table before they are deleted.

The SysUserLogCleanup class is a **RunBase** class. The **RunBase** class has a dialog box, where the user is prompted for parameters before the class is run. For this example, we will add a toggle button control to the dialog box, get the value of the control, act on the value in the run method, and make sure that the value is serialized via the pack and unpack methods. Serialization helps guarantee that the user's last selection is presented again if the dialog box is reopened. It also helps guarantee that the settings are applied if the class is run in the background.

To avoid collisions with other eventual extensions, we followed these best practices:

- **Prefix members and methods**. In the example, the prefix "my" is used. This practice is important, because it helps prevent name clashes with other extensions and future versions of the augmented class.
- **Use RunBase.packExtension() and RunBase.unpackExtension()**. These methods provide serialization in a nonintrusive manner. They enable serialization of multiple extensions of the same class. The methods are available starting in Platform Update 5.

The following example shows how to implement this scenario.

```
[ExtensionOf(classStr(SysUserLogCleanup))]
final class MySysUserLogCleanup_Extension
{
    // static members
    static private SysUserLogCleanup myRunningInstance;

    // Extending class state...
    private boolean myArchive;
    private DialogField myDialogArchive;
    #define.CurrentVersion(1)
    #localmacro.CurrentList
        myArchive
    #endmacro

    public Object dialog()
    {
        Dialog dialog = next dialog();

        myDialogArchive = dialog.addField(extendedtypestr(NoYesId), "Archive");
        myDialogArchive.value(myArchive);

        return dialog;
    }

    public boolean getFromDialog()
    {
        boolean result = next getFromDialog();
        myArchive = myDialogArchive.value();
        return result;
    }

    public void initParmDefault()
```

```
    public void initParmDefault()
    {
        next initParmDefault();
        myArchive = true;
    }

    public void run()
    {
        try
        {
            myRunningInstance = this;
            next run();
        }
        finally
        {
            myRunningInstance = null;
        }
    }

    public container pack()
    {
        container packedClass = next pack();
        return SysPackExtensions::appendExtension(packedClass, classStr(MySysUserLogCleanup_Extension),
this.myPack());
    }

    private boolean myUnpack(container packedClass)
    {
        Integer version = RunBase::getVersion(packedClass);
        switch (version)
        {
            case #CurrentVersion:
                [version, #currentList] = packedClass;
                break;
            default:
                return false;
        }
        return true;
    }

    private container myPack()
    {
        return [#CurrentVersion, #CurrentList];
    }

    public boolean unpack(container _packedClass)
    {
        boolean result = next unpack(_packedClass);

        if (result)
        {
            container myState = SysPackExtensions::findExtension(_packedClass,
classStr(MySysUserLogCleanup_Extension));
            //Also unpack the extension
            if (!this.myUnpack(myState))
            {
                result = false;
            }
        }

        return result;
    }

    private void myArchiveUserLog(SysUserLog _userLog)
    {
        if (myArchive)
        {
            //...
        }
    }
```

```
    }

    // Wire up event handler for deletion of the record
    [DataEventHandler(tableStr(SysUserLog), DataEventType::Deleting)]
    static public void SysUserLog_onDeleting(Common _sender, DataEventArgs _e)
    {
        if (myRunningInstance)
        {
            myRunningInstance.myArchiveUserLog(_sender as SysUserLog);
        }
    }
}
```

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Customize application startup by using delegates

2/18/2021 • 2 minutes to read • Edit Online

In Dynamics AX 2012, there were customization points that allowed you to subscribe to events (Application.Startup delegates) that were raised when the client was initializing. These events were deprecated because there is no concept of a rich client. On the server, only server sessions are considered, however because you can migrate logic from previous releases, new events have been added to the **ApplicationStartupEventManager** class.

The following sections highlight the new data sources that you can add to existing forms by using extensions.

static delegate void onSystemStartup()

- This event occurs when the system starts up.
- It is raised once per AOS upon startup.

static delegate void onFirstTimeUserInteractiveSessionCreated()

- This event occurs when the system is creating an interactive session for the first time for a user.
- It is raised once per user per AOS.

static delegate void onFirstTimeUserNonInteractiveSessionCreated()

- This event occurs when the system is creating a non-interactive session for the first time for a user.
- It is raised once per user per AOS.

static delegate void onInteractiveSessionCreated()

- This event occurs when an interactive session is created and ready for use.
- It is raised once per interactive session creation for any user.

static delegate void onSessionCreated(boolean _isBatch, boolean _isInteractive)

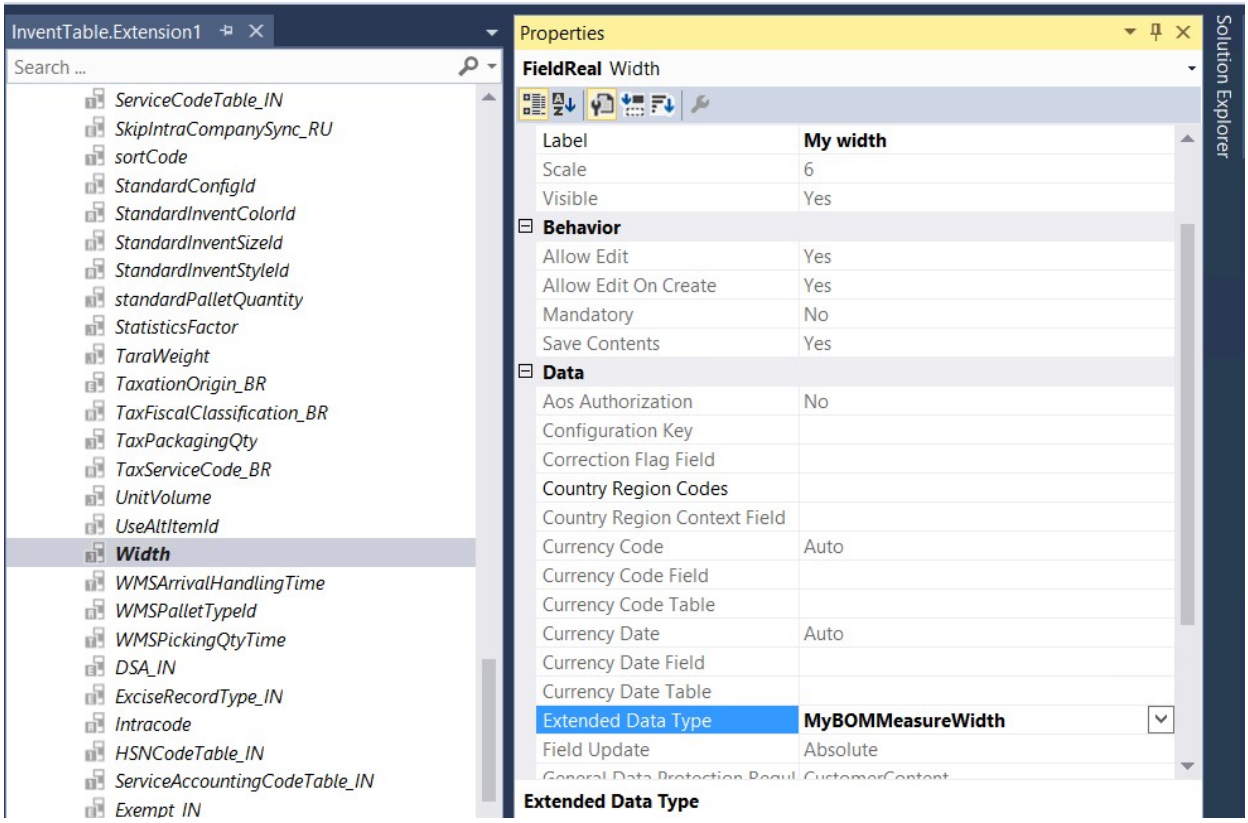
- This event occurs when the session is created and ready for use.
- It is raised once per interactive session creation for any user.
- *_isBatch* specifies whether the system is running a batch job.
- *_isInteractive* specifies whether the session is interactive.

Modify existing fields in a table through extension

2/18/2021 • 2 minutes to read • Edit Online

To modify properties on an existing field in a table, you must first create an extension for the table. You can modify the following properties:

- **Label**
- **Help text**
- **Country Region Codes**
- **Extended Data Type** – You can select only extended data types (EDTs) that are derived from the currently selected EDT. The lookup in the property sheet is filtered so that only those EDTs are shown. For example, to edit the EDT on the **Width** field in the InventTable table, you can create a derived EDT that is based on **BOMMeasureWidth**, and then modify the **Extended Data Type** property on the **Width** field in the **InventTable** extension. In this way, you can modify the look and feel of the **Width** field in the user interface when the new package is deployed.



NOTE

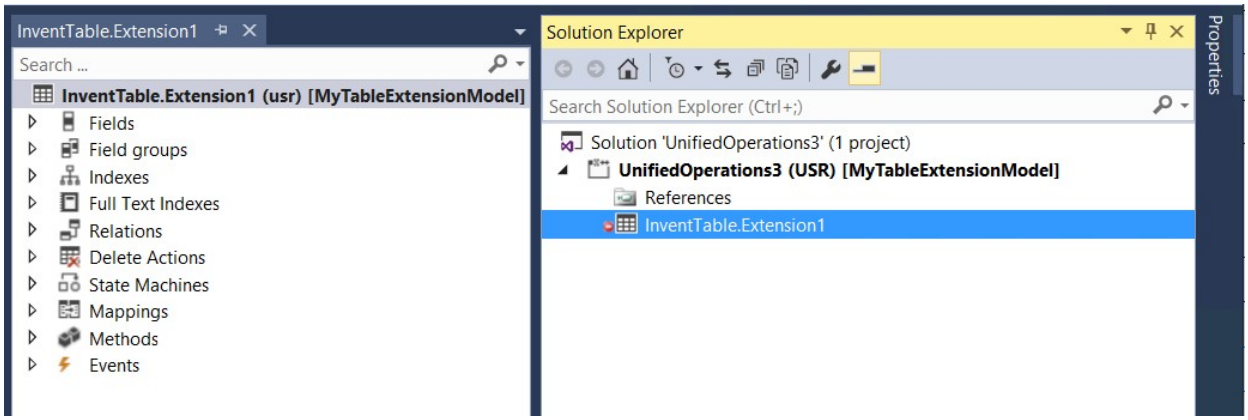
Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Add fields to tables through extension

2/18/2021 • 2 minutes to read • Edit Online

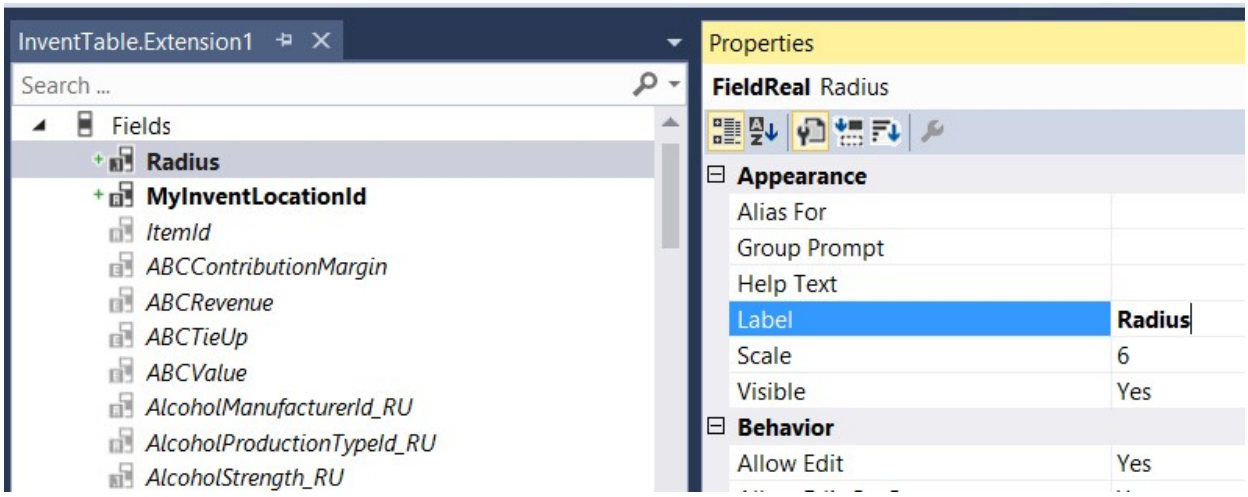
To add a new field to an existing table, you must first create a table extension. For example, to add a field that holds the radius of the released product, you must create an extension for the InventTable table in your model, as shown in the following illustration.



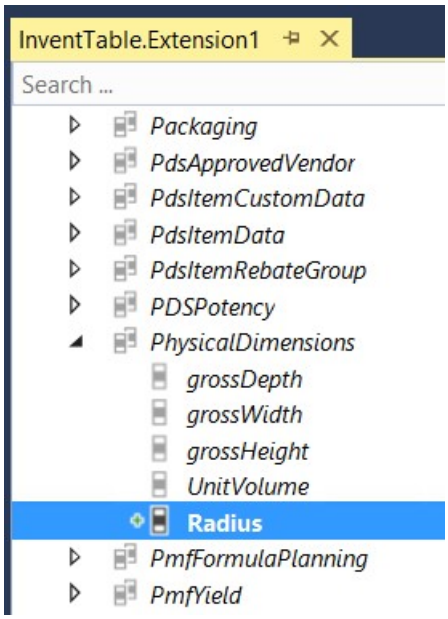
You can now add the field to the extension, just as you would add a field to a table in your model. You can use two methods:

- In the designer, right-click the **Fields** node, select **New**, and then select the type of field to add.
- Drag an existing Extended Data Type or Base Enumeration from your project onto the **Fields** node.

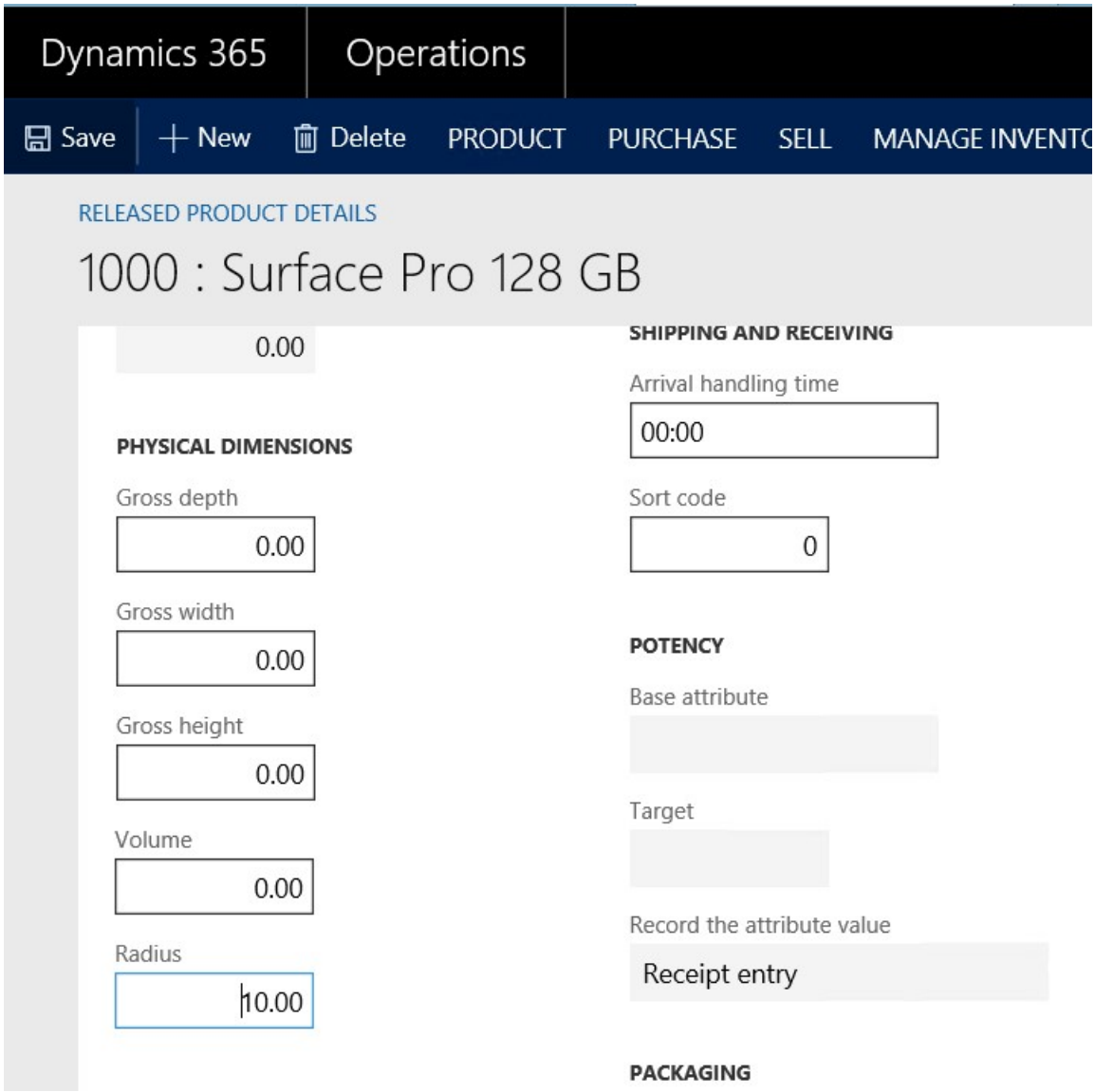
When you've finished, you can modify the properties of the new field. In the following illustration, only the **Label** property was modified.



You can now optionally add the new field either to one of the existing field groups or to a new field group that you create. In the following illustration, the **Radius** field was added to the **PhysicalDimensions** field group.



After compilation and synchronization of the database, you can see and edit the new field in the user interface.



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

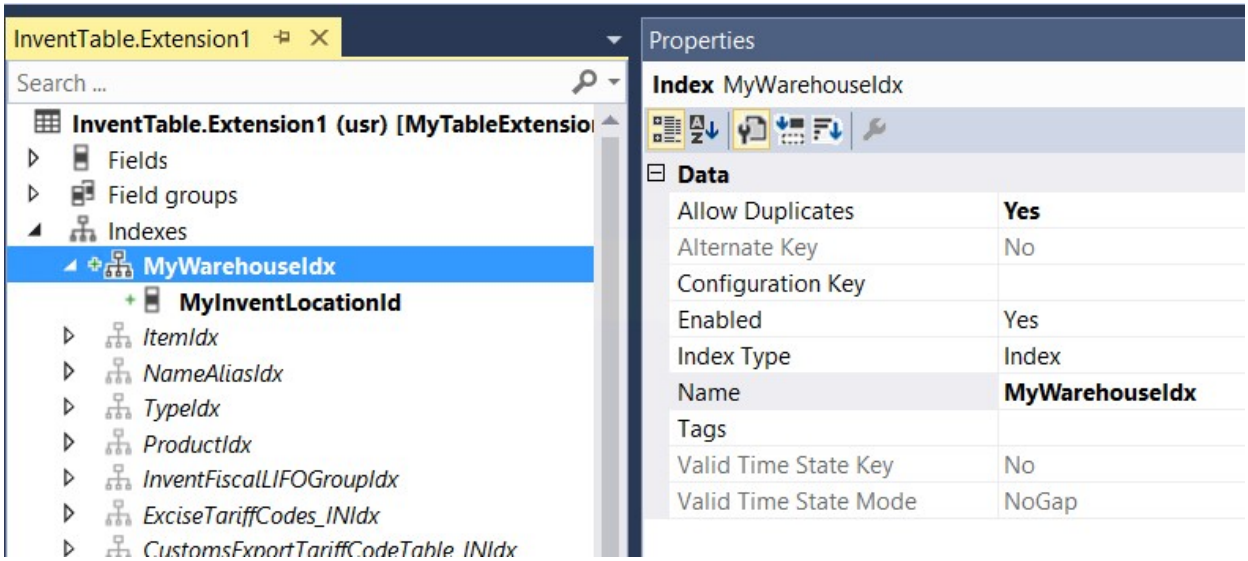
The survey will take about seven minutes. No personal data is collected (privacy statement).

Add indexes to tables through extension

2/18/2021 • 2 minutes to read • Edit Online

Often, you extend tables so that you can store additional data for later but also quickly access the data that is based on the new fields. Therefore, it's often beneficial to have a dedicated index that speeds up the database search. You can add a new index to an existing table through extension. To add an index to an existing table, you extend the selected table and then create an index just as you would create an index on a new table. You can add both new and existing fields so that they are part of the new index.

In the following illustration, an InventTable extension is used to define an index for a new field on the InventTable table.



WARNING

You should not use this approach to create unique indexes. This change is an intrusive change that might break the solutions of other independent software vendors (ISVs) if those solutions are deployed in the same environment. This capability will be removed in future platform releases.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Add relations to tables through extension

2/18/2021 • 2 minutes to read • Edit Online

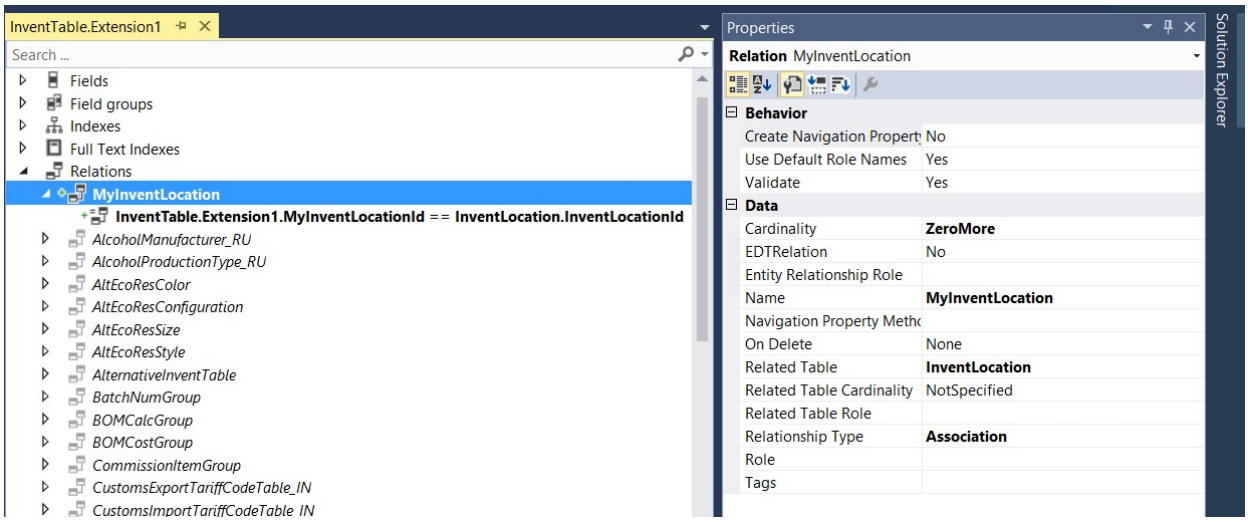
To enable rich and secure interactions with data in multiple tables, you must help guarantee referential integrity by defining relations that describe the link between two tables. By defining relations, you enable validation of the data that is entered and lookup capabilities for the related information.

You can add a new relation by extending a table.

In the following example, a new field, **MyInventLocationId**, is added to the InventTable table. This field is a reference to the InventLocation table that contains warehouses.

1. In the new extension model, create an extension of the InventTable table.
2. Create a new relation, just as you would create a relation on a regular table.
3. Specify the **Related Table**, **Relationship Type**, and **Cardinality** properties, and any other properties that apply to the relation.
4. Add the link by specifying the fields from the InventTable table and the InventLocation table that have the same value. In this case, the fields are **MyInventLocationId** in the InventTable table and **InventLocationId** in the InventLocation table.

The following illustration shows the new relation.



Troubleshooting

Navigation property methods not working

Issue - Navigation property methods do not work when a foreign key relation is created using a table extension. The compiler will not allow a call to a navigation method on the extended table.

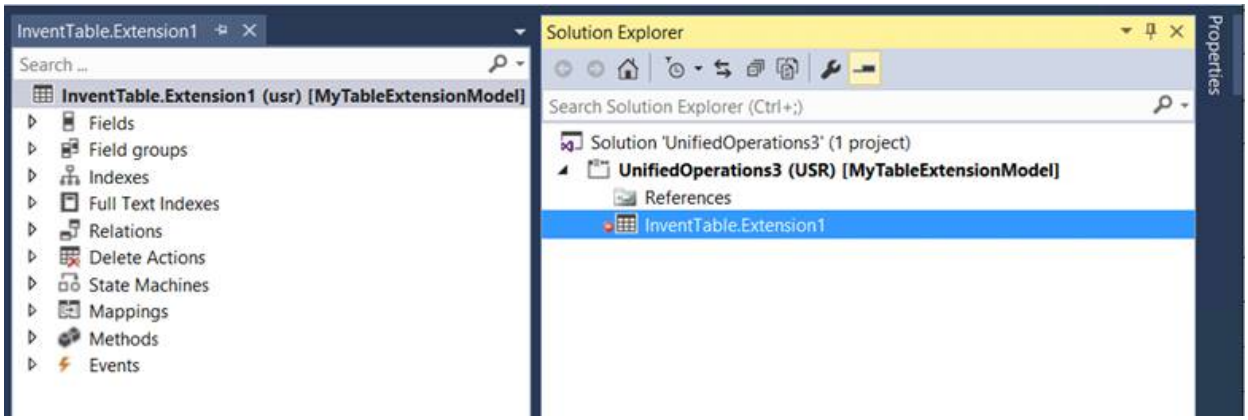
Solution - Navigation methods are not supported at this time.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Modify table properties through extension

2/18/2021 • 2 minutes to read • Edit Online

To modify properties on a table, you must create an extension of that table. In Application Explorer, right-click the table, and then select **Create extension**. A new table extension is created in the selected project, as shown in the following illustration.



You can now modify the following properties through the property sheet:

- Created By
- Created Date Time
- Modified By
- Modified Date Time
- Country Region Codes

By setting the **Created By**, **Created Date Time**, **Modified By**, or **Modified Date Time** property to **Yes**, you help guarantee that a corresponding field is added to the table. Corresponding tracking information about the user is then stored in the table when records are created or updated. You can't set these properties to **No** if they are set to **Yes** on the base table.

By adding country or region codes to the list, you help guarantee that the corresponding table is also applicable when the system runs in the context of the specified country or region.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Add methods to tables through extension

2/18/2021 • 2 minutes to read • Edit Online

When you extend the business logic that is related to a table, the general coding principles that help keep your code clean still apply. Therefore, you must eventually encapsulate actions in separate methods on the table. In Microsoft Dynamics AX 2012, you completed that task by adding the method directly on the table through overlayering. To complete the same task through extension, you use a different approach. Specifically, you create an augmentation class.

For example, a new field that is named **MyInventLocationId** was added to the InventTable table through extension. A data event handler was also created for the **Inserting** event, and you must implement the logic of filling the new field there. To encapsulate that action, you will create a new method on InventTable and name that method **myDefaultInventLocationId**.

You first create a new class in the extension model. This class will augment the InventTable table, and enable access to the table's fields and methods in a manner that is easy to read and understand. It's important that you choose the correct name for your augmentation class. This name must be unique across all types in all models that are deployed. For more information, see Naming guidelines for extensions.

```
[ExtensionOf(tableStr(InventTable))]
final class InventTableMy_Extension
{
    public void myDefaultInventLocationId()
    {
        // This would have partner specific logic to initialize the new field.
        this.MyInventLocationId = this.inventLocationId();
    }
}
```

You can now add new methods to the augmentation class. These methods will then appear in IntelliSense for variables of the **InventTable** type, just as if they were defined directly on the table. This behavior applies to both static methods and instance methods.

There are a few rules for augmentation classes:

- They must be final.
- They must be suffixed by **_Extension**.
- They must be decorated with the **[ExtensionOf()]** attribute.

Now you can use your new method, for example, from an event handler:

```
class InventTableMy_EventHandler
{
    [DataEventHandler(tableStr(InventTable), DataEventType::Inserting)]
    public static void InventTable_onInserting(Common sender, DataEventArgs e)
    {
        InventTable inventTable = sender as InventTable;
        // Call the method as if it was defined directly on InventTable.
        inventTable.myDefaultInventLocationId();
    }
}
```

NOTE

It is common for event handler classes to contain handlers for any number of events. However, it is **not** good practice to put event handlers in augmentation classes. Doing so makes the event handler methods available as methods on the augmented type. This is incorrect because the event handler is intended to be called through the event, not explicitly as a method on the type.

Perform business actions throughout the lifecycle of table records

2/18/2021 • 2 minutes to read • Edit Online

As part of your business workflows, records are regularly inserted, updated, and deleted. To customize the system behavior, you can hook into some of the record operations that are most often used. For example, you can fill additional fields on the record, perform additional data validation, or insert additional data into related tables. Several events that are available on the table let you achieve those customizations through extensions. Your code can subscribe to these events, and can insert your logic before or after the event is run.

For a list of the events that you can subscribe to, see the "Table extensions" section in Customize through extension and overlayering.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

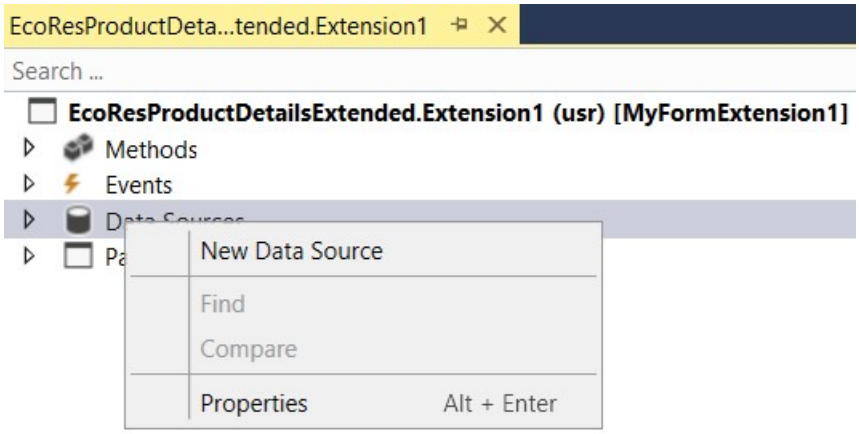
Add data sources to forms through extension

2/18/2021 • 2 minutes to read • Edit Online

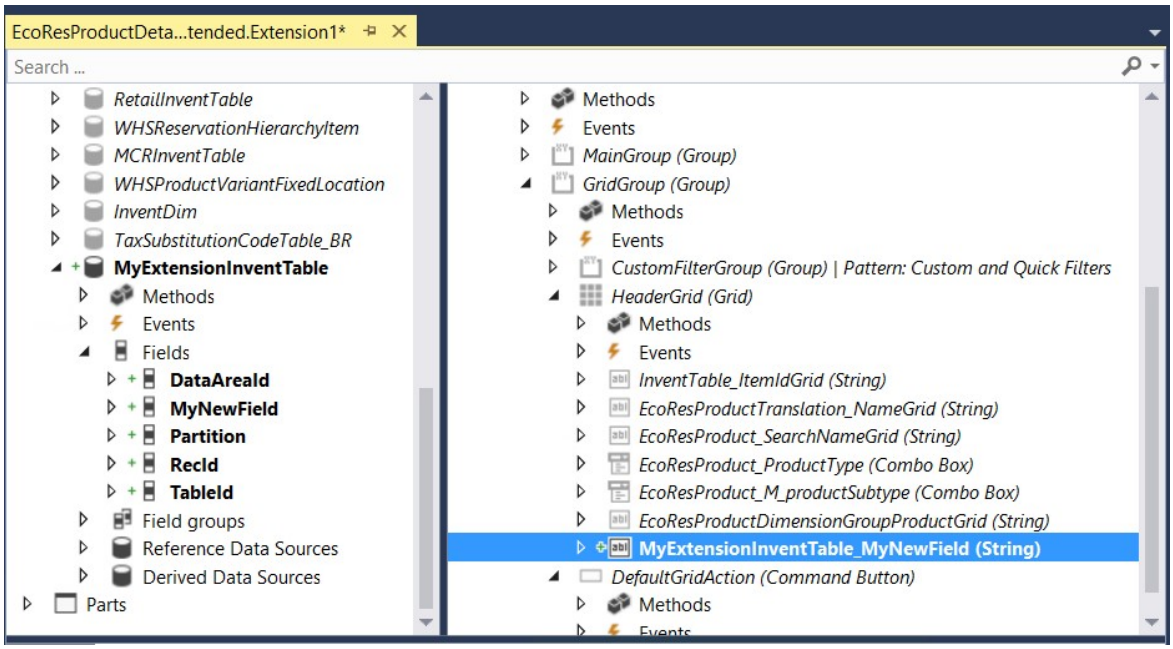
Often, the information that is stored in existing tables doesn't satisfy customer requirements. Therefore, additional tables must be created, and data from those tables must be shown on pages.

You can add new data sources to existing forms through extension. Follow these steps.

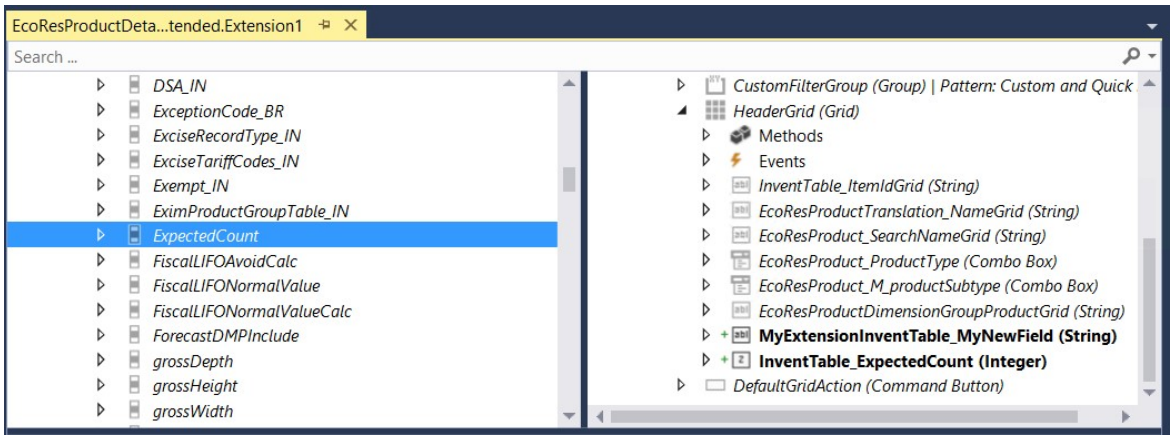
1. In the extension model, create a form extension for the selected form.
2. Right-click the form extension, and then select **New Data Source**.



3. Specify the **Table** property and other required properties on the data source. For example, define how the data source should be linked with the other data sources for the form.
4. Drag fields from the new data source into the form design, as shown in the following illustration.



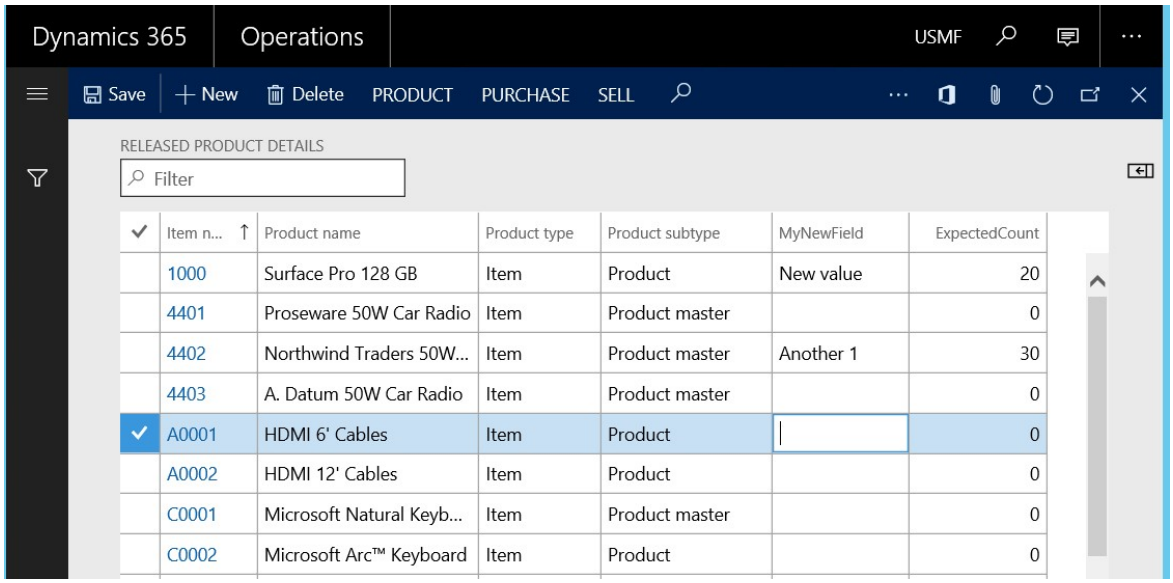
5. In a similar manner, you can add fields from existing data sources. For example, the table behind the form might have been extended with additional fields, as shown in the following illustration.



TIP

You might have to right-click the form extension data source and then select **Restore** to make the new fields appear in the list.

6. You can now view and edit the data in these new fields and tables, as shown in the following illustration.



Item n...	Product name	Product type	Product subtype	MyNewField	ExpectedCount
1000	Surface Pro 128 GB	Item	Product	New value	20
4401	Proseware 50W Car Radio	Item	Product master		0
4402	Northwind Traders 50W...	Item	Product master	Another 1	30
4403	A. Datum 50W Car Radio	Item	Product master		0
A0001	HDMI 6' Cables	Item	Product		0
A0002	HDMI 12' Cables	Item	Product		0
C0001	Microsoft Natural Keyb...	Item	Product master		0
C0002	Microsoft Arc™ Keyboard	Item	Product		0

NOTE

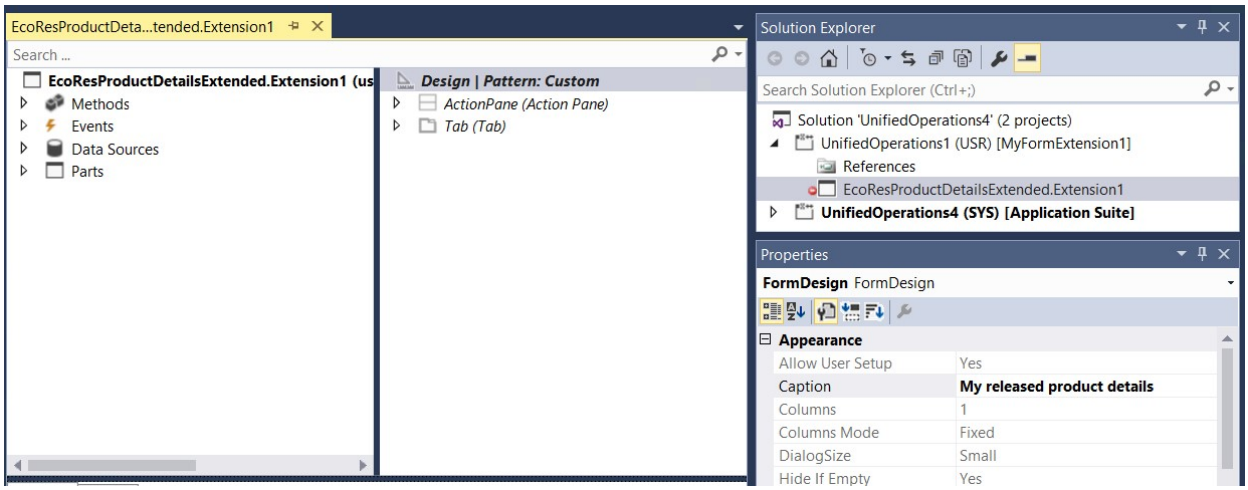
Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

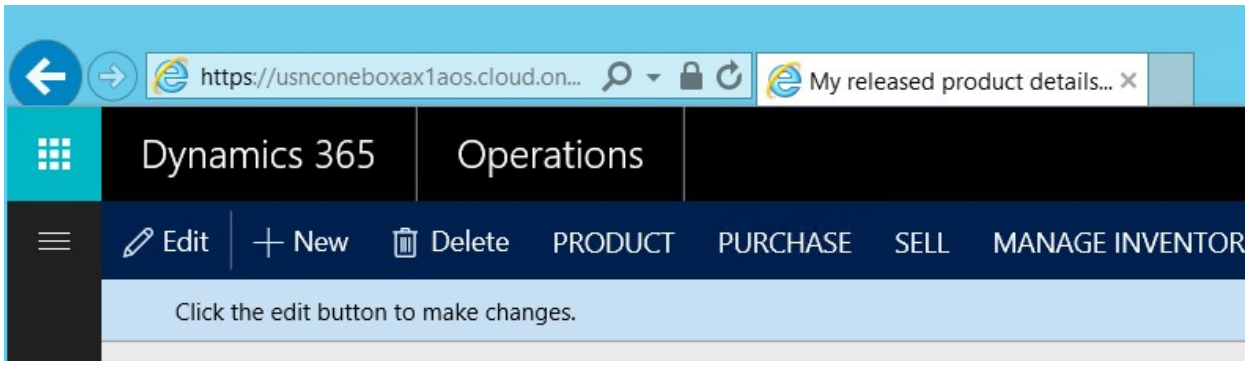
Change the captions of forms through extension

2/18/2021 • 2 minutes to read • Edit Online

The form caption appears in the page tab next to the web browser's Address bar and helps the user identify the page that is currently open. In metadata, the form caption is represented by a property on the form design. Therefore, to change the caption, you must modify the **Caption** property on the form design. You can make this change through extension. Create an extension of the selected form in the extension model, and then change the **Caption** property as usual.



The following illustration shows what the form caption looks like in a browser.



NOTE

None of the other properties on the form design can be changed.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Modify the properties of form controls through extension

2/18/2021 • 2 minutes to read • Edit Online

Often, the way that users interact with the application requires modification. Typically, you hide or disable controls on the page, replace standard labels with labels that are more appropriate, or even add new controls that the user requires. You can also create a form extension.

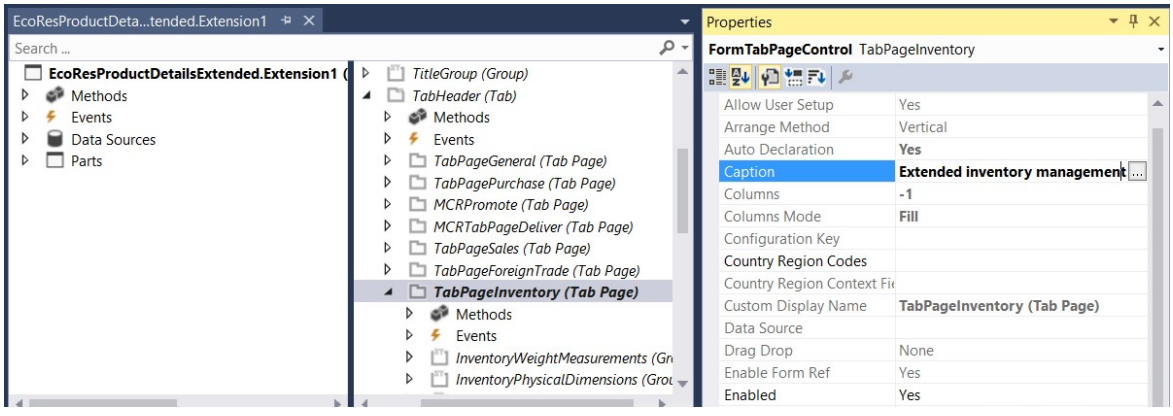
> **TIP**
>
> You can achieve even more flexibility through event subscription on form control events. This approach is discussed in other topics. In this topic, the focus is on metadata changes.

Example

The customer requires changes to the **Manage inventory** FastTab on the **Released product details** page. You must change the label of the FastTab, disable the field group that shows the catch weight configuration, and add new controls. (For this example, the new controls aren't bound to existing fields in the data source).

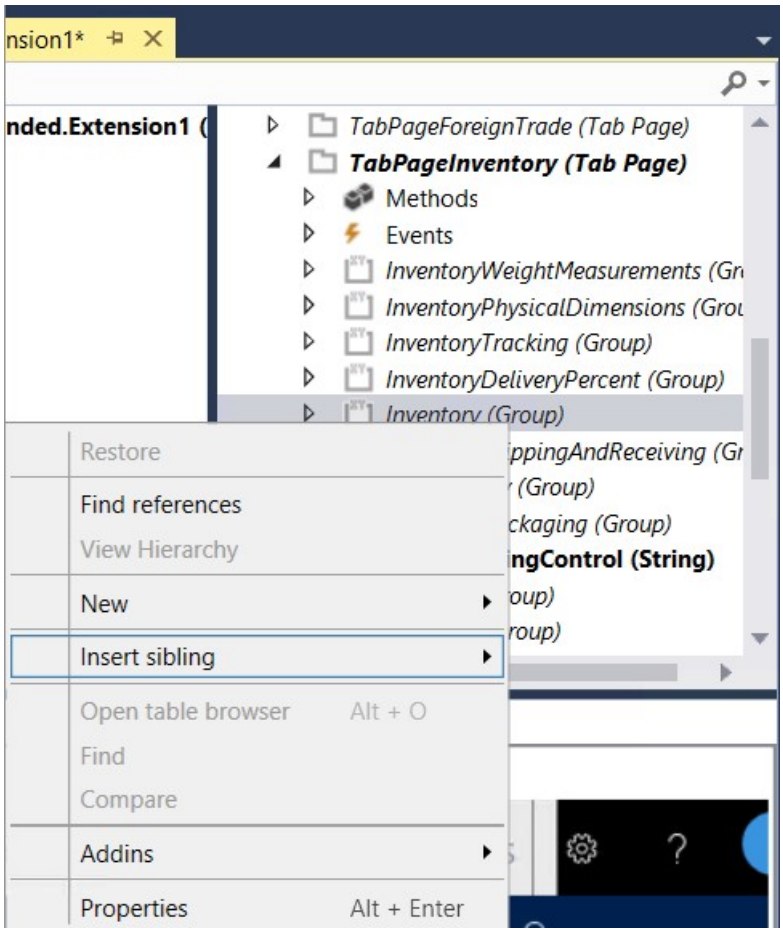
Follow these steps to make the required changes.

1. In the extension model, create an extension of the **EcoResProductDetailsExtended** form.
2. Navigate through the form design tree to the **TabPageInventory** tab page (**Design** > **Tab** > **Details** > **GroupDetails** > **TabHeader** > **TabPageInventory**), select it in the designer, and open the **Property** sheet.
3. Update the **Caption** property to the desired value.



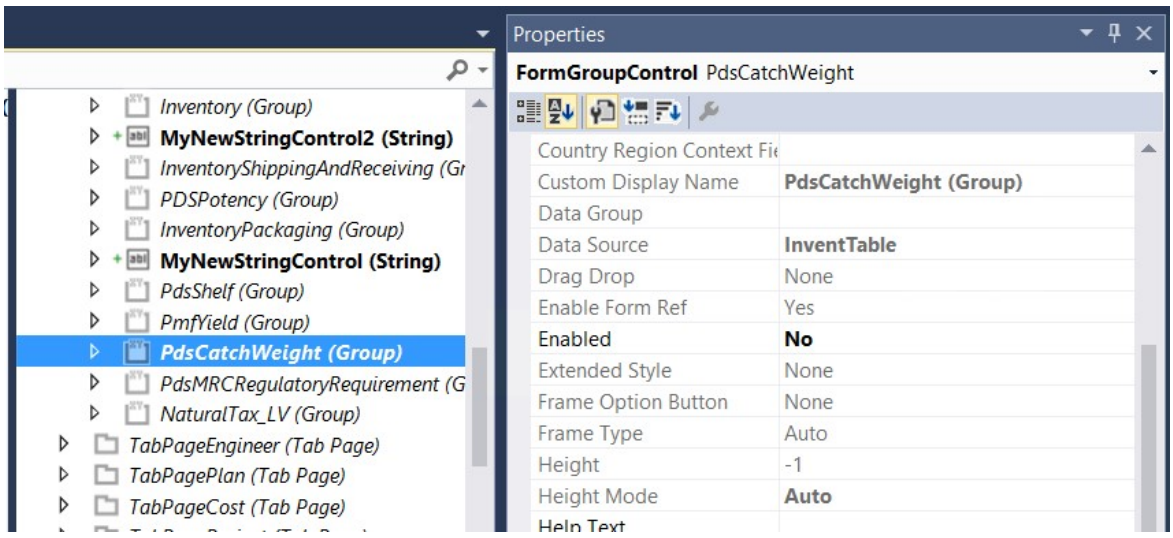
4. Right-click the tab page, and then select **New**. Set the required properties on the new control. You can also move the control up and down in the immediate container to position it correctly.

> **NOTE**
>
> Alternatively, right-click the subnode that the new control should appear after, select **Insert sibling**, and then select the type of control to add.



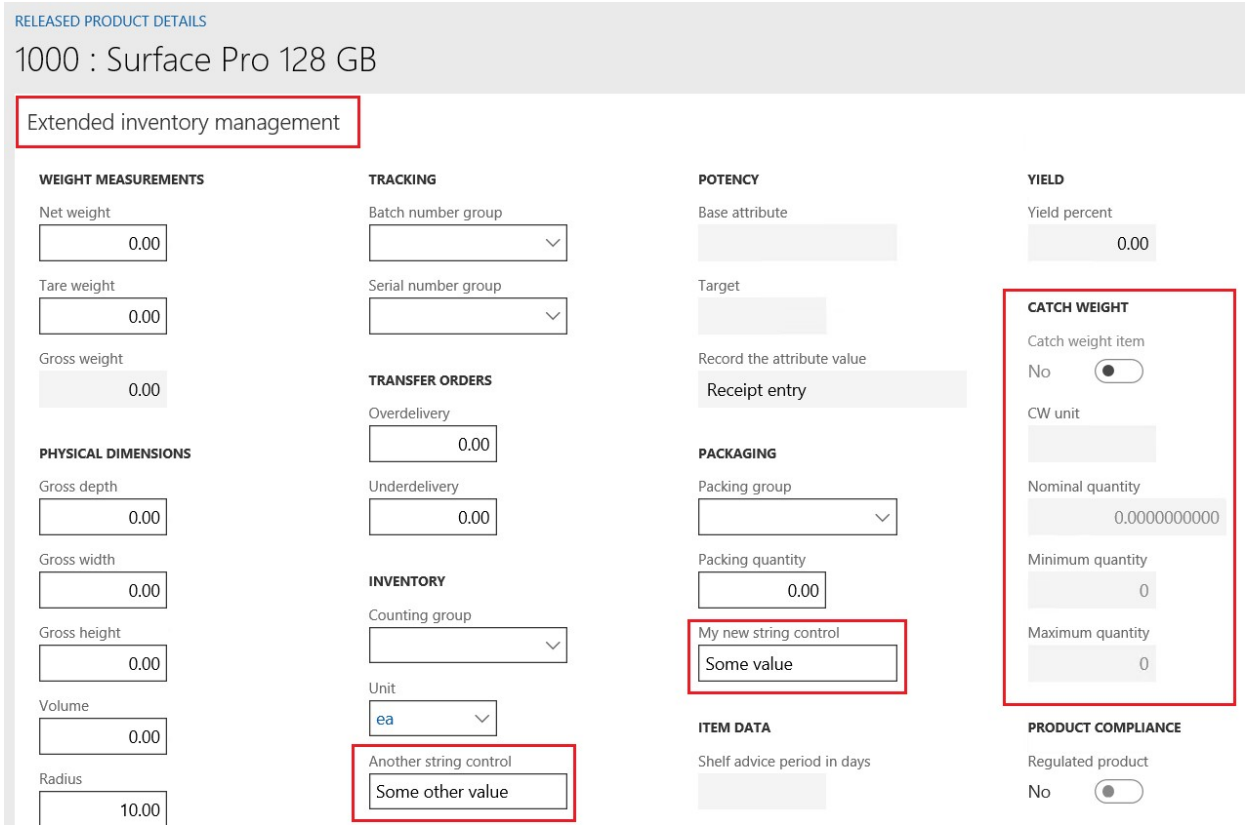
Of course, you can just drag bound controls over from the corresponding data source.

5. Select the **PdsCatchWeight** group control, and change the **Enabled** property to **No**.



> **NOTE**
>
> If you change metadata properties such as **Enabled** and **Visible**, there is no guarantee that the control will stay in that state at runtime. After a form is loaded, business logic on that form can change the state of controls through code.

When you've finished, the page includes additional fields, catch weight information can't be edited, and the whole FastTab has a different caption.



NOTE

You can't modify the **AutoDeclaration** property on controls. However, you can still access the controls by name from code.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Extend the scope of number sequences

2/18/2021 • 2 minutes to read • Edit Online

This topic shows you how to extend the number sequence scope.

The scope of a number sequence defines which organization uses the number sequence. The scope can be **Shared**, **Company**, **Legal entity**, or **Operating unit**. **Company** and **Legal entity** scopes can be combined with fiscal calendar periods to create even more specific number sequences. New number sequence scopes can be added through extensions.

To create a new scope and have it show up in the client, complete the following steps:

1. Create an enum extension for **NumberSeqParameterType**. In the extension, add a new enum value for the new scope type.
2. Create an enum extension for **NumberSequenceType**. Add a new enum value for the new scope type. The **NumberSequenceType** enum is used in **NumberSequenceTableEntity** and **NumberSequencesReferenceEntity**.
3. Create a table extension for the **NumberSequenceScope** table. Add a new field for the new scope type.
4. Create an extension class for **NumberSeqScope** class.
    a. Create a post handler for the **NumberSeqScope::getValidScopeTypes** method. In the event handler method, add the new scope type to the valid scope types list.
    b. Create an event handler for the **onGetFormatSegmentShortName** delegate. In the event handler, return the short name for the new scope type.
    c. Create a post handler for the **NumberSeqScope::find** method and add logic for the new scope type so the corresponding **NumberSeqScope** instance can be found.
    d. Create a post handler for the **NumberSeqScope::getId** method and add logic for the new scope type, so the corresponding record can be found (or created if it does not exist) in the **NumberSequenceScope** table.
5. Create an extension class for the **NumberSequenceScopeFactory** class. Add a method that initializes the new **NumberSeqScope** that represents the new scope type.
6. Create a form extension for the **NumberSequenceDetails** form. Add controls that show the new scope type to the **Scope** tab.
7. Create an extension class for the **NumberSequenceDetails** form.
    a. Create a post handler for the **updateScopeControlVisibility** method to show the new scope type when the new scope type is selected in the **Scope** box.
    b. Create a post handler for the **updateScopeControlValues** method to update the values of the controls in the **Scope** tab.
    c. Create a post handler for the **createScope** method to initialize a **NumberSeqScope** instance when the new scope type is selected.
    d. Create an event handler for the **getShortNameForParameterType** delegate to return the short name for the new scope type.
8. Add an extension class for the **NumberSequenceTableEntity** and **NumberSequencesReferenceEntity** data entities. Create post handlers for the **GenerateNumberSequenceScopeTypes** and **GenerateNumberSequenceScopeValues** methods to generate the **NumberSequenceScope** for the new scope type.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Add new inventory dimensions through extension

2/18/2021 • 4 minutes to read • Edit Online

This topic provides a high-level overview of how to add new inventory dimensions through extensions. It also includes information about how to access a sample application that contains an actual implementation.

Solution overview

The cornerstone in this solution is that multiple roles participate in the life cycle of adding new inventory dimensions through extensions. The following description simplifies and generalizes this solution, however, in real life there is overlap between the roles, and sometimes it might even be the same person filling several roles.

Microsoft's role

Microsoft provides a finite set of unused dimension fields.

In addition to the 15 existing dimensions, Microsoft supports 10 generic dimensions:

- 8 string-based
- 1 real-based
- 1 utcdatetime-based

This brings the total number of inventory dimensions in the standard application to 25:

- 5 product dimensions: Color, Size, Style, Config, and Version
- 5 tracking dimensions: Serial, Batch, Owner, Profile (Russia only), and GTD (Russia only)
- 6 storage dimensions: Site, Warehouse, Location, Status, License Plate, and Pallet (for upgrade and migration only)
- 12 unassigned generic dimensions: InventDimension1 to InventDimension12

Microsoft provides the physical schema.

ISV role

The ISV adds new inventory dimensions. The ISV solution provides all the specific functionality for the dimension - it must be strong-typed, maintainable, testable, and performant. In addition, the solution must be agnostic to other ISV's solutions. The ISV builds a solution that doesn't reference the physical schema directly, but goes through an indirection, which can be done seamlessly.

The ISV provides the logical implementation.

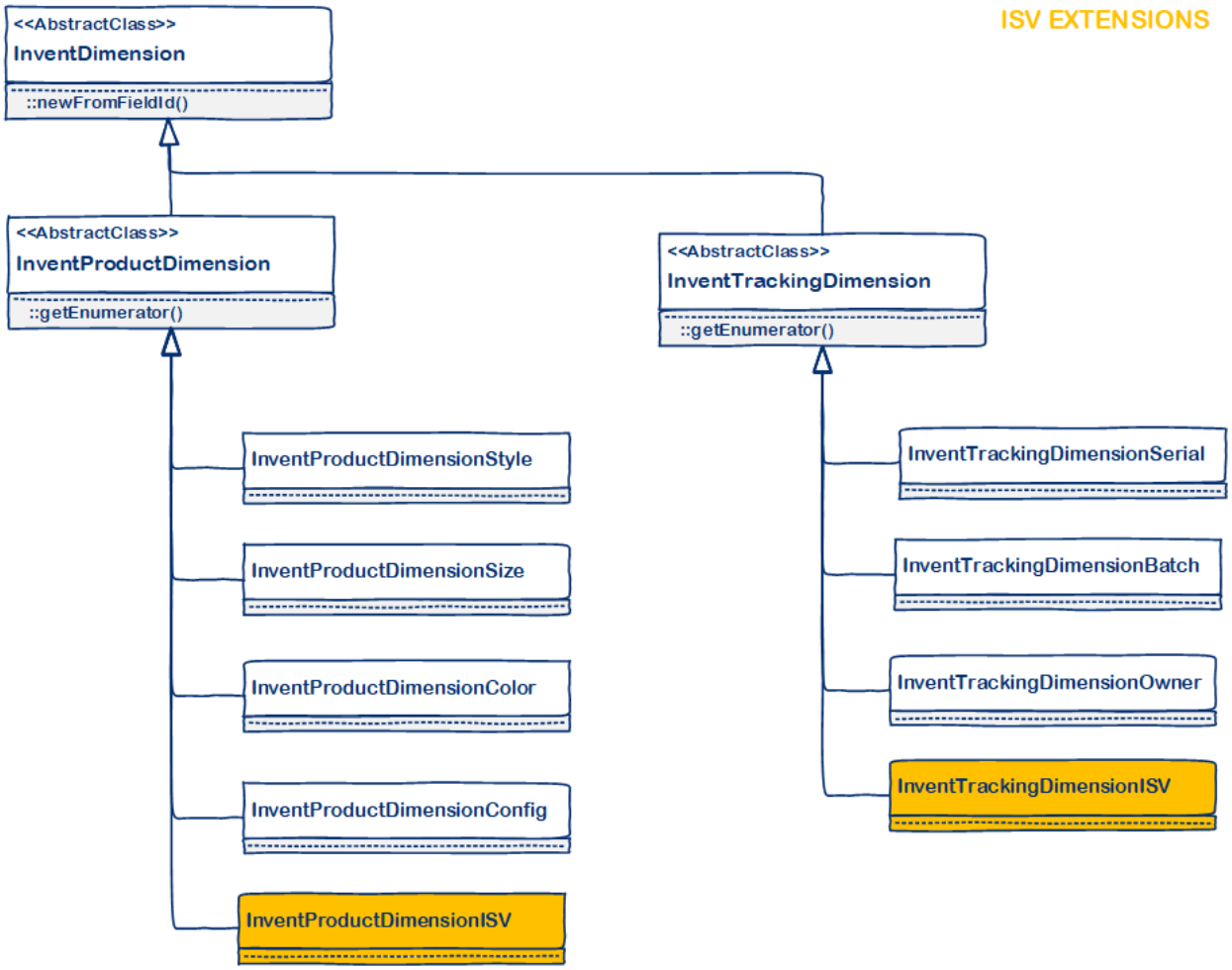
VAR role

The VAR must be able to deliver a fully functional system to a customer. The system can contain solutions from multiple ISV's - each potentially containing new inventory dimensions. In total, up to 10 ISV dimension fields are supported.

The VAR provides the binding between the physical data model and logical implementation.

Details

The first half of the solution is straight forward. A new class hierarchy is introduced. Each new dimension must be implemented in a new class deriving from either InventProductDimension or InventTrackingDimension. Currently, there is no support for storage dimensions. With this, ISVs can introduce new dimensions without having to change any of the logic on the InventDim table.



To reference the new dimension in a strongly-typed fashion, the ISV introduces a table extension class to the InventDim table. The extension classes for Style, Color, and Size can be used as templates.

Example: InventDimStyle_Extension

```
/// <summary>
/// The <c>InventDimStyle_Extension</c> class extends the <c>InventDim</c> table with behavior for the style
dimension.
/// </summary>
[ExtensionOf(tableStr(InventDim))]
final class InventDimStyle_Extension
{
    public EcoResItemStyleName parmInventStyleId(EcoResItemStyleName _style =
this.getValueForDimension(classStr(InventProductDimensionStyle)))
    {
        if (!prmIsDefault(_style))
        {
            this.setValueForDimension(classStr(InventProductDimensionStyle), _style);
        }
        return _style;
    }

    /// <summary>
    /// Returns the field id for the style dimension.
    /// </summary>
    /// <returns>The field id.</returns>
    public static FieldId fieldIdStyle()
    {
        return InventDim::fieldIdForDimension(classStr(InventProductDimensionStyle));
    }
}
```

The dimensions can be referenced like this.

```
//Setting a value
inventDim.parmISVDim("Some value");

//Select statements
select inventDim
    where inventDim.(InventDim::fieldIdISVDim()) == "Some value";
```

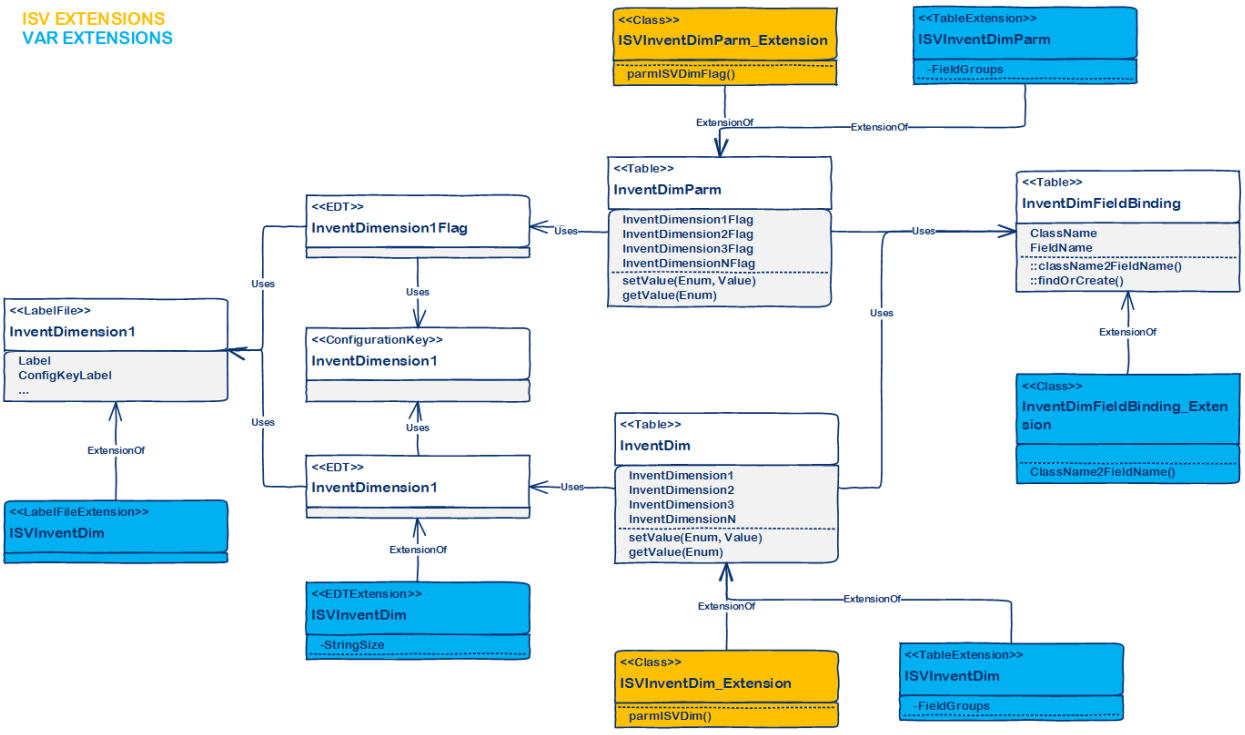
The ISV can now build logic, including the data model and user interface for maintaining the list of dimension values, for the new inventory dimension.

The second half of the solution is the data model. The standard application will contain the following for each new dimension:

- A label file.
- A configuration key.
- Two extended data types (EDTs) (for the field on InventDim and for the flag on InventDimParm).
- One field on InventDim table.
- One field on InventDimParm table.
- One field on InventDimFieldMap map and one field on each of the tables (approximately 30) mapped.

The VAR's job is to wire the ISV solutions to the available dimension fields on InventDim for a given customer. To minimize this work, it currently includes the following:

- Implement the binding mapping. This is accomplished by extending the method InventDimFieldBinding.className2FieldName().
- Enable the configuration key.
- Extend the EDT to specify the right string size.
- Extend the Label file, such as copy the ISV-provided labels into the correct label file.
- Extend the ProductDimensions or TrackingDimensions field groups on InventDim, and a few other tables, depending on the type of dimension.
- Extend relations and index as needed on InventDim.



Known issues

There are some technical limitations influencing the design of the solution. The most significant is the SQL statements throughout the application that contain where-clauses on InventDim. Most of these are implemented using macros, which doesn't change the fact that SQL statements are not extensible. Many of the SQL statements could be rewritten to use query objects to make them extensible, however many delete_from and update_recordset would remain. A viable solution cannot require changes to these SQL statements when adding new dimensions.

Another technical limitation is the amount of inventory dimensions that can be supported. Each adds a small overhead, and the InventDimFixed EDT sets an upper limit at 32. This EDT contains a bit mask for each dimension, and because the EDT is an integer, the limit is 32. The provided solution stays within the limit of 32. If required in the future, InventDimFixed could be changed to be an Int64, a container, or it could be removed.

Sample application

A sample application called "Product flavor dimension sample app" can be found on GitHub.

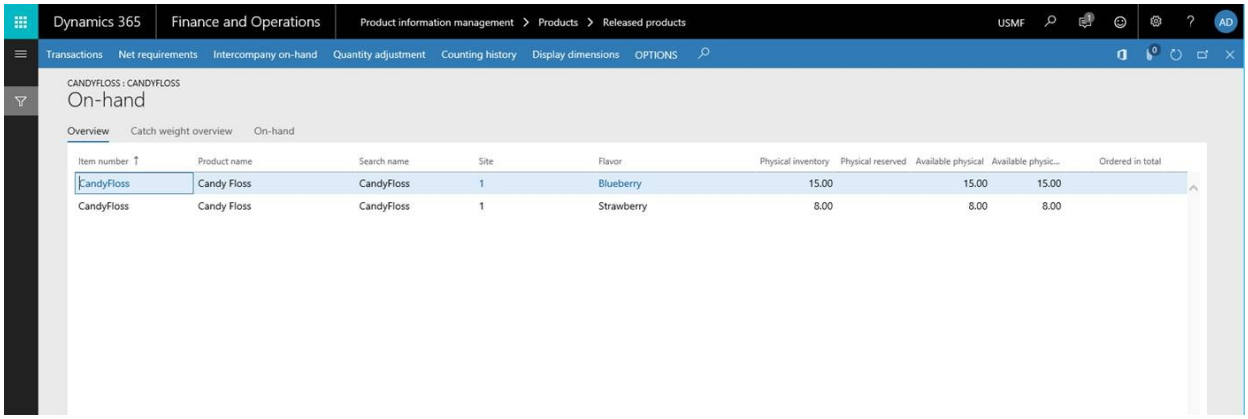
This sample consists of three models:

- ISV production code
- ISV test code
- VAR integration code

Together these models provide a great starting point for implementing new inventory dimensions. The sample application introduces a new product dimension: Flavor.

The application supports many end-to-end business scenarios, for example creating, buying, and selling items with various flavors.

If needed, please log issues directly in GitHub, and feel free to contribute to the sample application to provide additional coverage.



Price and discount extensibility

2/18/2021 • 2 minutes to read • Edit Online

In Microsoft Dynamics 365 for Finance and Operations, Enterprise edition 7.3 and later, the pricing area is extensible. Some common customizations for price and discounts include:

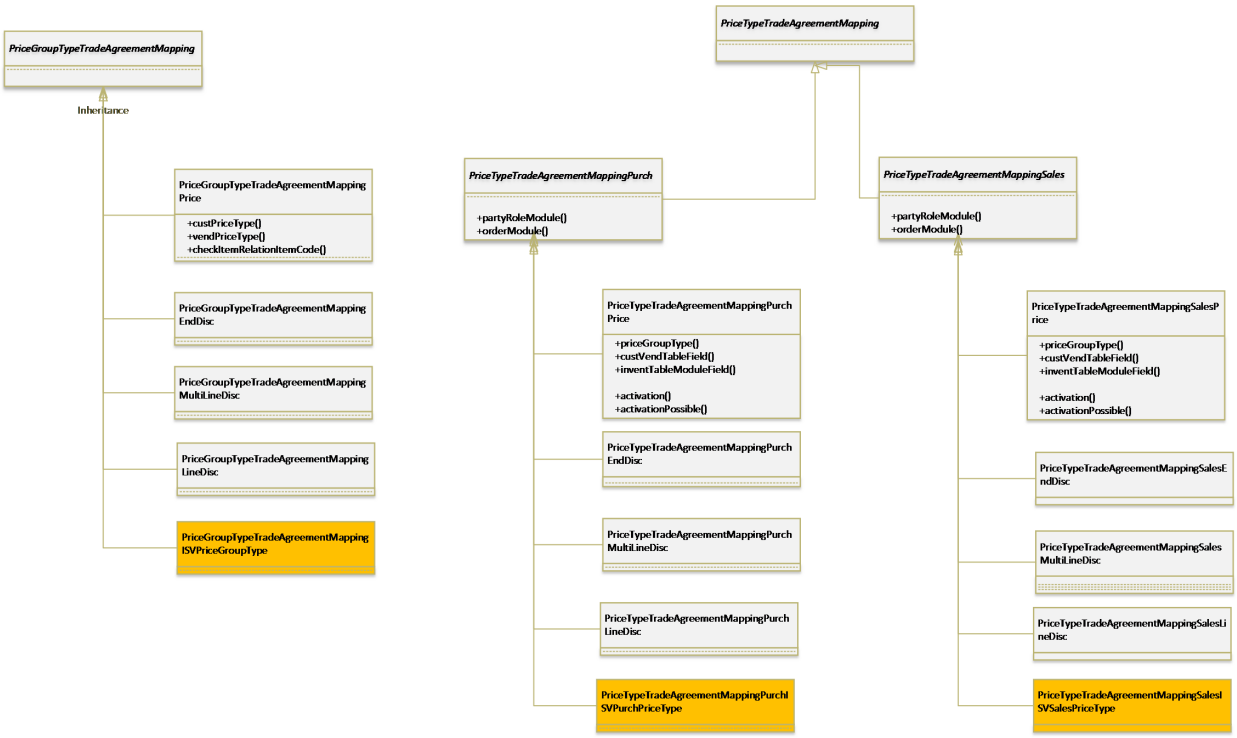
- Adding new price group types and the corresponding price types (enum values for **PriceType** and **PriceGroupType**), in addition to adding search mechanisms for the new price types.
- Modifying the price and discount search, including passing in any additional parameters to the **PriceDisc** class.

PriceType and PriceGroupType enums

Typically, adding a new type of price discount search starts with adding a new enum value in the two enums: **PriceType** and **PriceGroupType**. To support extensibility, **PriceType** and **PriceGroupType** enum values are now encapsulated in the class hierarchies **PriceGroupTypeTradeAgreementMapping** and **PriceTypeTradeAgreementMapping**, respectively. These can be extended for any new **PriceType** and **PriceGroupType** extended enum values.

The mapping of fields on the **Customer**, **Vendor**, and **InventTable** tables that correspond to the price types is defined in **PriceTypeTradeAgreementMapping**.

The following diagram highlights the implementation. Note that the methods show only one of the sub-classes. The implementation needs to be on each sub-class.

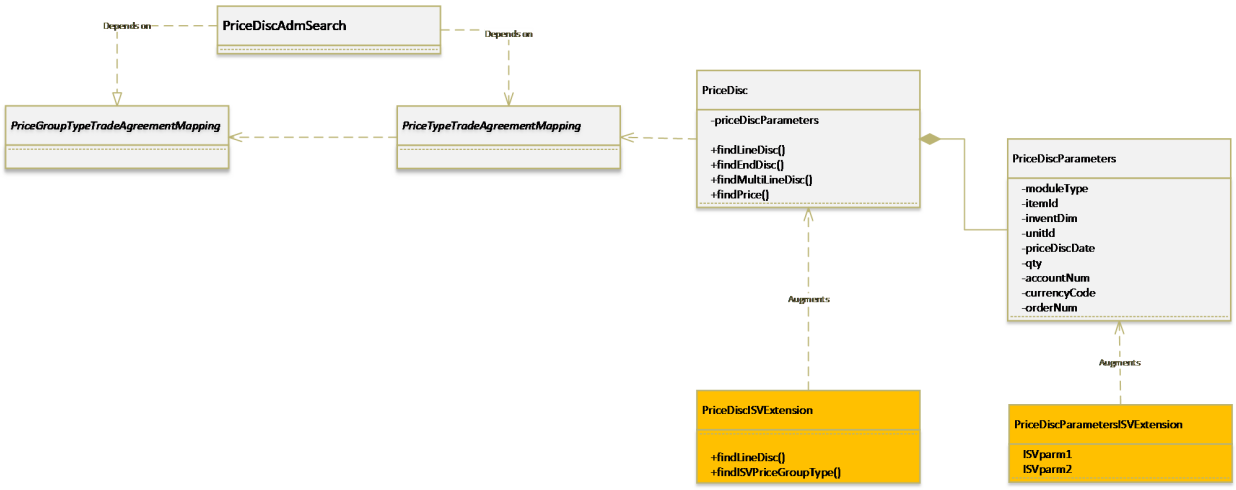


PriceDisc class

The **PriceDisc** class is the search engine for price and discounts. This class uses a **PriceDiscParameters** object as a member for passing in the parameters that are used in the price and discount search. This enables you to pass in the additional search parameters for the specific solutions. Only the parameters for a given **PriceGroupType** search are passed through the corresponding find methods on the **PriceDisc** class.

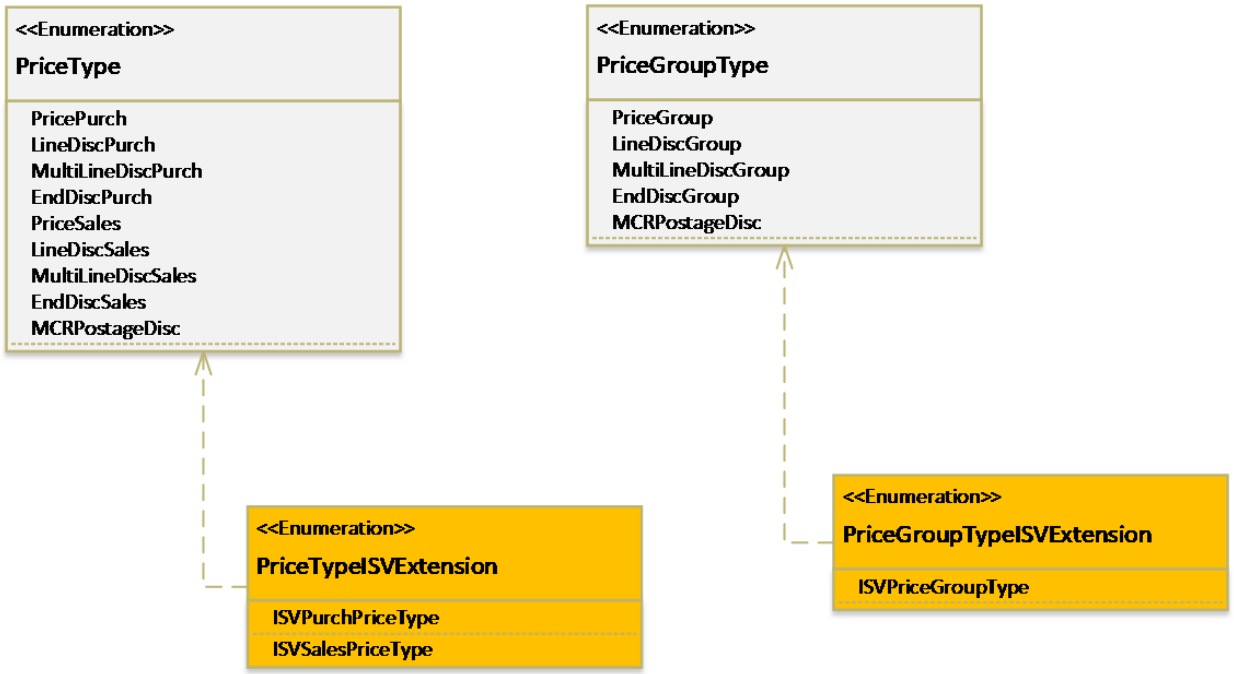
The ability to wrap and modify the instantiation of the **PriceDiscParameters** class is enabled for all price and discount search calls made throughout AppSuite.

In the following diagram, you can see how the **PriceDisc** class can be extended to modify existing searches or to add new search methods that correspond to the extended **PriceType** enum values.

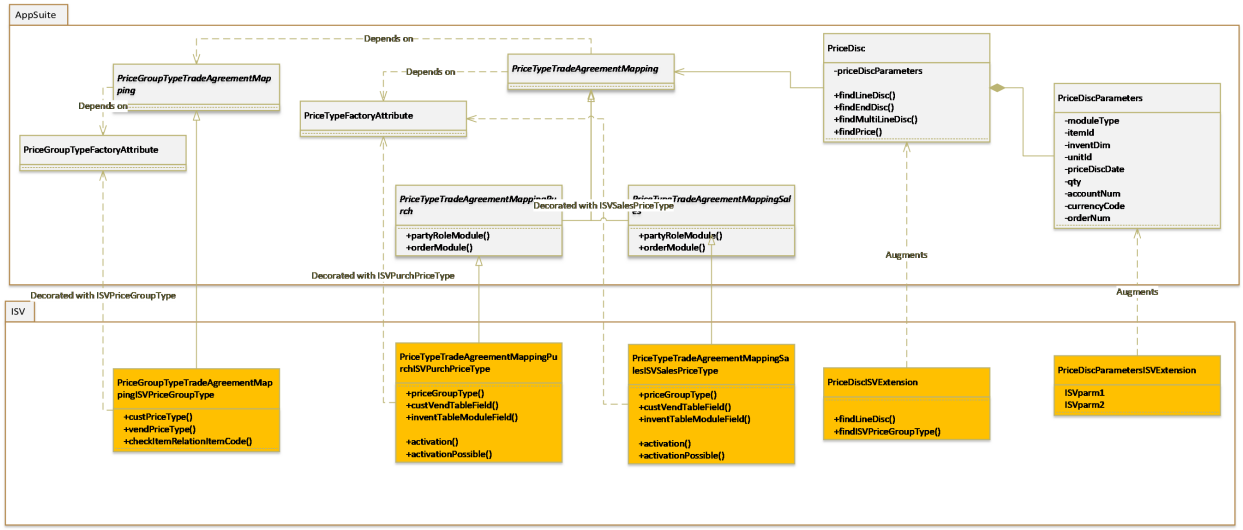


Add a new price search

In this scenario, you have extended the **PriceGroupType** enum with a new value **PriceGroupTypeISVExtension**, and two corresponding **PriceType** enum values - **ISVPurchPriceType** and **ISVSalesPriceType**.



The following diagram illustrates how a new price search can be added for the **PriceType** and **PriceGroupType** values.



This example shows the following:

- For the newly created **PriceGroupType** value, a **PriceGroupTypeTradeAgreementMappingISVPriceGroupType** class decorated with the attribute **ISVPriceGroupType** defines the behavior of the price group type.
- For the newly created **PriceType** value, the **PriceTypeTradeAgreementMappingISVPurchPriceType** and **PriceTypeTradeAgreementMappingISVSalesPriceType** classes correspond to Purchase and Sales.
- Augmenting the **PriceDiscParameters** class to add any generic parameters for the price discount search.
- Augmenting the **PriceDisc** class to create the new price discount search methods for the new price types.
- The **PriceDiscParameters** is accessible from all classes related to price and discount search and these could be augmented, based on the requirements.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

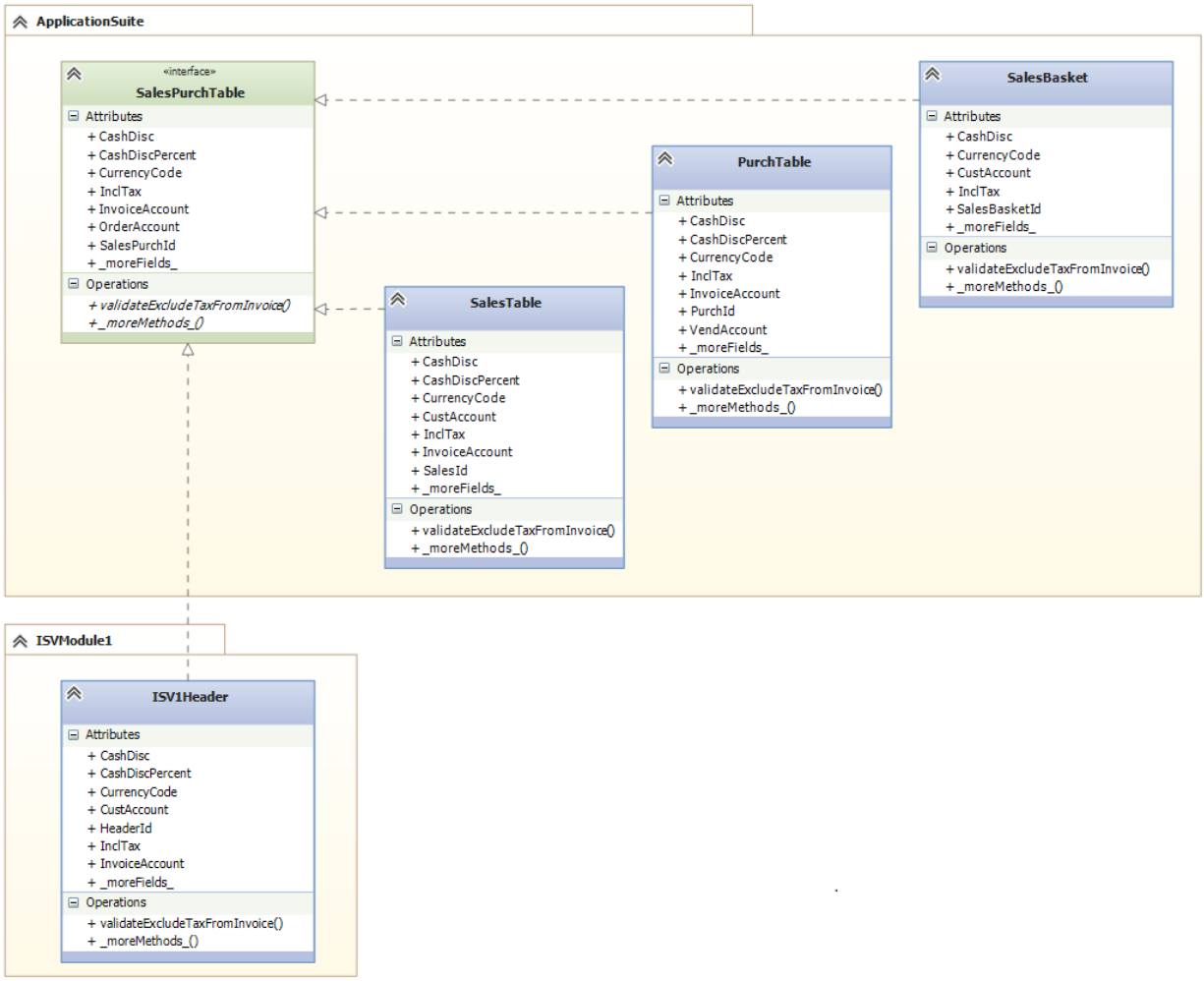
Table map extension

2/18/2021 • 2 minutes to read • Edit Online

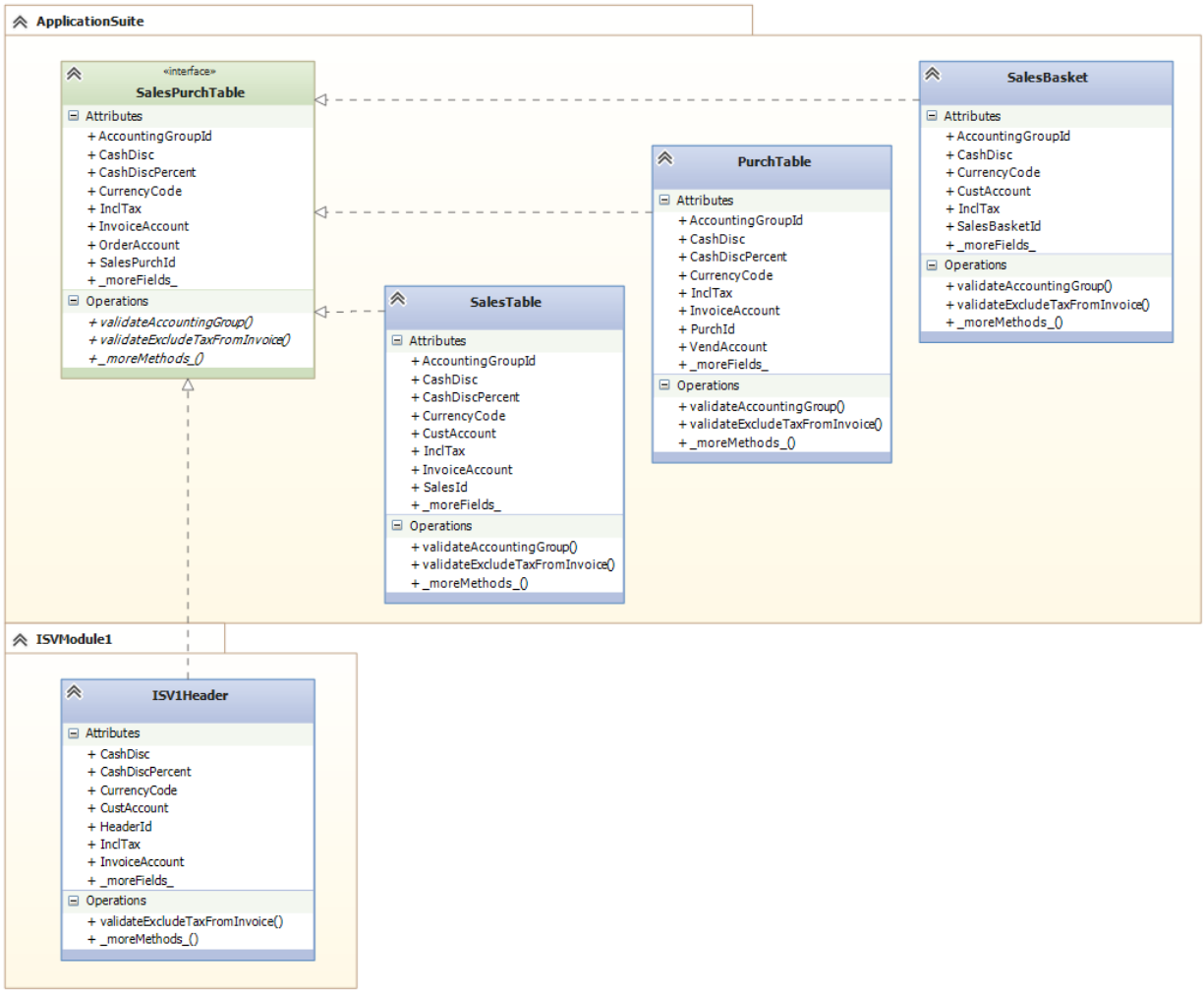
To extend table maps, we have refactored table maps into a model, which allows you to extend a solution with additional fields and methods. This topic discusses why you need a model to extend a table map.

Adding a field to an existing table map through extension can present some challenges. If these issues are not addressed during the implementation, there can be runtime errors. The errors occur because the developer cannot modify all the tables that are involved in implementing the table map. The same is true for adding a method to a table map if the method is called directly as an instance method on the table map. There is no way to enforce how fields on table maps must be mapped to fields on all tables that implement the table map. Similarly, there is no way to enforce how methods on table maps must be also be methods on all tables that implement the table map.

The following diagram shows the **SalesPurchTable** table map, which is implemented by the **SalesTable**, **PurchTable**, and **SalesBasket** tables in the **ApplicationSuite** model. In addition, the **ISV1Header** table implements the **SalesPurchTable** table map, but **ISV1Header** is part of an **ISVModule1** model.



For example, if a new field named **AccountingGroupId** and a new method named **validateAccountingGroup** are added to the table map in the **ApplicationSuite** model, then the tables that you implement the table map can be updated to include the field and method added as well. The **ISV1Header** table in the **ISVModule1** model is, however, outside of the control of the developer making the changes to the **ApplicationSuite** model.



If you add business logic to the **ApplicationSuite** model, and that logic queries the new **AccountingGroupId** field and the table map record is of type **ISV1Header**, a runtime error occurs.

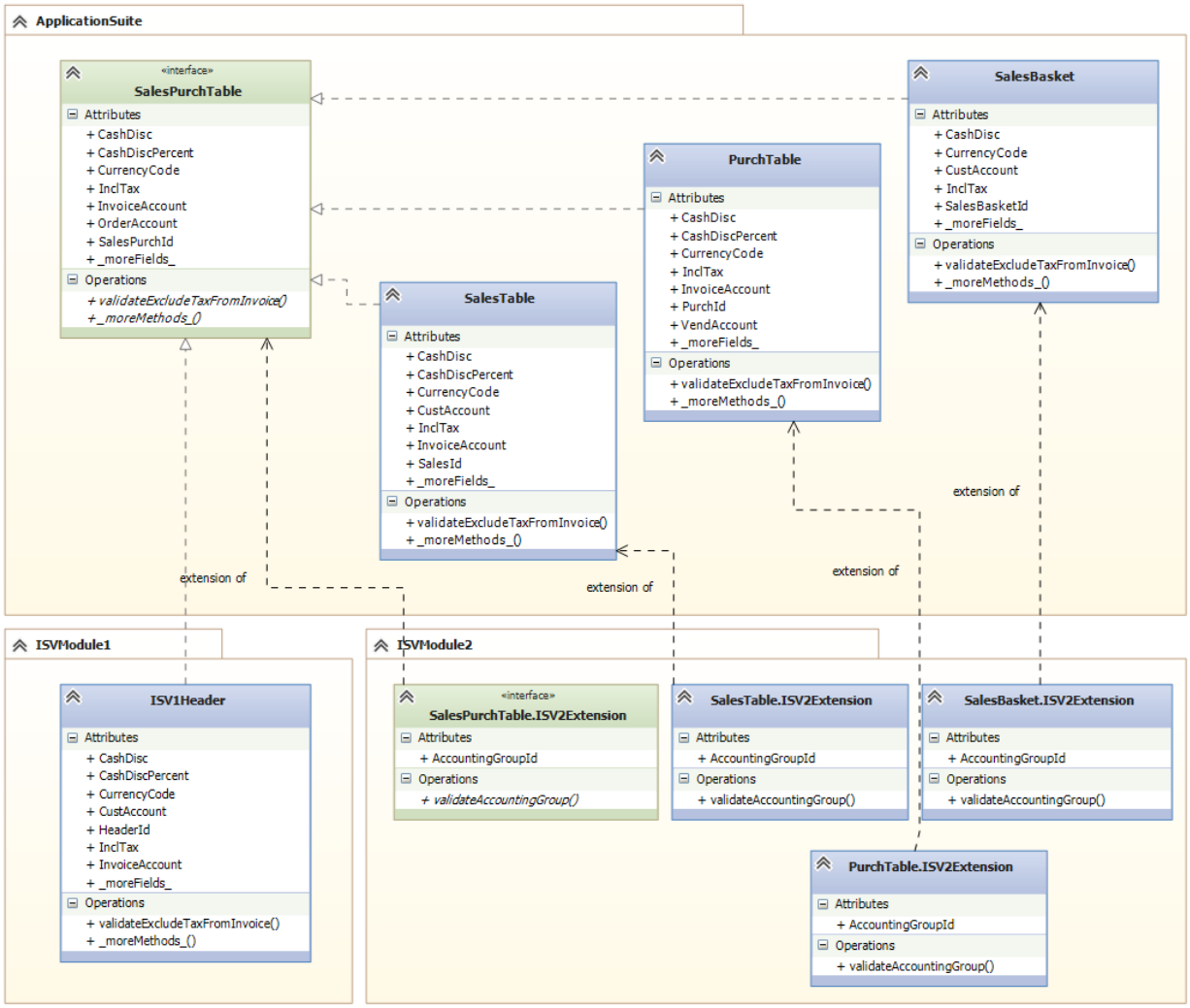
```
SalesPurchTable      headerTable;
...
...
if (headerTable.AccountingGroupId)
```

Similarly, if you add business logic to the **ApplicationSuite** model, and that logic queries **validateAccountingGroup**, then a runtime error occurs.

```
SalesPurchTable      headerTable;
...
...
if (headerTable.validateAccountingGroup())
```

As a result, the solution is broken, unless you add mapping to the new field and add the new method to the **ISV1Header** table.

The conflict is not resolved if the ability to add fields or methods is added to table maps through extension. This is illustrated in the following diagram, where **ISVModule2** includes extensions of the table map and the implementing tables in the **ApplicationSuite** model. The developer implementing **ISVModule2** has no control over the **ISV1Header** table in the **ISVModule1** model, so the **ISV1Header** table lacks a mapping of the **AccountingGroupId** field and implementation of the **validateAccountingGroup** method.



Even if the the compiler enforced that all fields and methods on a table map must be mapped to all tables implementing the table map, the conflict would not be resolved. Instead of receiving runtime errors, adding a field or a method would clear a breaking change, as tables not having a new field mapped or a new method implemented would compile when the model containing the added field/method is applied. To extend table maps, we have refactored table maps into a model, which allows you to extend a solution with additional fields and methods.

Extend table maps that are used as interfaces

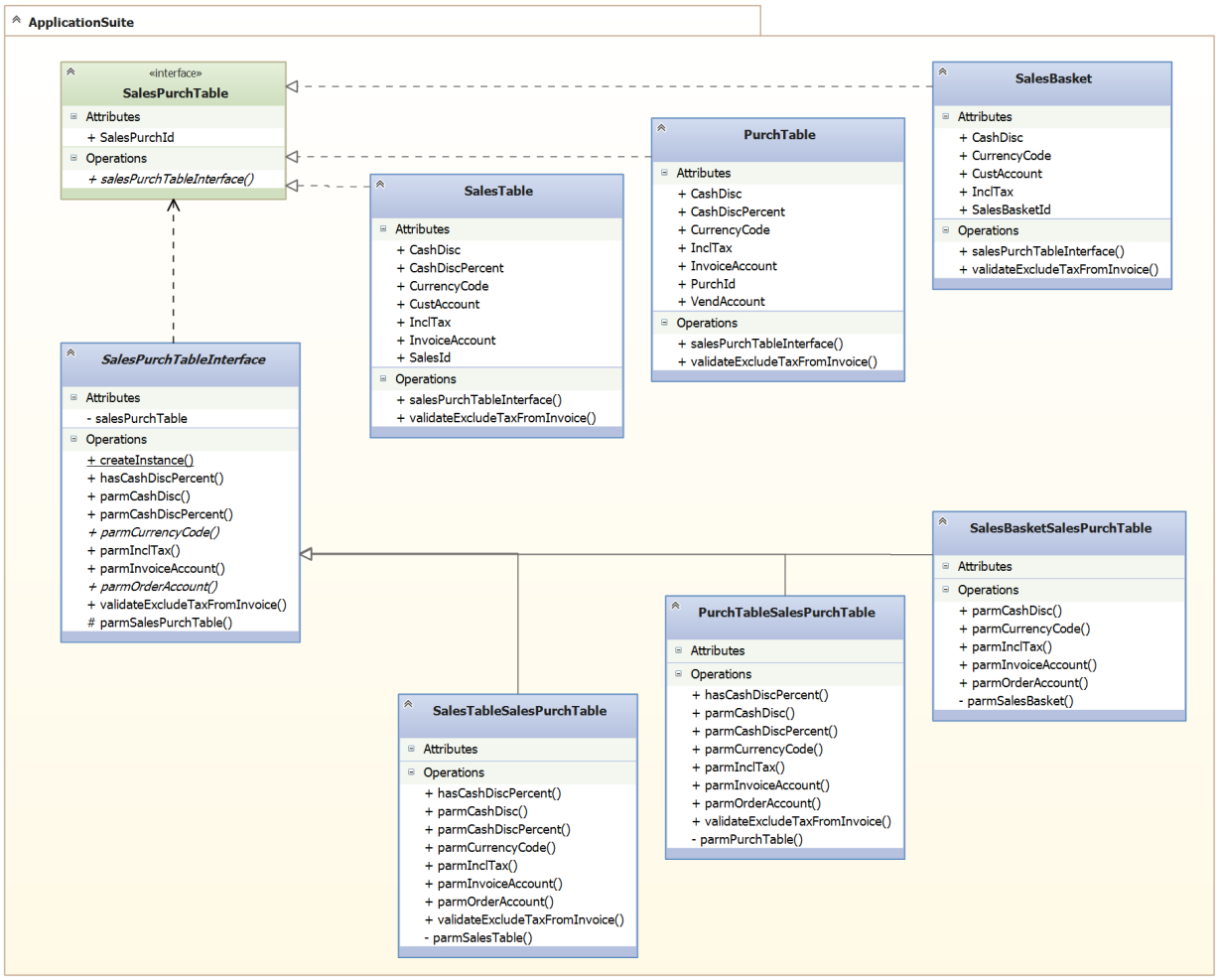
2/18/2021 • 3 minutes to read • Edit Online

The **SalesPurchLine** and **SalesPurchTable** table maps expose a set of common fields and methods that are used by a variety of product features. The mapping of fields and the implementation of methods have been refactored into a class hierarchy. Some of these changes include:

- Methods on the table maps have been moved to the class hierarchy.
- Fields are exposed through parm-methods on the class hierarchy.
- The table map still exists and tables still implement the mapping to the table map, but the fields on the table map have been made obsolete, and field mapping has been removed.
- The methods on the table map have been made obsolete.

SalesPurchTableInterface hierarchy

The new **SalesPurchTableInterface** class is the abstract base class for the ApplicationSuite functionality that has been introduced by the refactoring of the **SalesPurchTable** table map. The class contains abstract methods for fields and methods, which must exist for each table implementing the interface. It also contains the default logic in methods, which is common across all implementations. Each table that implements the **SalesPurchTable** table map must be represented as a derived class in the **SalesPurchTableInterface** class hierarchy. Each derived class must be decorated with a **SalesPurchTableInterfaceFactory** attribute class. The attribute is used to associate the derived class with the table, so that it is possible to instantiate a class of the correct type depending on a **SalesPurchTable** record.



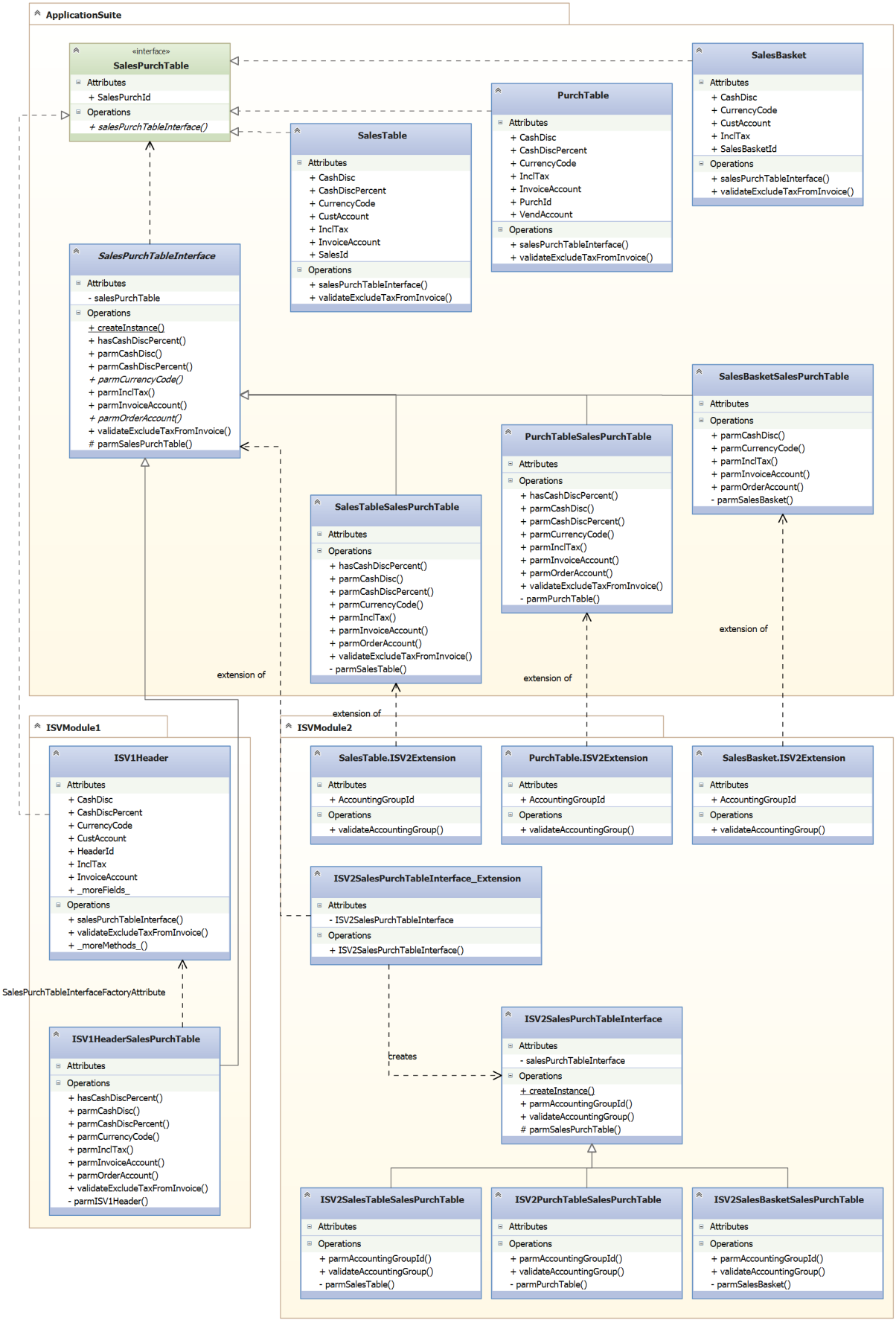
Even though the table map methods have been made obsolete, the corresponding methods still exist on the implementing tables. The logic in these methods have been refactored from delegating calls to the method on the table map, to the corresponding method on the base class of the hierarchy.

Extension scenario

In this example, if you want your ISVModule2 model to extend the interface class hierarchy and the tables implementing the **SalesPurchTable** table map, you must perform the following steps:

1. Add the fields and methods through table extension to the necessary tables implementing the **SalesPurchTable** table map.
2. Create a new interface class hierarchy, which exposes the new fields as parm-methods and other additional methods. The base class of the class hierarchy must be concrete and not abstract.
3. Create derived classes for each table that has been extended.
    a. Decorate each derived class with the **SalesPurchTableInterfaceFactory** attribute class to associate the class with the correct table.
    b. Create a static factory method on the base class of the new class hierarchy. The factory method should instantiate the proper derived class that leverages the **SalesPurchTableInterfaceFactory** attribute. If no derived class can be found, then an instance of the base class must be returned.
4. Create an extension class of the **SalesPurchTableInterface** class. The class augments the **SalesPurchTableInterface** class with a method that creates an instance from the new class hierarchy by calling the factory method on the new class hierarchy.

The class and extensions described above are shown in the following diagram.



The diagram contains an ISVModule1 model, which includes the **ISV1Header** table that implements the **SalesPurchTable** table map and contains its own **SalesPurchTableInterface** derived class. The model is independent of the ISVModule2, so when logic in the ISVModule2 creates an instance from the **ISV2SalesPurchTableInterface** class hierarchy, then an instance of the base class will be returned when the **SalesPurchTable** record is of type **ISV1Header**. If the methods on the base class return a reasonable result for unknown tables, then the two ISV models will co-exist within the same installation.

Code example

The following code example demonstrates a way to extend table maps.

```
[ExtensionOf(classStr(SalesPurchTableInterface))]
final public class ISV2SalesPurchTableInterface_Extension
{
    private ISV2SalesPurchTableInterface ISV2SalesPurchTableInterface;

    public ISV2SalesPurchTableInterface ISV2SalesPurchTableInterface()
    {
        if (!ISV2SalesPurchTableInterface)
        {
            ISV2SalesPurchTableInterface = ISV2SalesPurchTableInterface::createInstance(this);
        }

        return ISV2SalesPurchTableInterface;
    }

}

public class ISV2SalesPurchTableInterface
{

    SalesPurchTableInterface salesPurchTableInterface;

    private void initializeSalesPurchTableInterface(SalesPurchTableInterface _salesPurchTableInterface)
    {
        salesPurchTableInterface = _salesPurchTableInterface;
    }

    public SalesPurchTable parmSalesPurchTable()
    {
        return salesPurchTableInterface.parmSalesPurchTable();
    }

    protected void new()
    {
    }

    public static ISV2SalesPurchTableInterface createInstance(SalesPurchTableInterface
_salesPurchTableInterface)
    {
        SalesPurchTableInterfaceFactoryAttribute attr =
  new SalesPurchTableInterfaceFactoryAttribute(
   tableId2Name(_salesPurchTableInterface.parmSalesPurchTable().tableId));

        ISV2SalesPurchTableInterface instance =
  SysExtensionAppClassFactory::getClassFromSysAttribute(classStr(ISV2SalesPurchTableInterface), attr)
  as ISV2SalesPurchTableInterface;

        instance.initializeSalesPurchTableInterface(_salesPurchTableInterface);

        return instance;
    }

    public AccountingGroupId parmAccountingGroupId()
    {
        return '';
    }

}
[SalesPurchTableInterfaceFactoryAttribute(tableStr(SalesTable))]
public class ISV2SalesTableSalesPurchTable extends ISV2SalesPurchTableInterface
{
    private SalesTable parmSalesTable()
    {
```

```
        return this.parmSalesPurchTable();
    }

    public AccountingGroupId parmAccountingGroupId()
    {
        return this.parmSalesTable().AccountingGroupId;
    }

}

[SalesPurchTableInterfaceFactoryAttribute(tableStr(PurchTable))]
public class ISV2PurchTableSalesPurchTable extends ISV2SalesPurchTableInterface
{

    private PurchTable parmPurchTable()
    {
        return this.parmSalesPurchTable();
    }

    public AccountingGroupId parmAccountingGroupId()
    {
        return this.parmPurchTable().AccountingGroupId;
    }

}
```

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

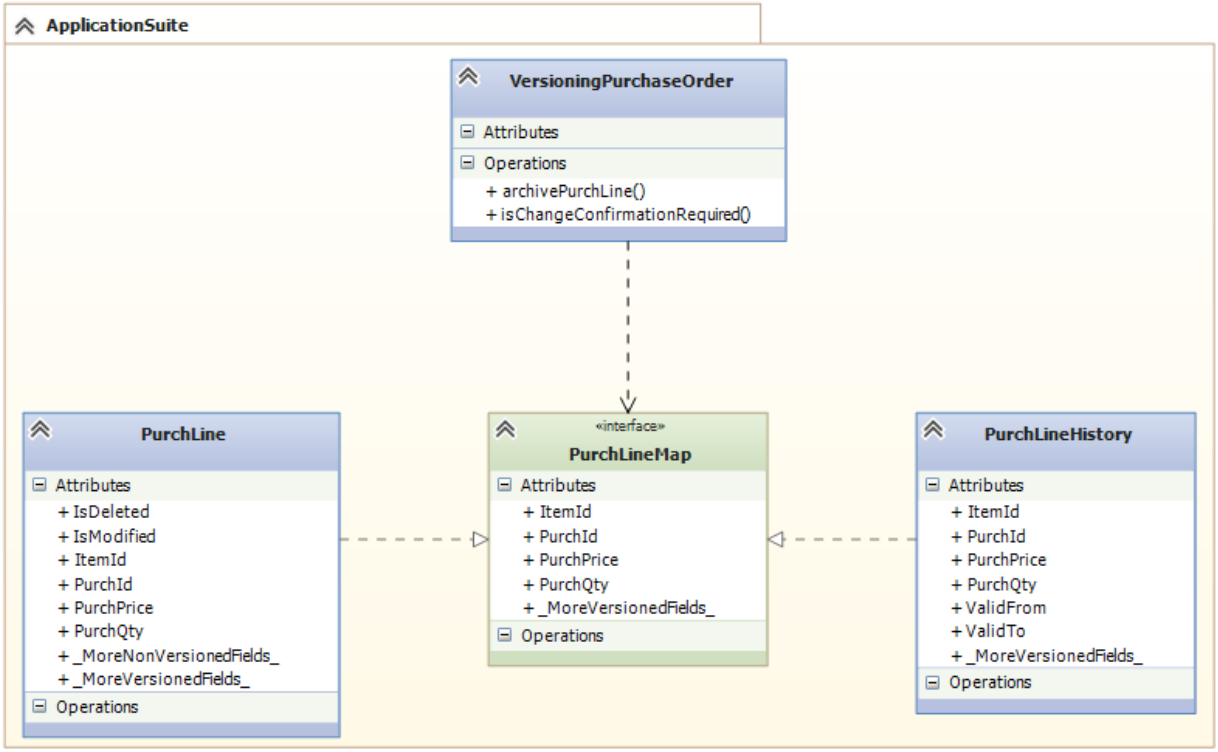
The survey will take about seven minutes. No personal data is collected (privacy statement).

Extend table maps that are used for versioning

2/18/2021 • 2 minutes to read • Edit Online

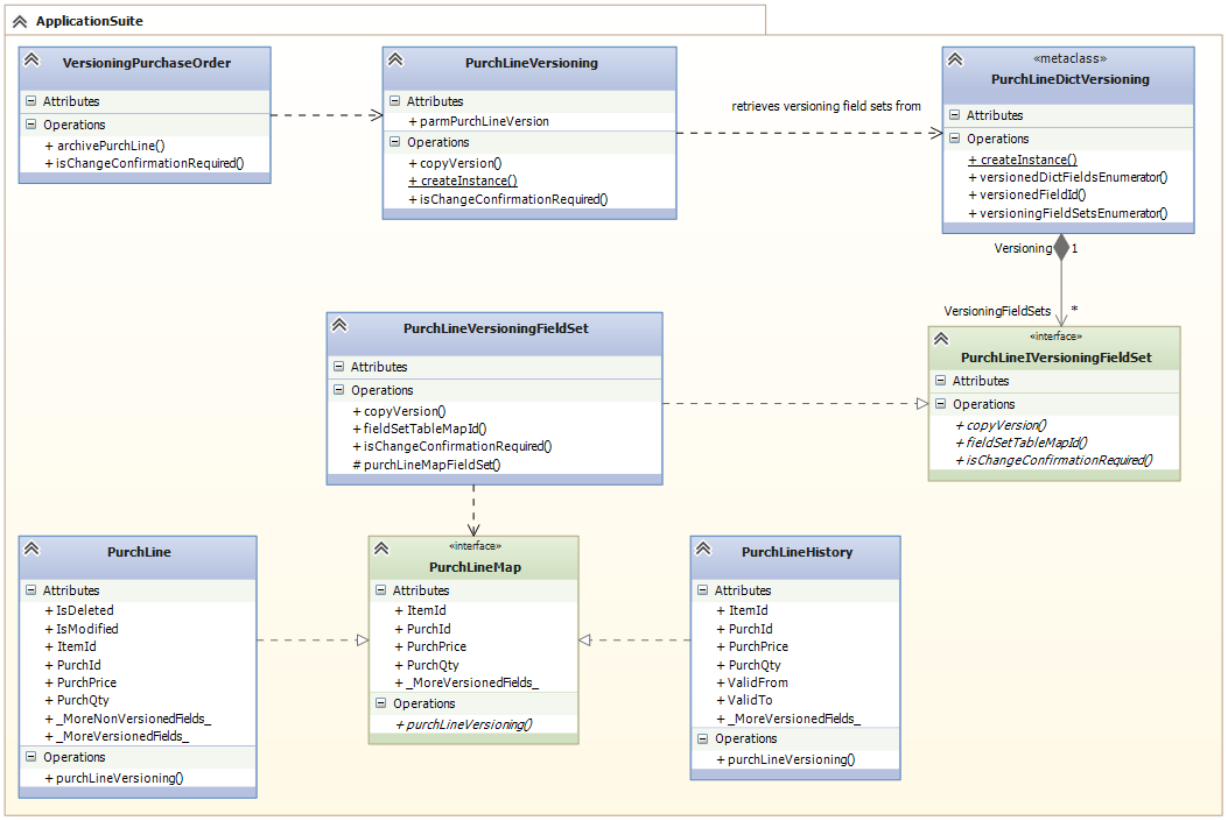
PurchLineMap table map logic

When new fields are added to the **PurchLine** and **PurchLineHistory** tables using table extensions, the new fields must be copied between the tables when a purchase order is versioned. The **PurchLineMap** table map specifies the fields that must be copied between the **PurchLine** table and the **PurchLineHistory** table when a new purchase order version is created or edited. To accomplish this, extend the **PurchLineMap** map table to include the additional fields. Additionally, the **PurchLineMap** is used by the **VersioningPurchaseOrder** class when archiving purchase order lines. The model is shown in the following diagram.



To be able to specify new fields to be copied, the **PurchLineMap** table map logic and its usage have been refactored. The copy logic has been moved to the **PurchLineVersioning** class, so the **VersioningPurchaseOrder** class references the **PurchLineVersioning** class instead of the **PurchLineMap** table map. The **PurchLineVersioning** class delegates the logic to copy the fields and the logic to determine whether a confirmation is required from the classes that implement the **PurchLineIVersioningFieldSet** interface. Each class that implements the interface is associated with a table map that specifies the fields to copy.

The **PurchLineDictVersioning** class instantiates the **PurchLineIVersioningFieldSet** object using reflection. The **PurchLineDictVersioning** class collects the entire set of fields which need to be copied. The field data is collected based on all the table maps associated with a class that implements **PurchLineIVersioningFieldSet**. The following diagram displays the new classes and their dependencies.



How to extend PurchLine and PurchLineHistory tables with new fields

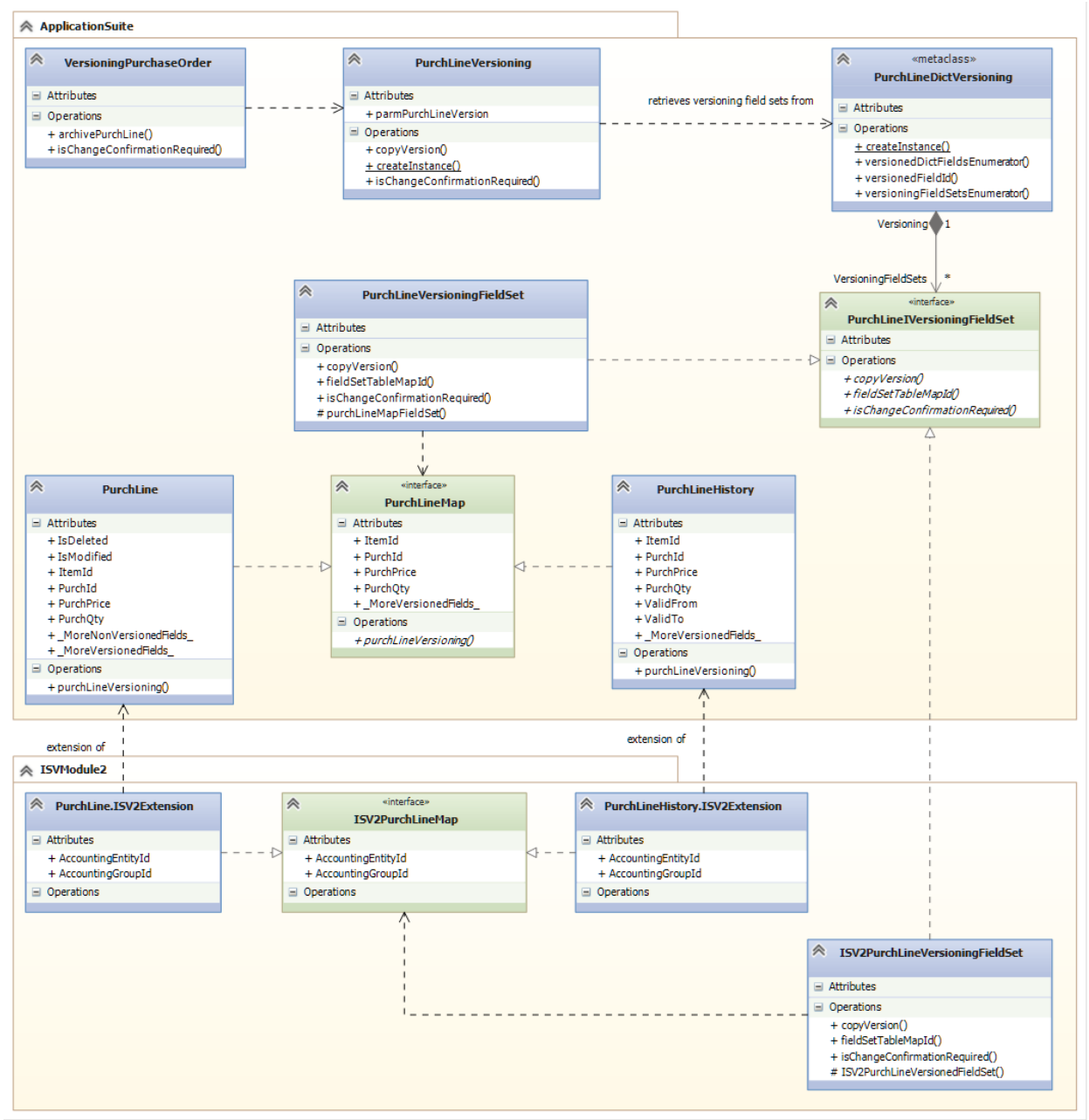
Suppose that you want the ISVModule2 model to extend the **PurchLine** and **PurchLineHistory** tables with new fields that must be copied when creating a new version of a purchase order.

> **NOTE**
>
> You must have developer access to the ISVModule2 model.

To complete this task, you must perform the following steps:

1. Add fields by using table extensions to the **PurchLine** and **PurchLineHistory** tables.
2. Create a new table map containing the fields that must be copied, and implement the new table map on the two new table extensions.
3. Create a new class to implement the **PurchLineIVersioningFieldSet** interface and implement the following required methods.
   - **copyVersion** method - Copies data between two records of the new table map type.
   - **fieldSetTableMapId** method - Returns the ID of the new table map.
   - **isChangeConfirmationRequired** method - Returns true or false based on whether the change to the newly added field values requires a confirmation to be created.

The classes, interfaces, and extensions described in these steps are shown in the following diagram.



Extending decimal point precision for selected data types

2/18/2021 • 5 minutes to read • Edit Online

This topic describes how to extend decimal point precision for selected data types. You can create extensions of specific extended data types of the type Real, to change the decimal point precision for certain scenarios. To change the decimal point precision, change the **NoOfDecimals** property as needed.

Extended data types are hierarchical and inherit behavior from the data type they extend. When changing the number of decimals for one extended data type, the number of decimals on all derived extended data types will follow. In other words, if you find an extended data type where **NoOfDecimalsIsExtensible** is false, then check the parent extended data type, as the number of decimals might be extensible in this wider scope.

Weight

Weight data can be maintained with a maximum of two decimals by default. If you require the ability to enter, maintain, and view weight data with a maximum precision of six decimals, you must extend the decimal point precision for the **WeightBase** extended data type.

Product width, height and depth

These physical dimensions can be maintained with a maximum of two decimals by default. If you require the ability to enter, maintain, and view this data with a maximum precision of six decimals, you must extend the decimal point precision for the **InventWidth**, **InventHeight**, and **InventDepth** extended data types respectively.

Product quantity

Quantity data that is related to the procuring, consuming, producing, storing, and selling of products can be maintained with a maximum of two decimals by default. If you require the ability to enter, maintain, and view product quantities with a maximum precision of six decimals points, you must extend the decimal point precision of the **ProductQuantity**, **CostQuantity**, and **CAMMagnitude** extended data types.

Bill of materials, formulas, and production orders allow maintaining quantities with four decimals by default. If you require more than four decimals, extend the decimal point precision for the **BOMProductQuantity** extended data type.

Related data types

Price unit, **Price quantity**, and **Charge quantity data** can be extended independently from product quantities. You can extend the **PriceUnit** extended data type to change the decimal point precision to a value other than the default two for price units.

You can extend the **PriceQty** extended data type to change the decimal point precision to a value other than the default two for price and charge quantities.

Overloaded data types

There are two extended data types that are used for storing both quantity data and other types of data. These data types must be extended separately.

The **AmountQty** extended data type is used for storing and presenting both amounts and quantities. The

AmountQty extended data type should be extended to the maximum amount of required decimals for both amounts and quantities. For example, if amounts need to be maintained with three decimal points, but quantities still need to be maintained with two decimal points, then the data type should be extended to three decimal points.

The **ProductQuantityHourValue** extended data type is used for storing and presenting both hours and quantities. The **ProductQuantityHourValue** extended data type should be extended to the maximum amount of required decimal points for both hours and quantities. For example, if quantities need to be maintained with four decimal points, but hours still need to be maintained with two decimal points, then the data type should be extended to four decimal points.

Unit amounts

By default, unit amounts including prices, line discount amounts, and line charge amounts can be maintained with a maximum of two decimals.

If you require the ability enter, maintain, and view unit amounts with a maximum precision of six decimal points, you must extend the decimal point precision of the **UnitAmountCur**, **UnitAmountMST**, and **CostPriceNonMonetary** extended data types.

If you require a decimal point precision of more than four, you should also extend the PriceRoundOff extended data type.

Overloaded data types

There are five extended data types that are used for storing both unit amount data and other types of data.

The **PriceDiscAmount** extended data type is used for storing and presenting amounts and unit amounts. The **PriceDiscAmount** extended data type should be extended to the maximum amount of required decimals points for both amounts and unit amounts. For example, if amounts need to be maintained with three decimal points, but unit amounts need to be maintained with four decimal points, the data type should be extended to four decimal points.

The **MCRRoyaltyValue**, **PdsRebateValue**, **TAMRebateValue**, and **MarkupValue** extended data types are used for storing and presenting amounts, unit amounts, and percentages. The extended data types should be extended to the maximum amount of required decimal points for amounts, unit amounts, and percentages. For example, if amounts need to be maintained with three decimal points, but unit amounts need to be maintained with four decimal points and percentages should remain maintained with two decimal points, then the data type should be extended to four decimal points.

Amounts

Amounts, including unit amounts, can be maintained with a maximum of two decimals by default.

If you require the ability to enter, maintain, and view amounts including unit amounts with a precision of maximum six decimal points, you must extend the decimal point precision of the **Amount**, **AmountMST**, and **CostAmountNonMonetary** extended data types. If you require a different precision for unit amounts other than for amount, follow the description for how to extend the decimal point precision for unit amounts.

Overloaded data types

There are three extended data types that are used for storing amount data and other types of data. This means that they must be extended separately.

The **AmountQty** extended data type is used for storing and presenting amounts and quantities. The **AmountQty** extended data type should be extended to the maximum amount of required decimals for both amounts and quantities. For example, if amounts need to be maintained with three decimal points, but quantities still need to be maintained with two, then the data type should be extended to three decimal points.

The **PriceDiscAmount** extended data type is used for storing and presenting amounts and unit amounts. The **PriceDiscAmount** extended data type should be extended to the maximum amount of required decimals points for amounts and unit amounts. For example, if amounts need to be maintained with three decimal points, but unit amounts need to be maintained with four decimal points, then the data type should be extended to four decimal points.

The **MarkupValue** extended data type is used for storing and presenting amounts, unit amounts, and percentages. The extended data types should be extended to the maximum amount of required decimals points for amounts, unit amounts, and percentages. For example, if amounts need to be maintained with three decimal points, unit amounts need to be maintained with four decimal points, and percentages should remain with two decimal points, then the data type should be extended to four decimal points.

Write extensible code

2/18/2021 • 4 minutes to read • Edit Online

X++ and the metadata model provide a powerful foundation for building business solutions. One of the design pillars is to automate as many technical concerns as possible, so that the engineer can focus on the business domain. For example, if you put all the text resources in label files, one technical concern that you don't have to worry about is the localization of text resources.

Extensibility is another technical concern. You want other people to be able to extend your solution in a safe, robust, and maintainable way. By default, your solution is highly extensible. However, there are a few guidelines that you should follow to help guarantee completeness.

Responsibilities

Any Finance and Operations environment runs a business solution that includes components from many sources. Typically, each solution has code from Microsoft, independent software vendors (ISVs) and partners, and also internally developed code. Each contributor is responsible for its own contribution to the solution and for the way that its contribution interacts with other contributions.

When you write extensible code, you invite other people to interact with your solution. Your responsibility is to enable other people to be good guests. Here is how you meet this responsibility:

- **Make robust extension points** – Extension points are the foundation of extenders' solutions. They must be well-defined and robust from release to release.
- **Invite side-by-side customizations** – Recognize that multiple extenders might use the same extension point. Enable 1:n (one-to-many) interactions instead of 1:1 (one-to-one) interactions.
- **Trust that extenders will be well-behaved** – All responsible parties share the same goal: to create great, lasting solutions for the customer. When you create extension points, you give up control and share the responsibility with other people. Assume that extenders will be cautious and use your extension points as they were intended.

Proven principles

All the good engineering practices that you're already using still apply. Everything that you've learned still applies. You don't have to learn new principles or unlearn old practices. This topic is just highlighting three principles of software craftmanship that have been sought and taught for decades. These principles not only make your code easier to read, maintain, test, review, and refactor, but also make your code easier to extend. Apply and advocate these principles.

Extend code by using the SOLID principles

SOLID is an acronym for five principles that you can use to make your code easier to extend:

- **Single responsibility** – Classes and method should have a single responsibility and should not have side-effects. By following this principle, you help guarantee that extension points that are automatically created on public and protected methods will be great extension points.
- **Open/closed**
  - **Open for extension** – Open your solution for extensions by designing and considering the extension surface. After an extension point is made available, you're responsible for maintaining it. This responsibility adds significant restrictions to future development. It's often preferable to open a

solution up for extension by demand. For example, use internal methods over public methods or private methods over protected methods.

- **Closed for modification** – Make your properties private, and make your methods either private or final-protected. In this way, no one can take advantage of a dependency on your logic, either through inheritance or extension.

- **Liskov substitution** – Derived classes must be able to be substituted for their base classes. For example, this substitution can be done by providing factories, by using SysExtension, and by using simple construct methods.

- **Interface segregation** – Create concise interfaces. This principle lets extenders provide replacement implementations and is particularly valuable when it's used together with the next SOLID principle, dependency inversion.

- **Dependency inversion** – Depend on abstractions, not concretions. This principle enables decoupling and lets extenders provide concrete instances that conform to the abstraction that your logic depends on.

Write clean code

Clean code can be read like an article. The name of a method provides the heading of the article. The body of the method comes next and really consists of just a few lines of summary. This summary calls a few other methods that have good descriptive names. In this way, the reader can keep exploring details and can also stop at any time without missing any conceptual information.

When code is written like in this manner, methods are short, often less than 5 to 10 code lines. Additionally, the number of parameters is low, often less than two, and the conditions and blocks of code are always a single code line.

Here is an example.

```
public void processOrder(SalesOrder _salesOrder)
    {
        if (this.approveOrder(_salesOrder))
        {
            this.confirmOrder(_salesOrder);
        }
        else
        {
            this.rejectOrder(_salesOrder);
        }
    }
```

In X++, every protected and public method is an extension point. By writing clean code, you automatically produce extensible code. In the previous example, an extender can change how approval, confirmation, and rejection are implemented. If the implementations had been inline, the code would not be extensible.

Don't repeat yourself (DRY)

To help prevent misalignment of implementations, avoid redundancy in your logic. This principle is especially important in the case of extensible code, because the extender might not extend all required pieces and might therefore unintentionally leave the solution broken.

Best practices to create an extensible solution in X++

The following best practices can help you create extensible solutions in X++, so that consumers of your code can extend your solution:

- Classes
- Methods

- Forms
- Extended data types
- Extensible enums
- Delegates
- Tables
- Attributes that make methods extensible

Breaking changes

When you make your solution extensible, you also help guarantee that you won't break extension points later. For more information, see Breaking changes.

External resources

The following external resources can help you make sure that you're writing clean code:

- SOLID Principles
- Pluralsight - Clean Code: Writing Code for Humans
- Clean Code: A Handbook of Agile Software Craftsmanship
- Clean Coders
- Don't repeat yourself

Write extensible classes

2/18/2021 • 2 minutes to read • Edit Online

A class and its methods should have a single responsibility. Keep the following in mind in order to design classes that are resilient to changes in the long run. Class should have:

- A clear purpose
- Good names (class name and method names)
- Only methods that should be extended are exposed for extensibility, i.e. the key rule is allow extending as little as necessary.
  - Every public and protected method is an extensibility point in X++. Every time that a new method is introduced, downstream consumers get a new way to inject additional logic into the method.

Example

Non-extensible code

```
    void calculatePrice(SalesLine _saleLine, AmountMST _amount)
    {
        // cannot add extra condition if needed
        if(_saleLine.QtyOrdered > 0 && _saleLine.SalesType == SalesType::Sales)
        {
            ttsbegin;
            // calculation of SalesPrice is locked and cannot be extended
            _saleLine.SalesPrice = _saleLine.QtyOrdered * _amount;
            _saleLine.update();
            ttscommit;
        }
    }
```

Extensible code

```
protected boolean canUpdateSalesPrice(SalesLine _saleLine)
   {
       return (_saleLine.QtyOrdered > 0 &&
   _saleLine.SalesType == SalesType::Sales);
   }

   protected SalesPrice calculateSalesPrice(
   SalesLine _saleLine, AmountMST _amount)
   {
       return _saleLine.QtyOrdered * _amount;
   }

   public void updateSalesPrice(SalesLine _saleLine, AmountMST _amount)
   {
       // extra condition can be added in CoC on the method
       if(this.canUpdateSalesPrice(_saleLine))
       {
           ttsbegin;
           // extra calculation/value can be added in CoC on the method
           _saleLine.SalesPrice = this.calculateSalesPrice(_saleLine, _amount);
           _saleLine.update();
           ttscommit;
       }
   }
```

Class hierarchies

For new or existing class hierarchies where a factory pattern can be used, the SysExtension framework enables easy extensions. Because these extensions are truly decoupled, new subclasses can be added without requiring any changes to the base class. Therefore, less code is required. Additionally, because the contract of the construct remains the same, there is no change to the public application programming interface (API). Therefore, refactoring involves low risk.

Some existing factory methods might not instantiate subclasses by using the instance constructor. Instead, they might call a static constructor, such as **construct**. If the SysExtension framework is used for these factory methods, a breaking change occurs, because the static constructors on the subclasses are no longer invoked. In these situations, use the SysExtension framework only for the default case.

For more information about class hierarchies, see the following blog posts:

- SysExtension Framework – to the rescue
- Embrace the extensions mindset with Dynamics 365 for Finance and Operations #2 – SysExtension framework

Deprecation

If a class or a public or protected method is no longer required, always use a warning first to notify consumers that the method is obsolete. Then, when all consumers have had the chance to uptake the changes or the new API, the method can be deprecated. Deprecation of classes and methods (or removal of class members in other cases) is a breaking change. For more information, see Breaking changes.

Write extensible methods

2/18/2021 • 5 minutes to read • Edit Online

Before you make a method extensible, you should assess the exposed functionality of the method and the impact that the extensions might have on the scenario where the method is used. For example, depending on the business scenario, there is low risk if you enable extensions to initialize a table record but high risk if you enable extensions to skip a specific validation. You might also want to consider the impact if the method is extended in parallel with other extensions.

After you've made a method extensible, future modifications to the method are restricted because of the potential user impact if the method signature or logic is changed.

Here are some guidelines to follow when you write extensible code:

- **Write short and concise methods** – A method should have only one responsibility. This approach enables easy extensions of the method, where the extensions can act only on the specific responsibility of the method. As a simple example, keep the construction and initialization of a class object in two separate methods.
- **Expose only what is necessary** – For any new class members or methods that are added, 'Keep any new class members or methods that you add private, to allow minimal access to them.
- **Use private, protected, public, and final explicitly** – For methods and class fields, this approach will guide any extenders of your code to your extension points but still let you keep full control of the parts that the extenders should not care about or depend on.
- **Method parameters**
  - The method is most likely long and should be refactored. Consider whether you should refactor the whole method into a class or split the method into smaller methods that require fewer parameters.
  - In other cases, when several parameters are required, the parameters often have a coherence that can be expressed by a class. By encapsulating these parameters in a class, you make it easy for extenders to add additional parameters to the base method, without breaking application programming interfaces (APIs) later.
- **Switch blocks**
  - Avoid switch blocks in the middle of methods. A switch block should be in its **own method** to enable it to be extended.
  - **Long case blocks** are good candidates for being refactored into a class/class hierarchy that has a subclass for each case block. For an example, see the **SalesLineCopyFromSource** class hierarchy.
  - Avoid **default blocks** in switch statements, because they make the method that has the switch block non-extensible.
  - Avoid **throw statements in the default block** of a switch statement, because they make the switch statement non-extensible. One way to handle the throw in the default case is to refactor the switch block to a separate method that is extensible. Alternatively, you can make the whole method replaceable.

    In the following example, **findOrderHeader** is replaceable.

```
private Common findOrderHeader(boolean _forUpdate)
  {
      switch (this.InventTransType)
      {
          case InventTransType::Sales:
              return this.salesTable(_forUpdate);

          default:
              return this.findOrderHeaderDefault(_forUpdate);
      }
  }

  [Replaceable]
  protected Common findOrderHeaderDefault(boolean _forUpdate)
  {
      throw error(Error::wrongUseOfFunction(funcName()));
  }
```

- **While** – Avoid **while** blocks in the middle of methods, because it becomes more difficult to extend the **while** blocks. Ideally, logic in a **while** block should be in a separate method that enables extensions.

  Refactoring logic within a while loop

```
        while select forupdate markupTransOrig
                        where markupTransOrig.TransTableId == origLineUpdate.TableId
                            && markupTransOrig.TransRecId == origLineUpdate.RecId
        {
            markupTransNew.clear();
            markupTransNew.initFromSalesLine(_newLine);
            markupTransNew.initFromMarkupTrans(markupTransOrig);

            markupTransNew.LineNum        = MarkupTrans::lastLineNum(_newLine.TableId, _newLine.RecId) + 1;

            markupTransNew.OrigRecId = origLineUpdate.RecId;
            markupTransNew.OrigTableId = origLineUpdate.TableId;
            markupTransNew = this.initializeNewMarkupTrans(_origLine, _newLine, markupTransOrig);

            if (markupTransOrig.MarkupCategory == MarkupCategory::Fixed)
            {
                markupTransNew.Value = (_newLine.ExpectedRetQty/_origLine.ExpectedRetQty) * markupTransOrig.Value;
                markupTransNew.Value = CurrencyExchangeHelper::price(markupTransNew.Value, markupTransNew.CurrencyCode);
                markupTransOrig.Value -= markupTransNew.Value;
                markupTransOrig.update();
            }
            else
            {
                markupTransNew.Value = markupTransOrig.Value;
            }
```

  Extensible method after refactoring

```
protected MarkupTrans initializeNewMarkupTrans(SalesLine _origLine, SalesLine _newLine, MarkupTrans _markupTransOrig)
{
    MarkupTrans markupTransNew;
    markupTransNew.clear();
    markupTransNew.initFromSalesLine(_newLine);
    markupTransNew.initFromMarkupTrans(_markupTransOrig);

    markupTransNew.LineNum        = MarkupTrans::lastLineNum(_newLine.TableId, _newLine.RecId) + 1;

    markupTransNew.OrigRecId = _markupTransOrig.TransRecId;
    markupTransNew.OrigTableId = _markupTransOrig.TransTableId;

    if (_markupTransOrig.MarkupCategory == MarkupCategory::Fixed)
    {
        markupTransNew.Value = (_newLine.ExpectedRetQty/_origLine.ExpectedRetQty) * _markupTransOrig.Value;
        markupTransNew.Value = CurrencyExchangeHelper::price(markupTransNew.Value, markupTransNew.CurrencyCode);
        _markupTransOrig.Value -= markupTransNew.Value;
    }
    else
    {
        markupTransNew.Value = _markupTransOrig.Value;
    }

    return markupTransNew;
}
```

- If..else statements

- To enable extension of the conditions in an if statement, extract the logic in the if condition into a separate method.
- Avoid nested if..else blocks, because they make it difficult to change the logic in one of the blocks. One way to resolve this issue is to refactor each condition and the logic in each block into a separate method. In this way, you can extend the conditions or the logic in each block.
- When the if..else blocks handle specialization, consider moving the logic into a class hierarchy. For an example, see **SalesLineCopyFromSource**.
- In some scenarios, a throw in an 'else' block of a method (when the method only has an if..else) makes the method non-extensible. One way to handle the throw in the else is to refactor the conditions for the throw into a separate method.

- **Avoid using PrmIsDefault** – When the method is overridden or wrappable, the caller of **super()** or **next()** provides all parameters. Therefore, **prmIsDefault()** always returns false.

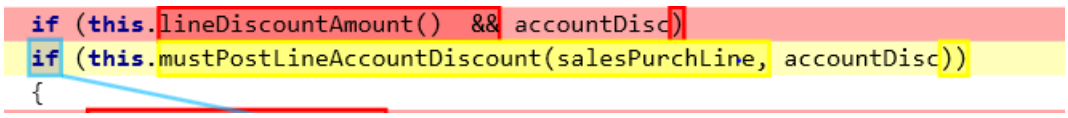
- **Avoid using enumCnt** – At compile time, this method uses a numeric literal of the number of values that an enum has. If the enum is extended or made extensible later, your code will have to be recompiled. Use **DictEnum.values()** instead.

- **Construct methods**
  - Use the **SysExtension** framework to enable easy extensions.
  - Avoid a throw in factory methods. One way to resolve this issue is to extract the conditions for the throw into a separate method that is extensible. For more details, see the guidelines for throw statements later in this list.
- **Static methods** – Static methods can't be extended with extra state. For example, a method extender can introduce properties that can be set by using parameter methods. Use instance methods instead, whenever this approach is possible.

- **Ability to extend part of the logic in a long method** – If it isn't possible to refactor a whole method, but the goal is to make part of the method extensible, apply the extract method refactoring. The new protected method must have a single responsibility, and it must also have a name that conceptually and precisely describes that responsibility. In this way, owners and all extenders can use the method without breaking each other. For example, initialization, insertion, updates to a table record, or instantiation and initialization of a class can be extracted into smaller methods, and each of these smaller methods can be enabled for for extensions. The original method then calls these individual methods. Therefore, the callers to this method aren't broken.

- **Throw statements** – A throw that is added to an existing method that is extensible could break extenders. Consider adding the conditions for the throw in an extensible method. In this way, extenders can take advantage of the method, and you can get rid of the throw.

  If condition refactored out to a protected method



```
if (this.lineDiscountAmount()  && accountDisc)
if (this.mustPostLineAccountDiscount(salesPurchLine, accountDisc))
{
```

  Extensible method after refactoring

```
/// <summary>
/// Determines if the line account discount must be posted.
/// </summary>
/// <param name = "_salesPurchLine">A <c>SalesPurchLine</c> record.</param>
/// <param name = "_accountDisc">A <c>LedgerDimensionDefaultAccount</c> value.</param>
/// <returns>true if the line account discount must be posted; otherwise, false.</returns>
protected boolean mustPostLineAccountDiscount(SalesPurchLine _salesPurchLine, LedgerDimensionDefaultAccount _accountDisc)
{
    return (this.lineDiscountAmount() && _accountDisc);
}
```

- **Create, read, update and delete (CRUD) statements**

- Use Query objects in scenarios where the queries should be extensible. Implement a protected method that builds the query. In addition, you might want to build several separate methods to add joined data sources, ranges, and selection fields. In this way, different parts of the query can be extended individually.
- Use **SysQueryInsertRecordSet** to convert insert_recordset to a query.
- Avoid field lists in select statements. In this way, you enable extenders to retrieve their additional fields without having to extend.
- Use the **in** keyword in query ranges to enable extenders to add more values to the query range. We recommend this approach especially for query ranges that have enum values.

Write extensible forms

2/18/2021 • 2 minutes to read • Edit Online

Methods on forms

- In general, the guidelines for writing extensible methods also apply to form methods.
- Chain of Command (CoC) gives access to the form's non-private members, which are the same as the non-private members for classes.
- CoC is enabled for nested classes. Therefore, methods that are defined in various levels on the form are extensible.

> **NOTE**
>
> One limitation of methods on forms, that for form data source methods only methods that are defined in the kernel are enabled for extensions, i.e. methods defined on the form data source are not extensible. This will be available in an upcoming Platform Update.

Field groups

Consider using field groups whenever possible. In this way, independent software vendors (ISVs) can add their fields for free when they extend a field group.

Form controls

Moving around form controls could potentially cause a break if the controls are made non-extensible by moving.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Extended data types

2/18/2021 • 2 minutes to read • Edit Online

Extended data types (EDTs) have a rich extension model that lets extenders change specific behaviors.

To provide an extensible solution, keep the following guidelines in mind when you work with EDTs.

Label/Help text

Labels and Help text properties can be changed by an extension, but only one value can remain. If multiple solutions change the label of the same EDT, the various labels are, in functional terms, mutually exclusive. Therefore, those labels can't all be installed on the same system.

String size

String size can be defined only on root EDTs. The system will use the largest value that is defined across the EDT and its extensions.

For derived EDTs, string size can't be changed by an extension, because the IS-A relationship between the EDTs will be broken.

Assignments to string EDTs will truncate the string to match the defined string size.

Extends

The **extends** property can't be changed by an extension. Any change that is made to this property after release will cause a breaking change. Therefore, you must make sure that the property is set correctly before release.

If you set this property, neither you nor extenders will be able to make changes to the string size later.

Avoid unnecessary dependencies. For example, don't extend generic EDTs such as Name and Description.

Number of decimals

The **Number of decimals** property can't be changed by an extension.

If you set this property to **True**, extenders can change the number of decimal places.

If you set this property to **True**, make sure that the following conditions are met:

- All truncation logic honors the number of decimal places that is specified on the EDT, so that no implicit or hardcoded rounding will occur.
- The value isn't assigned to other incompatible EDTs that don't correctly handle rounding.

Write extensible enums

2/18/2021 • 2 minutes to read • Edit Online

An enumeration (enum) is made extensible by setting the following enum properties:

- Is Extensible = True
- UseEnumValue = No

If you set these properties, downstream implementors can extend the enum with more elements. The values of the elements are determined at deployment time and won't be identical across systems. However, the following behavior is ensured:

- Data upgrade scripts aren't required. Enum values are persisted in the database, regardless of the enum that is extended. Therefore, when an enum is made extensible, the enum values that are used on any system will prevail.
- The first element in the enum gets a value of 0 (zero). Therefore, an extensible enum can still be used with the **not** operator. The only exception is when the first element of the enum had a non-zero value before the enum was made extensible.

Using extensible enums in code

Because enum values are no longer controlled by the developer, there is no certainty about the enum values. When you use extensible enums in code, remember that extensible enums can't be used in comparisons. For example, **MyEnum::Value1 > MyEnum::Value2**.

Also, look for any conversions between integers and enums. For example, modeled ranges in views and queries, and queries that are created from code by using comparisons, such as < and > or by using hardcoded integer values in comparisons.

When the model and all dependent models are compiled, the comparisons and conversions to integers will be detected by the compiler as errors.

Make sure that logic where the enum values are used is extracted in **smaller methods**. In that way, an extension that uses Chain of Command (CoC) can handle the enum values that are added.

For **construct** methods where the instantiation is based on enum values, replace switch blocks with **SysExtension** wherever such a replacement is possible. In other cases, make sure that the default block is extensible. For an example, see the **PurchRFQCaseCopying** class.

If the enum is used in **switch blocks**, avoid having default blocks that either have or don't have throws that aren't extensible. When there are **long switch case blocks** or **if...else blocks** for the enum values, consider creating a class hierarchy to handle specific logic that is related to the enum. For an example, see the **PriceGroupTypeTradeAgreementMapping** class hierarchy.

Use the **in** keyword for query ranges that use the enum values, and make the container that the **in** keyword uses extensible.

Potential issues

Some enums require the elements to have a certain order or value, and cannot be made extensible. This could be status enums, where the values represent a logical progressive sequence, like: Draft, Approved, Completed, or Archived. It could also be enums where the values must have a fixed integral value to match another artifact, like

another enum or a tabpage control's number.

Some enums have many elements. Enums support up to 250 elements. If your enum has many elements, such as more than 100, consider redesigning the solution instead of making the enum extensible. If the enum is extensible, then adding more elements in the future might break customers's combined solution as the addition might exceed the limit.

Delegates

2/18/2021 • 2 minutes to read • Edit Online

Although you can subscribe to existing delegates, don't create new delegates. The Chain of Command (CoC) provides a richer, more robust, and more concise extension mechanism that supersedes delegates.

Instead of creating new delegates, structure your code in small methods that have good names, as described in the guidelines for writing extensible methods.

If you decide to use delegates, consider ensuring no more than one response where applicable. For more information, see EventHandlerResult classes in request or response scenarios.

Write extensible tables

2/18/2021 • 2 minutes to read • Edit Online

Tables have a rich extension model that lets extenders add fields, field groups, indexes, relations, methods, and more.

Unique indexes

Unique indexes can't be changed by an extension. Unique indexes define a table constraint, and they often also define the key of the rows in the tables. You aren't allowed to change unique indexes, because such changes change the nature of the table. Therefore, there is a high risk that the changes will cause logical conflicts with future versions of the solution that defines the table, or with other solutions that consume the table.

Avoid unique indexes that are likely to be changed either now or in the future. For example, don't create a unique index on product dimensions such as color, size, style, and configuration. Instead, create a unique index on a distinct product variant, so that the index doesn't have to be changed if new product dimensions are added.

If you require an extensible uniqueness constraint on multiple columns, consider creating a hash of the column's values. For an example, see the NumberSequenceScope table.

Data events

Tables have many predefined data events that are automatically raised.

Avoid calling **doInsert()**, **doUpdate()**, and **doDelete()**. These methods prevent the data events from being raised and make your table harder to extend. Instead, call **insert()**, **update()**, and **delete()**.

Field groups

Always use field groups to group related fields, and to build forms and reports. By consistently using this approach, you enable the extension to surface additional fields in forms and on reports by extending the field group.

Attributes that make methods extensible

2/18/2021 • 3 minutes to read • Edit Online

This topic describes the various attributes that can be used to control extensibility capabilities for methods.

The following table provides an overview of the default support for extensibility and accessibility on methods. The table also provides guidance on the method signature changes.

	HOOKABLE	WRAPPABLE	REPLACEABLE	ACCESSIBILITY	SIGNATURE
private	No	N/A	N/A	Accessible from within class it is defined in.	Signature can be changed
protected internal	No	Yes, from Platform update 25 onward	No	Accessible from with the class it is defined, from derived classes, and from classes in the same model.	Signature must remain compatible
internal	No	No	No	Accessible in the same model.	Signature can be changed
protected	No	Yes, unless marked final	No	Accessible from with the class it is defined and from derived classes.	Signature must remain compatible
public	Yes	Yes, unless marked final	No	Accessible from within the class it is defined, derived classes, and other classes that have access to the defining class.	Signature must remain compatible

Hookable

If a method is hookable, extenders can subscribe to pre-events and post-events.

For public methods, you can opt out by adding **[Hookable(false)]** to the method.

You can opt in for private and protected methods by adding **[Hookable(true)]** to the method.

If a method is explicitly marked as **[Hookable(false)]**, then it is not wrappable.

Best practices when you write code

When a method is hookable, the compiler generates extra intermediate language (IL) code to enable the method as an extension point. Although the extra code has performance overhead, this overhead is negligible in most cases. However, for performance-critical methods, consider marking the method as non-hookable.

Wrappable

If a method is wrappable, extenders can wrap it by using Chain of Command (CoC). Extenders must call next, because they aren't allowed to break the CoC.

For protected and public methods, you can opt out by adding **[Wrappable(false)]** to the method.

You can't opt in for private methods.

Best practices when you write code

CoC resembles inheritance in many ways. Typically, if you want other people to be able to call your method but not change it, you mark the method as final. Consider marking these methods as non-wrappable or non-hookable.

Replaceable

If a method is replaceable, extenders can wrap it by using CoC, but they don't have to unconditionally call next. Although extenders can break the CoC, the expectation is that they will only conditionally break it. The compiler doesn't enforce calls to next.

To be replaceable, a method must also be wrappable.

For wrappable methods, you can opt in by adding **[Replaceable]** to the method.

Best practices when you write code

When a method is replaceable, it can be extended by using CoC, and the execution of next can be skipped. Before you enable a method to be replaceable, you should thoroughly assess the functional impact if an extender skips the execution of the method.

- **Do** make sure that methods that have **[Replaceable]** have XML documentation that describes the responsibility of the method.
- **Don't** use **[Replaceable]** to let consumers skip the replaced logic and do nothing.
- **Don't** use **[Replaceable]** for factory methods when **SysExtension** can be used instead.
- **Avoid** using **[Replaceable]** when the method changes databases or class state.
- **Avoid** using **[Replaceable]** if the method performs multiple operations and has multiple responsibilities. Instead, refactor the method into separate methods, each of which has a single responsibility, and consider which methods should actually be replaceable.
- **Consider** using **[Replaceable]** to solve transformations.

  Example: Enum conversion that uses a switch statement over enum values, where the default block has a throw.
- **Consider** using **[Replaceable]** to override lookups and jumprefs.

Best practices for extenders

- **Don't** write logic that has a different responsibility than the logic that is being replaced.
- **Do** call the base functionality (call next) when the replacement logic doesn't apply.
- **Avoid** replacing logic completely by not calling the base functionality (call next).

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Breaking changes

2/18/2021 • 4 minutes to read • Edit Online

When you make your solution extensible, you also help guarantee that you won't break the extension points later. A breaking change is any change that can break a consumer of your code.

This topic lists some of the types of changes that can break your code.

> **IMPORTANT**
> This list isn't exhaustive. Other types of changes that aren't listed here could also be breaking changes.

Data model changes

General

If any data model change requires that a data upgrade script be run, consumers might no longer be able to synchronize their data model, or they might lose access to data.

Data types

- **Changing an enumeration (enum) from extensible to non-extensible** – Consumers might have extensions to the enum.
- **Changing an enum from non-extensible to extensible** – Consumers might be using the enums in comparisons. For more details, see Write extensible enums.
- **Decreasing the decimal precision of an extended data type (EDT) of the real type** – Consumers might have dependencies on the ability to enter data that uses the precision.
- **Decreasing the size of an EDT of the string type** – Consumers might have dependencies on the size of the string.
- **Specializing the EDT by making it extend another EDT** – Consumers might have string length or decimal precision extensions to the EDT.
- **Changing the enum type of an EDT of the enum type when the enum is extensible** – Consumers might have extensions to the enum.

Data model

- **Making a table obsolete and stopping information from being entered in the table** – Consumers might have a dependency on the table that information is entered in.
- **Making a table field obsolete and stopping information from being entered in the field** – Consumers might have a dependency on the field that information is entered in.
- **Renaming a field group** – Consumers might have extensions to field group, or code or metadata dependencies to it.
- **Changing constraints or indexes**
- **Making a table map obsolete and stopping the table map from being used**
- **Make a table map field obsolete**
- **Adding a table map field**
- **Removing a table map field mapping**
- **Making a data entity obsolete**
- **Making a data entity field obsolete**

Code changes

Class members

- **Deleting or renaming public or protected class-level members** – Consumers might be using these members in the extension classes.
- **Changing a class member from public or protected to protected or private** – Consumers might have queried or assigned values to the field.

Classes and interfaces

- **Adding an abstract method to a class** – Consumers might have created a derived type.
- **Adding final to a class** – Consumers might have created a derived type.
- **Adding a method to an interface** – Consumers might have implemented the interface on their own type.
- **Making a public class obsolete and stopping instantiation of the class** – Consumers might have overridden, wrapped, or subscribed to the instance methods.

Methods

- **Changing the access modifier from protected or public to another access modifier** – Consumers might have called, overridden, wrapped, or subscribed to the method.
- **Making a concrete public or protected method abstract** – Consumers might have subclasses to the class, and those subclasses might not have implemented the method, or they might even call super.
- **Renaming method parameters** – Consumers might have dependencies on parameters by name (via arguments or externally from C#, for example).
- **Adding, removing, or changing the type of a method parameter on a protected or public method** – Consumers might have called, overridden, wrapped, or subscribed to the method.
- **Adding or removing a default method parameter on a protected or public method** – Consumers might have wrapped or subscribed to the method.
- **Changing the return type of a method** – Consumers might have called, overridden, wrapped, or subscribed to the method.
- **Adding Hookable(false) to a protected or public method** – Consumers might have wrapped or subscribed to the method.
- **Adding Wrappable(false) to a protected or public method** – Consumers might have wrapped the method.
- **Removing Hookable(true) from a private or protected method** – Consumers might have subscribed to the method.
- **Removing Wrappable(true) from a private method** – Consumers might have wrapped the method.
- **Removing Replaceable(true) from a method** – Consumers might have conditionally wrapped the method.
- **Adding final to a protected or public method** – Consumers might have overridden, wrapped, or subscribed to the method.
- **Changing a method from instance to static or from static to instance** – Consumers might have called, overridden, wrapped, or subscribed to the method.
- **Making a method obsolete and stopping invocation of the method** – Consumers might have called, overridden, wrapped, or subscribed to the method.
- **Changing the responsibility of a method** – Consumers might have called, overridden, wrapped, or subscribed to the method.
- **Removing the reference to a method** – Consumers might have overridden, wrapped, or subscribed to the method, and they might expect their logic to run.

Delegates

- **Making any change in signature** – Consumers might have subscribed dynamically.

- **Removing the reference to a delegate** – Consumers might have subscribed, and they might expect their logic to run.

Label changes

- **Modifying or deleting a label** – Consumers might be using the label in the current context of the label text and the parameters that were passed, and so on. We recommend that, going forward, you add new labels in the event of a label change.

Application element changes

- **Removing any element** – Consumers might have a compile time dependency on the existence of the element.

Metadata extensions

- **Not following the naming guidelines for metadata or augmentation classes** – Consumers might have elements that have the same name.

Compatibility checker tool

2/18/2021 • 4 minutes to read • Edit Online

The compatibility checker tool can detect metadata breaking changes against a specified baseline release or update. In this way, it helps ensure backward compatibility. Microsoft uses the tool to help ensure metadata compatibility.

The compatibility checker tool is available as one of the dev tools in Platform update 34. You can use it to ensure that your solutions are backward-compatible with earlier releases before you install or push updates to customers.

What the tool detects

The tool compares metadata of the current version with metadata of a baseline version. It detects and reports metadata changes that Microsoft has identified as breaking and added to the tool.

For a list of breaking changes that the tool detects, see the List of breaking changes detected by the tool section later in this topic.

> **NOTE**
>
> - The list in this topic doesn't include all the breaking changes that the tool can detect.
> - The tool doesn't detect all breaking changes.

What the tool doesn't detect

The tool detects only breaking changes that can be identified by comparing data. For example, it doesn't detect the following breaking changes that often occur:

- The reference to a protected or public method is removed.
- The responsibility of a method is changed.

Using the tool

You can use the tool to detect metadata compatibility issues that a new version has against the version that it's replacing. Microsoft uses the tool to detect any breaking changes that a new monthly update has against the previous monthly update.

Usage

```
CompatibilityChecker.exe -BaselineDirectory=\<Path to baseline metadata\> -CurrentDirectory=\<Path to
current metadata\> -ModuleName=\<Module name\> -OutputFile=\<Output file path\> -LogFile=\<Log file path\>
```

Example

```
CompatibilityChecker.exe -BaselineDirectory="\\servername\archive\Build1\BaselineMetadata" -
CurrentDirectory="E:\\MyCode\\retail\\amd64\\BaselineMetadata" -ModuleName="Directory"
-OutputFile="E:\\Logs\\Directory\\Diagnostics.xml" -LogFile="E:\\Logs\\Directory\\Checkerlog.txt"
```

Description

The tool identifies breaking changes by comparing the current metadata with specified baseline metadata.

You must specify the following paths:

- **BaselineDirectory** – The path of the baseline metadata.
- **CurrentDirectory** – The path of the current (new) metadata.
- **OutputFile** – The path of the file that contains the list of breaking changes.

The following rules apply:

- You must compile the current metadata before you run the tool.
- The *OutputFile* file contains the list of breaking changes that the tool identifies.
- The *BaselineDirectory* directory must exist and must have the metadata for the specified module and its dependencies (if the module has any dependencies).
- The metadata paths for *BaselineDirectory* and *CurrentDirectory* should have metadata for *StaticMetadata*. This metadata should be present in a folder that is named **StaticMetadata** in the specified paths.
- You can suppress any breaking change that the tool identifies by adding an entry to the model's ignore list. This file is present in the **AxIgnoreDiagnosticList** folder for the model.

List of breaking changes detected by the tool

> **NOTE**
> The tool identifies metadata compatibility changes as breaking changes only if they have been defined as breaking in the tool.

Class members

- **Changing the access modifier of protected or public class members (including making a member read-only)** – Consumers might have read from the field or assigned values to it.
- **Deleting or renaming public or protected class-level members** – Consumers might be using these members in some extension classes.

Methods

- **Changing the method signature of a protected or public method** – Wrappers and callers of the method will be broken.
- **Making a protected or public method obsolete** – Consumers might be wrapping or overriding the methods.

Classes and interfaces

- **Making a class final** – Consumers might have created a derived type.
- **Making a class abstract** – Consumers might be instantiating the class.
- **Adding an abstract method to a class** – Consumers might have created a derived type.
- **Adding a method to an interface** – Consumers might have implemented the interface on their own type.
- **Making a public class obsolete and stopping instantiation of the class** – Consumers might have overridden, wrapped, or subscribed to the instance methods.

Delegates

- **Any change in signature** – Consumers might have subscribed dynamically.

Tables

Any of the following changes will break table extensions and table references to tables and table fields:

- Deleting or renaming table fields, field groups, indexes, table mappings, or table relations

- Modifying these table properties: **Extends**, **SuportInheritance**, **TableType**, **SaveDataPerCompany.Yes**, or **SaveDataPerPartition**
- Modifying these table field properties: **ExtendedDataType**, **Scale**, or **String size**
- Modifying these table index properties: **AllowDuplicates.No** or **IndexType**

Forms

Any of the following changes will break form extensions that reference the controls or methods:

- Deleting or renaming form controls, form data sources, and form data source fields.
- All changes that are breaking for methods are also breaking for form methods.

Enumerations (enums)

- Modifying these properties: **IsExtensible** or **Value**

Extended data types (EDTs)

- Modifying these properties: **Extends**, **EnumType**, or **Scale**

Entities

- All changes that are breaking for tables are also breaking for entities.
- Renaming a public entity.

Labels

- **Modifying or deleting a label** – Consumers might be using the label in the current context of the label text and the parameters that were passed. We recommend that you add new labels instead of changing existing labels.

Application elements

- **Removing any element** – Consumers might have a compile-time dependency on the existence of the element.

Take traces by using Trace parser

2/18/2021 • 2 minutes to read • Edit Online

This tutorial provides guidelines on how to take traces.

In this tutorial, you'll take a tour of how to collect and download traces. The trace analysis tool works largely similar to the Microsoft Dynamics AX 2012 version, but it is not backward compatible and can't be used to analyze AX 2012 traces. The trace parser tool can be found in the PerfSDK folder on your development deployments.

Prerequisites

This tutorial requires that you access the environment as an administrator on the instance. The administrator can also grant rights to other users to take a trace. In this way, you can trace scenarios that can't be reproduced with administrative rights.

Capture the trace

1. Before you trace make sure your scenario is in a warm state, meaning that you have run the scenario you want to trace once before you take the trace. Being in a warm state prevents things like metadata loading and other possible warm-up tasks from being in the trace.
2. In the navigation bar, select **Help**, and then select **Trace**.
3. Name the trace that you are about to capture, and then select **Start trace**.
4. Perform actions that need to be analyzed, for example, opening **Accounts payable > Vendors > All vendors**.
5. When you are finished, select **Stop trace**. Then, you can select one of the following options (for this tutorial, select the second option):

   - **Download trace** – Store the captured trace on a local machine. You can analyze a downloaded trace with the desktop version of Trace Parser.

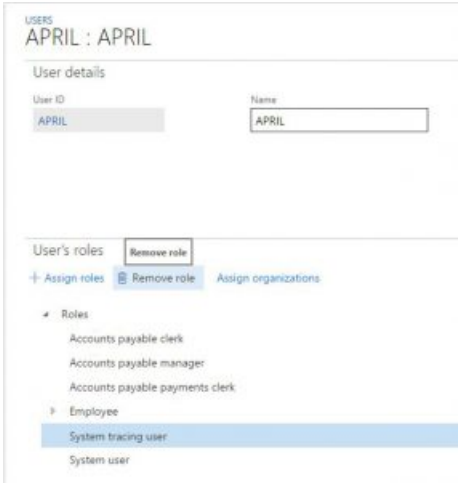
   > **NOTE**
   > If you download a trace it will not be available for later uploading.

   - **Upload trace** – Store the trace in the cloud for later downloading by, for example, the admin, it will be automatically deleted after 7 days and can also be deleted manually from the captured traces form.

> **NOTE**
> If your scenario takes more than 1-2 minutes it is better to try to take multiple smaller traces of 30 seconds each as the trace will likely get too big to be easily analyzed and there is a risk for losing data if the trace gets too big.

Assign trace rights to user

1. To give a user rights to capture a trace, go to **System administration > Users > Users**.
2. Select the user and assign the **System tracing user** role.



Remove the user role again once the user is done with tracing to avoid unwanted tracing.

Open captured trace

1. In the navigation bar, select **Help**, and then select **Trace**.
2. Select Captured traces.

> **NOTE**
> The captured traces button can only be seen by users with administrative rights.

3. Select the trace name to download to open and analyze it with the desktop version of trace parser or
4. Select the user name to get to the user options.
5. Delete the trace if you want. You might do delete the trace if you have downloaded it.

> **NOTE**
> The trace will be deleted after 7 days. For more information about the desktop version of trace parser, see Diagnose issues and analyze performance by using Trace parser.

> **NOTE**
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Single-user testing with Task recorder and the Performance SDK

2/18/2021 • 4 minutes to read • Edit Online

Use the information in this topic to do single-user testing by using Visual Studio and the Performance software development kit (SDK) together with a performance test script that is generated by using Task recorder.

> **IMPORTANT**
> Visual Studio 2019 will be the last version of Visual Studio with web performance and load test features. In the future, we will publish some recommendations for alternative solutions.
>
> - If you are using the Visual Studio and Test Controller/Test Agent for on-premises load testing, Visual Studio 2019 will be the last version. You can continue using it until the end of the support cycle.
> - If you are using the cloud-based load testing service, it will continue to run through March 31, 2020. Until then, you can continue to use all of the experiences powered by this service without interruption. Alternatively, you can switch to on-premises load testing. For more information, see Cloud-based load testing service end of life.

Prerequisites

To complete the steps in this topic, you must have a development environment that has Platform update 21 or later.

> **IMPORTANT**
> Your development must be in Platform Update for 10.0.11 or later if your Finance and Operations apps were deployed in 21Vianet.

Use Task recorder to define and record an end-to-end business scenario

Before you run a single-user test, you must work with your business team to define your end-to-end scenarios and then use Task recorder to create a recording of the steps in each scenario. For more information about how to create a task recording, see Task recorder resources. The scenarios that you should test depend on your customer's business requirements. In this topic, you will use the "Create and confirm a sales order" sample scenario.

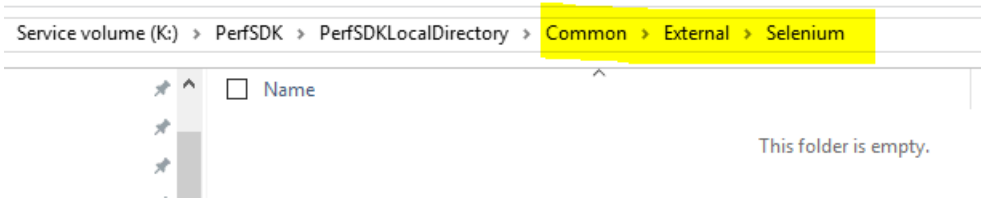
1. Sign in as a Sales persona.
2. Turn on Task recorder, and create and confirm a sales order that includes the following information:
   - Customer account
   - Item number
   - Sales quantity
   - Site
   - Warehouse
   - Sales price
3. When you've finished, select **Save as developer recording** to download the XML file.

Configure a development environment

1. Download the selenium-dotnet-strongnamed-3.13.1.zip and IEDriverServer_Win32_3.13.0.zip files.
2. Unblock and unzip the files.
3. In the **dist** folder, rename the .nupkg files as .zip files, and then unzip them.

ORIGINAL FILE NAME	NEW FILE NAME
Selenium.Support.StrongNamed.3.13.1.nupkg	Selenium.Support.StrongNamed.3.13.1.zip
Selenium.WebDriver.StrongNamed.3.13.1.nupkg	Selenium.WebDriver.StrongNamed.3.13.1.zip

4. Under your **PerfSDK** folder, create a folder that is named **Common\External\Selenium**.

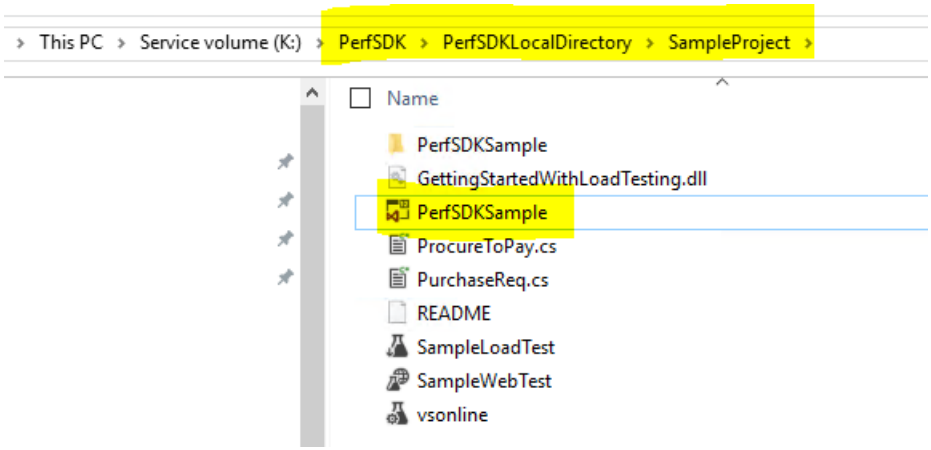


5. Copy the following files, and save them to the folder **Common\External\Selenium** that you created in the previous step:
   - IEDriverServer.exe from the unzipped IEDriverServer_Win32_3.13.0.zip file
   - WebDriver.dll and WebDriver.xml from the lib\net45 folder in the unzipped Selenium.WebDriver.StrongNamed.3.13.1.zip file
   - WebDriver.Support.dll and WebDriver.Support.xml from the lib\net45 folder in the unzipped Selenium.Support.StrongNamed.3.13.1.zip file

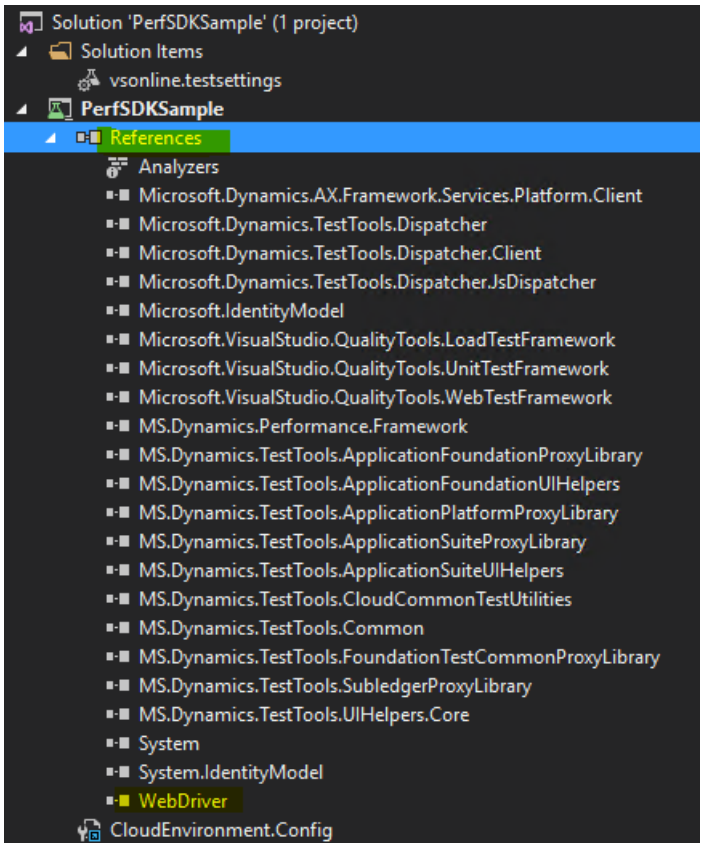
Generate a C# performance test from Task recorder

When you've finished recording the end-to-end scenario, you must generate a C# performance test script that is based the task recording.

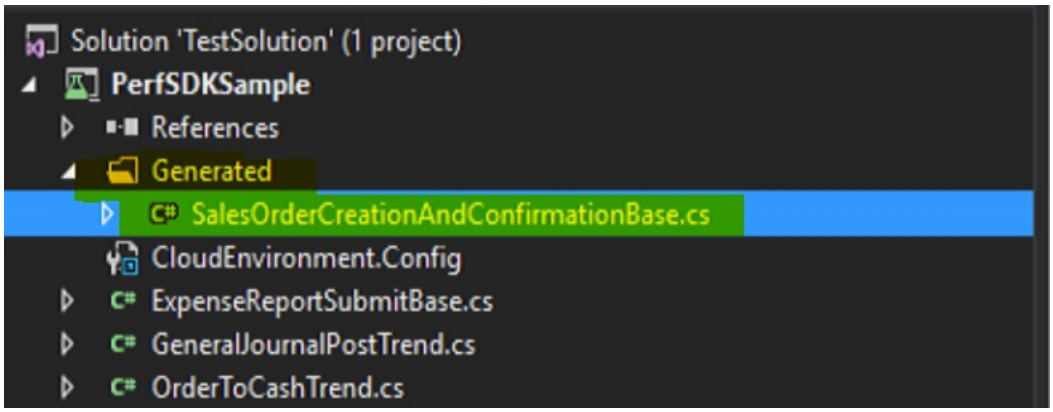
1. In a development environment, open Microsoft Visual Studio as an admin.
2. From your **PerfSDK** folder, open the **PerfSDKSample** solution. In a tier-1 sandbox or a cloud-hosted-environment, the PerfSDK folder is typically in <Service volumne>:\PerfSDK\PerfSDKLocalDirectory.



3. Add a reference to the WebDriver.dll file in the Common\External\Selenium folder.



4. On the **Dynamics 365** menu, point to **Addins**, and then select **Create C# perf test from recording**.
5. In the **Import Task Recording** dialog box, enter the following required details:
   - **Recording path** – The file location of the developer recording of your end-to-end scenario.
   - **Project path** – The location of the PerfSDKSample project. Typically, the path is <Your_PerfSDK_Folder>\SampleProject\PerfSDKSample\PerfSDKSample.csproj.
   - **PerfSDK path** – The location of PerfSDK. Typically, the path is <ServiceVolumeDrive>\PerfSDK\PerfSDKLocalDirectory.
6. When you've finished, select **Import**. A new C# class is created under the **Generated** folder of your PerfSDKSample project.



7. Build the solution.

Run single-user testing by using Test Explorer in Visual Studio

1. Update the **CloudEnvironment.config** file of the PerfSDKSample project in the following ways, so that it reflects the configuration of your environment:
   - Verify that the **HostName** and **SOAPHostName** match your development environment.
   - Verify that the **UserName** for **SelfMintingAdminUser** matches the admin account of your

development environment.

- In each **AuthenticatorConfiguration** element under the **AuthenticatorConfigurationCollection** element, replace **AadAuthenticator** with **SelfMintedTokenAuthenticator**.
- Comment out the **AzureActiveDirectoryConfiguration** and **KeyVaultConfigurations** elements.

```
<?xml version="1.0" encoding="utf-8"?>
<EnvironmentalConfigSettings xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <EnvironmentalConfigSettingsCollection>
    <EnvironmentalConfigSetting ConfigName="DEVFABRIC">
      <!-- NOTE: The HostName and SoapHostName value needs to be specified -->
      <ExecutionConfigurations Key="HostName" Value="[redacted]sandbox.ax.dynamics.com" />
      <ExecutionConfigurations Key="SoapHostName" Value="[redacted]ossoap.sandbox.ax.dynamics.com" />
      <ExecutionConfigurations Key="AdminAuthenticatorConfigurationId" Value="SelfMintingAdminUser" />
      <ExecutionConfigurations Key="DefaultBrowser" Value="InternetExplorer" />
      <ExecutionConfigurations Key="FederationRealm" Value="spn:00000015-0000-0000-c000-000000000000" />
      <ExecutionConfigurations Key="DefaultDispatcher" Value="Microsoft.Dynamics.TestTools.Dispatcher.JsDispatcher, Microsoft.Dynamics.TestTool
      <ExecutionConfigurationsNodes ConfigurationName="SVC">
        <ConfigurationSpecificDetails Key="AppConfig" Value="DEVFABRIC.Config" />
      </ExecutionConfigurationsNodes>
      <ExecutionConfigurationsNodes ConfigurationName="PRF">
        <!-- NOTE: Leave IsPerfSdkTest set to true -->
        <ConfigurationSpecificDetails Key="IsPerfSdkTest" Value="True" />
        <ConfigurationSpecificDetails Key="UserCount" Value="10" />
        <!-- NOTE: The UserFormat tenant should match the SelfMintingAdminUser tenant -->
        <ConfigurationSpecificDetails Key="UserFormat" Value="TST_{0}@{Tenant}" />
        <ConfigurationSpecificDetails Key="UserRole" Value="-SYSADMIN-" />
        <ConfigurationSpecificDetails Key="ThinkTime" Value="0" />
        <ConfigurationSpecificDetails Key="Company" Value="USMF" />
      </ExecutionConfigurationsNodes>
    </EnvironmentalConfigSetting>
  </EnvironmentalConfigSettingsCollection>
  <AuthenticatorConfigurationCollection>
    <AuthenticatorConfiguration Id="SelfMintingRunnerUser" Class="MS.Dynamics.TestTools.CloudCommonTestUtilities.Authentication.SelfMintedToken
      <Credentials IsFromKeyVault="false" Username="daxrunneruser@daxmdsrunner.com" NetworkDomain="urn:Microsoft:Dynamics:Cloud:DaxRunner" />
    </AuthenticatorConfiguration>
    <AuthenticatorConfiguration Id="SelfMintingSysUser" Class="MS.Dynamics.TestTools.CloudCommonTestUtilities.Authentication.SelfMintedTokenAut
      <Credentials IsFromKeyVault="false" Username="testuser@microsoft.com" />
    </AuthenticatorConfiguration>
    <!-- NOTE: Specify the PerfSDK user with the System Administrator role -->
    <AuthenticatorConfiguration Id="SelfMintingAdminUser" Class="MS.Dynamics.TestTools.CloudCommonTestUtilities.Authentication.SelfMintedTokenA
      <Credentials IsFromKeyVault="false" Username="[redacted]com" />
    </AuthenticatorConfiguration>
  </AuthenticatorConfigurationCollection>
  <!-- NOTE: The AAD identity provider and App ID must be specified  -->
  <!--<AzureActiveDirectoryConfiguration IdentityProvider="https://login.windows.net/{Tenant}" ClientAppId="" />-->
  <!-- NOTE: Key vault configuration is optional for tests run locally. If this is not specified, a popup will prompt for the App Authenticatio
  <!--<KeyVaultConfigurations>
    <KeyVaultConfiguration Key="KeyVaultUrl" Value="" />
    <KeyVaultConfiguration Key="KeyVaultCertificateThumbprint" Value="" />
    <KeyVaultConfiguration Key="KeyVaultClientId" Value="" />
    <KeyVaultConfiguration Key="KeyVaultSecretName" Value="" />
  </KeyVaultConfigurations>-->
</EnvironmentalConfigSettings>
```

2. In Visual Studio, on the **Test** menu, point to **Windows**, and then select **Test Explorer**.
3. Right-click your test case, and then select **Run selected tests**.

Tips and tricks

Use the following tips and tricks for single-user testing that uses Task recorder and the Performance SDK:

- Run your business end-to-end scenario first, before you capture it by using Task recorder.
- When you record your scenario by using Task recorder, enter values manually instead of selecting them in drop-down lists.
- Replay your task recording to make sure that everything works as you expect.
- Restart Visual Studio if you don't see your test case after the solution is built.

Troubleshooting

For information about single-user or multi-user testing that uses the Performance SDK, see Troubleshooting guide for single-user or multi-user testing with the Performance SDK.

Multi-user Performance SDK testing with a local test controller

2/18/2021 • 8 minutes to read • Edit Online

IMPORTANT

Visual Studio 2019 will be the last version of Visual Studio with web performance and load test features. In the future, we will publish some recommendations for alternative solutions.

- If you are using the Visual Studio and Test Controller/Test Agent for on-premises load testing, Visual Studio 2019 will be the last version. You can continue using it until the end of the support cycle.
- If you are using the cloud-based load testing service, it will continue to run through March 31, 2020. Until then, you can continue to use all of the experiences powered by this service without interruption. Alternatively, you can switch to on-premises load testing. For more information, see Cloud-based load testing service end of life.

Prerequisites

Before you complete the steps in this topic, verify that the following prerequisites are met:

- You have a development environment that has Platform update 21 or later in your Microsoft Azure subscription.

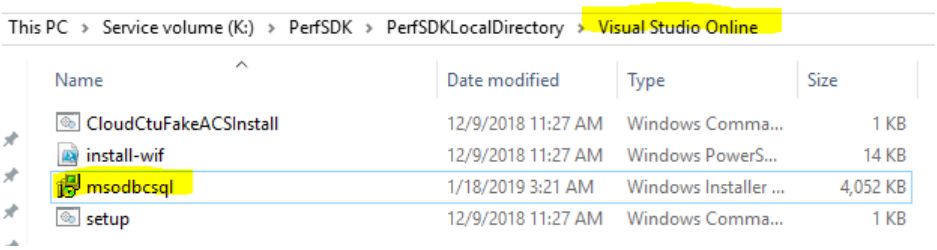
IMPORTANT

If your Finance and Operations apps were deployed in 21Vianet, the platform update of your environment must be Platform Update for 10.0.11 or above.

- You have Microsoft Visual Studio Enterprise edition in a development environment.
- You have a tier-2 or above sandbox environment that has the same release (application version and platform update) as your development environment.
- You've configured your development environment by following the steps in Single-user testing with Task recorder and the Performance SDK.
- C# performance testing classes have been generated for your end-to-end (E2E) scenarios, and you can run a single-user test by following the steps in Single-user testing with Task recorder and the Performance SDK.

Configure a development environment for multi-user testing

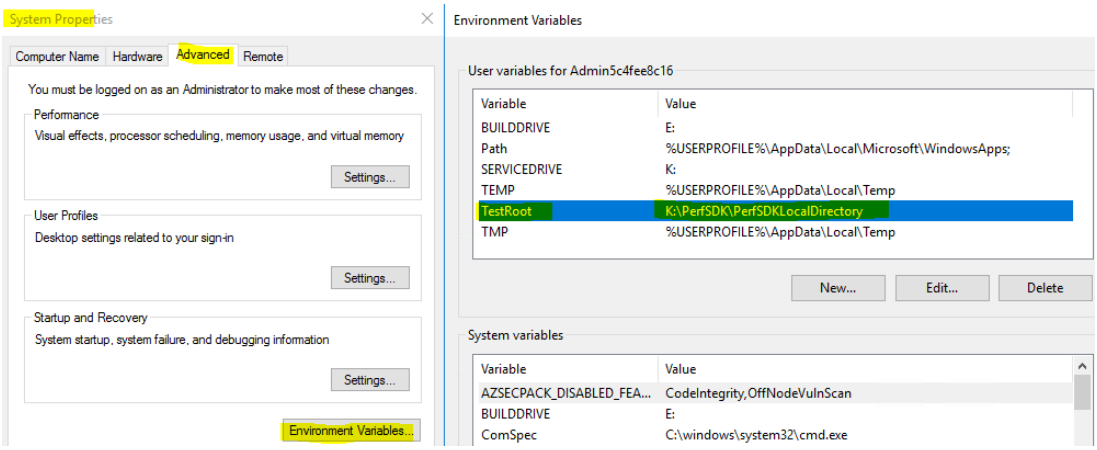
1. Download ODBC Driver 17 for SQL Server, rename the download file **msodbcsql**, and copy it to the **Visual Studio Online** folder under the **PerfSDK** folder.



2. Create an environment variable that is named **TestRoot**, and point it to the **PerfSDK** folder by running the following cmdlet in Microsoft Windows PowerShell.

```
[ENVIRONMENT]::SETENVIRONMENTVARIABLE("TESTROOT", "K:\PERFSDK\PERFSDKLOCALDIRECTORY", "USER")
```

To view the new environment variable, in the **System Properties** dialog box, on the **Advanced** tab, select **Environment Variables**.



3. Open a **Command Prompt** window as an admin, and enter the following commands to generate and install the required certificate. When you're prompted for a private key password, select **None**.

```
"C:\Program Files (x86)\Windows Kits\8.1\bin\x64\makecert" -n "CN=127.0.0.1" -ss Root -sr
LocalMachine -a sha256 -len 2048 -cy end -r -eku 1.3.6.1.5.5.7.3.1 -sv c:\temp\authcert.pvk
c:\temp\authcert.cer

"c:\Program Files (x86)\Windows Kits\8.1\bin\x64\pvk2pfx" -pvk c:\temp\authCert.pvk -spc
c:\temp\authcert.cer -pfx c:\temp\authcert.pfx
```

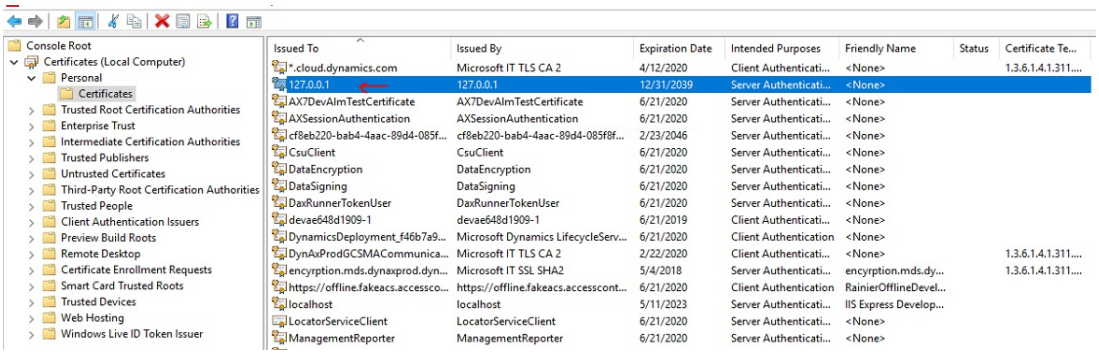
Here is an explanation of the elements in preceding commands:

- **-n "CN=127.0.0.1"** gives a human-readable name to the certificate. It's very important that the name of this certificate be **127.0.0.1**. Otherwise, the single-user tests won't be able to run.
- **-eku 1.3.6.1.5.5.7.3.1** gives the purpose of the certificate. It indicates that the certificate can be used as a Secure Sockets Layer (SSL) server certificate.

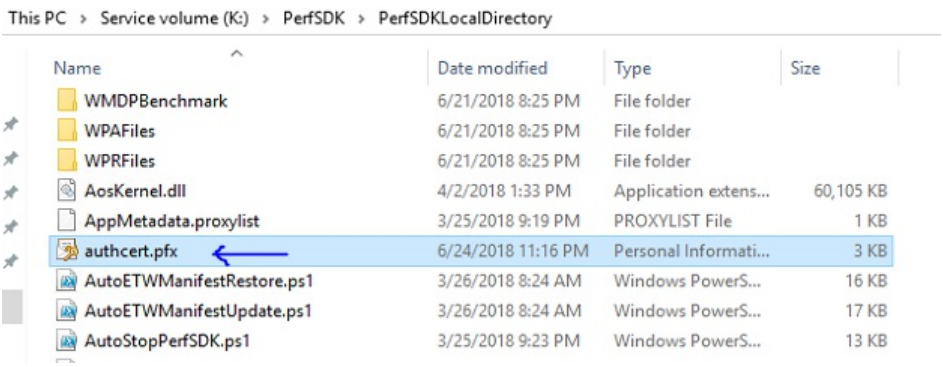
The commands generate the following certificates and save them in C:\Temp:

- Authcert.pvk
- Authcert.cer
- Authcert.pfx

4. Install the **authcert.pfx** and **authcert.cer** certificates in the **Local Machine\Personal** certificate store.



5. Copy the **authcert.pfx** certificate to the **PerfSDK\PerfSDKLocalDirectory** folder.



6. Replace the **setup.md** file in the **Visual Studio Online** folder with the following.

```
setx testroot "%DeploymentDirectory%"
ECHO Installing D365 prerequisites
ECHO MSIEXEC /a %DeploymentDirectory%\msodbcsql /passive /norestart IACCEPTMSODBCSQLLICENSETERMS=YES
MSIEXEC /a %DeploymentDirectory%\msodbcsql /passive /norestart IACCEPTMSODBCSQLLICENSETERMS=YES
%windir%\sysnative\windowspowershell\v1.0\powershell.exe -File %DeploymentDirectory%\install-wif.ps1
Md %DeploymentDirectory%\Common\Team\Foundation\Performance\Framework
%DeploymentDirectory%\CloudCtuFakeACSInstall.cmd %DeploymentDirectory%\authcert.pfx
```

7. In the **Visual Studio Online** folder, in the **CloudCtuFakeACSInstall.cmd** file, remove **%TestCertPassword%** from the **Import** command.

```
set TestCertPath=%1
set TestCertPassword=%2
ECHO Install the CloudCtu test certificate.
set MyStoreInstallCmd="&{$pfxcert = new-object
system.security.cryptography.x509certificates.x509certificate2;$pfxcert.Import('%TestCertPath%', '',
'Exportable,PersistKeySet');$store = new-object
System.Security.Cryptography.X509Certificates.X509Store('My','LocalMachine');$store.open('MaxAllowed');$store.Add(
$pfxcert)}"
ECHO CALL powershell -command %MyStoreInstallCmd%
CALL powershell -command %MyStoreInstallCmd%
```

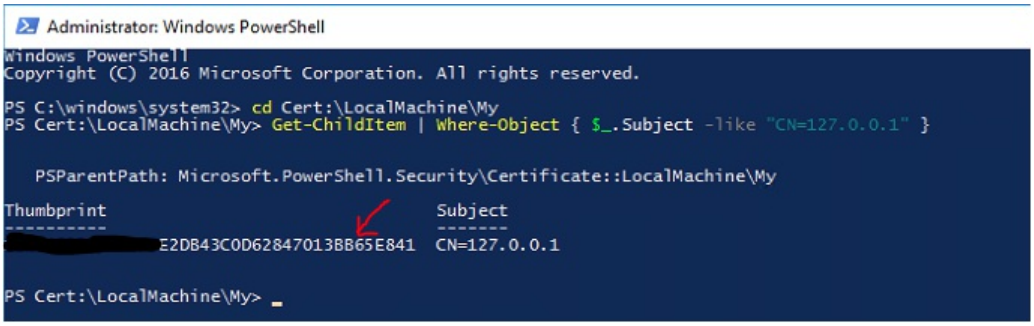
Prepare the PerfSDKSample solution for multi-user testing

1. In Microsoft Visual Studio, on the **Test** menu, point to **Test settings**, point to **Default processor architecture**, and then select **x64**.

2. Retrieve the thumbprint of the **authcert.pfx** certificate in your development environment by running the following cmdlets as an admin. Save the thumbprint somewhere, because you will need it when you configure the tier-2 or above sandbox environment.

```
cd Cert:\LocalMachine\My
Get-ChildItem | Where-Object { $_.Subject -like "CN=127.0.0.1" }
```

The following illustration shows an example of a thumbprint that these cmdlets return.



> **NOTE**
>
> In an environment that has Platform update 21 or later, a certificate that has **127.0.0.1** as the issuer is automatically installed. Verify that you retrieve the thumbprint of the certificate that you generated.

3. Update the **CloudEnvironment.config** file to reflect your configurations. As part of this update, follow these steps:

   - Verify that **HostName** and **SOAPHostName** match your tier-2 or above sandbox environment.

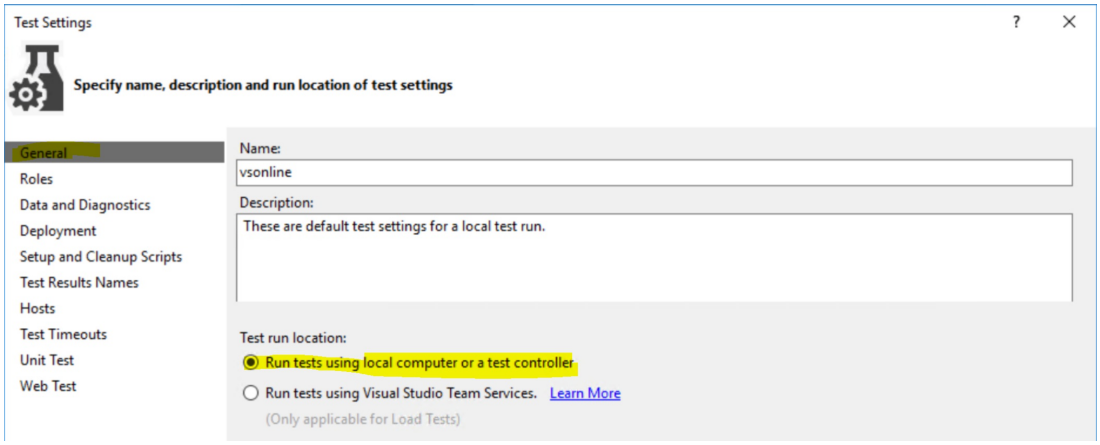
> **NOTE**
>
> If you don't know the **HostName** or **SOAPHostName** of your test environment, you can find it in the web.config file for the AOS server in **Infrastructure.HostUrl** or **Infrastructure.SoapServicesUrl**.

   - Add the thumbprint of the **authcert.pfx** certificate as a value of **SelfSigningCertificateThumbprint**.
   - Update **UserCount** to reflect the number of test users in your case.
   - Update **UserFormat** to reflect your naming convention for test users.

```
<?xml version="1.0" encoding="utf-8"?>
<EnvironmentalConfigSettings xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <EnvironmentalConfigSettingsCollection>
    <EnvironmentalConfigSetting ConfigName="DEVFABRIC">
      <!-- NOTE: The HostName and SoapHostName value needs to be specified -->
      <ExecutionConfigurations Key="HostName" Value="d365perftechtalkdev2abc5b5f03e5be852devaos.cloudax.dynamics.com" />
      <ExecutionConfigurations Key="SoapHostName" Value="d365perftechtalkdev2abc5b5f03e5be852devaossoap.cloudax.dynamics.com" />
       <ExecutionConfigurations Key="SelfSigningCertificateThumbprint" Value="01B47080E3C4171E5B6BC667E52870E8A2091759" />
      <ExecutionConfigurations Key="AdminAuthenticatorConfigurationId" Value="SelfMintingAdminUser" />
      <ExecutionConfigurations Key="DefaultBrowser" Value="InternetExplorer" />
      <ExecutionConfigurations Key="FederationRealm" Value="spn:00000015-0000-0000-c000-000000000000" />
      <ExecutionConfigurations Key="DefaultDispatcher" Value="Microsoft.Dynamics.TestTools.Dispatcher.JsDispatcher, Microsoft.Dynamics.TestTools.Dispatcher.JsDispatcher" />
      <ExecutionConfigurationsNodes ConfigurationName="SVC">
        <ConfigurationSpecificDetails Key="AppConfig" Value="DEVFABRIC.Config" />
      </ExecutionConfigurationsNodes>
      <ExecutionConfigurationsNodes ConfigurationName="PRF">
        <!-- NOTE: Leave IsPerfSdkTest set to true -->
        <ConfigurationSpecificDetails Key="IsPerfSdkTest" Value="True" />
        <ConfigurationSpecificDetails Key="UserCount" Value="5" />
        <!-- NOTE: The UserFormat tenant should match the SelfMintingAdminUser tenant -->
        <ConfigurationSpecificDetails Key="UserFormat" Value="TEST00{0}@axsolutionsarchitecture.com" />
        <ConfigurationSpecificDetails Key="UserRole" Value="-SYSADMIN-" />
        <ConfigurationSpecificDetails Key="ThinkTime" Value="0" />
        <ConfigurationSpecificDetails Key="Company" Value="USMF" />
      </ExecutionConfigurationsNodes>
    </EnvironmentalConfigSetting>
  </EnvironmentalConfigSettingsCollection>
  <AuthenticatorConfigurationCollection>
    <AuthenticatorConfiguration Id="SelfMintingRunnerUser" Class="MS.Dynamics.TestTools.CloudCommonTestUtilities.Authentication.SelfMintedTokenAuthenticator">
      <Credentials IsFromKeyVault="false" Username="daxrunneruser@daxmdsrunner.com" NetworkDomain="urn:Microsoft:Dynamics:Cloud:DaxRunner" />
    </AuthenticatorConfiguration>
    <AuthenticatorConfiguration Id="SelfMintingSysUser" Class="MS.Dynamics.TestTools.CloudCommonTestUtilities.Authentication.SelfMintedTokenAuthenticator">
      <Credentials IsFromKeyVault="false" Username="testuser@microsoft.com" />
    </AuthenticatorConfiguration>
    <!-- NOTE: Specify the PerfSDK user with the System Administrator role -->
    <AuthenticatorConfiguration Id="SelfMintingAdminUser" Class="MS.Dynamics.TestTools.CloudCommonTestUtilities.Authentication.SelfMintedTokenAuthenticator">
      <Credentials IsFromKeyVault="false" Username="sa@axsolutionsarchitecture.com" />
    </AuthenticatorConfiguration>
  </AuthenticatorConfigurationCollection>
  <!-- NOTE: The AAD identity provider and App ID must be specified  -->
  <!--<AzureActiveDirectoryConfiguration IdentityProvider="https://login.windows.net/{Tenant}" ClientAppId="" />-->
  <!-- NOTE: Key vault configuration is optional for tests run locally. If this is not specified, a popup will prompt for the App Authentication key. -->
  <!--<KeyVaultConfigurations>
    <KeyVaultConfiguration Key="KeyVaultUrl" Value="" />
    <KeyVaultConfiguration Key="KeyVaultCertificateThumbprint" Value="" />
    <KeyVaultConfiguration Key="KeyVaultClientId" Value="" />
    <KeyVaultConfiguration Key="KeyVaultSecretName" Value="" />
  </KeyVaultConfigurations>-->
</EnvironmentalConfigSettings>
```

   - If your Finance and Operations apps were deployed in 21Vianet, make sure to specify **NetworkDomain="https://sts.chinacloudapi.cn/"** in **SelfMintingSysUser** and **SelfMintingAdminUser**.

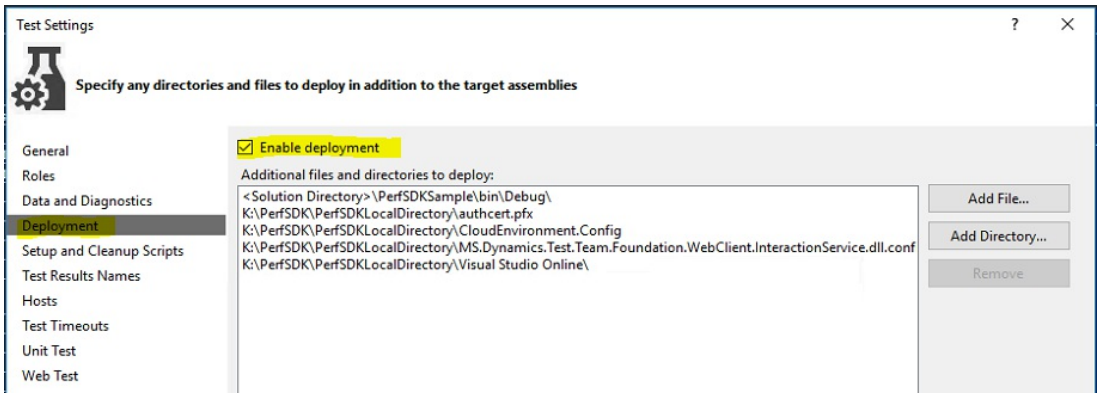
4. Configure **vsonline.testsettings**. In the **Test Settings** dialog box, on the **General** tab, in the **Test run Location** field group, select the **Run tests using local computer or a test controller** option.



> **NOTE**
>
> If you don't see **vsonline.testsettings**, reopen the solution.

5. On the **Deployment** tab, select the **Enable deployment** check box, and then use the **Add Directory** and **Add File** buttons to add the following folders and files to the **Additional files and directories to deploy** field.

   - **K:\PerfSDK\PerfSDKLocalDirectory\SampleProject\PerfSDKSample\bin\Debug** folder
   - **K:\PerfSDK\PerfSDKLocalDirectory\Visual Studio Online** folder
   - **K:\PerfSDK\PerfSDKLocalDirectory\CloudEnvironment.Config** file
   - **K:\PerfSDK\PerfSDKLocalDirectory\authcert.pfx** file
   - **K:\PerfSDK\PerfSDKLocalDirectory\MS.Dynamics.Test.Team.Foundation.WebClient.InteractionService.dll.co** file



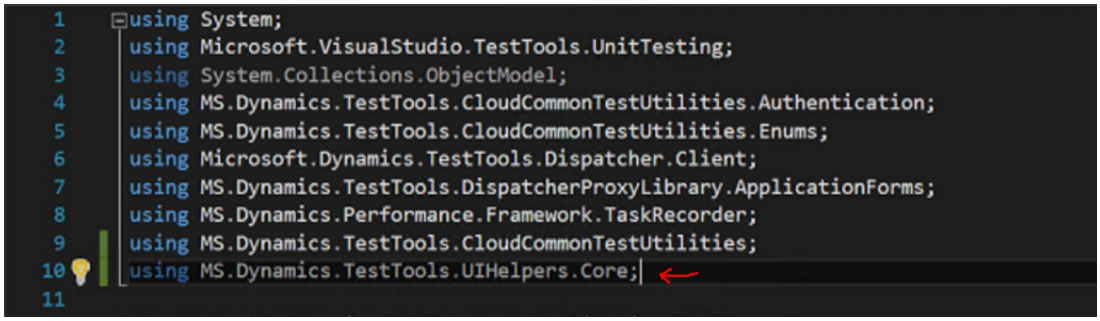
> **NOTE**
>
> When you receive a warning that states, "Deployment item not in solution folder," select **OK** to continue.

6. On the **Hosts** tab, in the **Run tests in 32 bits or 64 bits process** field, select **Run test in 64 bits process on 64 bits machine**.

7. Select **Apply**, and then close the **Test Settings** dialog box.

8. Modify your C# performance class by adding the following statement to your C# performance tests.

```
using MS.Dynamics.TestTools.UIHelpers.Core;
```



9. Modify the **TestSetup** method by adding the following lines to the beginning of the **TestSetup()** method.

```
var testroot = System.Environment.GetEnvironmentVariable("DeploymentDir");
if (string.IsNullOrEmpty(testroot))
{
    testroot = System.IO.Directory.GetCurrentDirectory();
}
Environment.SetEnvironmentVariable("testroot", testroot);
```

```
37          public void TestSetup()
38          {
39              var testroot = System.Environment.GetEnvironmentVariable("DeploymentDir");
40              if (string.IsNullOrEmpty(testroot))
41              {
42                  testroot = System.IO.Directory.GetCurrentDirectory();
43              }
44              Environment.SetEnvironmentVariable("testroot", testroot);
45
46              // For multi-user uncomment following lines
47              //if (this.TestContext != null)
48              //{
49              //    timerProvider = new TimerProvider(this.TestContext);
50              //}
51              SetupData();
52              _userContext = new UserContext(UserManagement.AdminUser);
53              // For multi-user testing use this line
54              // Client = new DispatchedClientHelper().GetClient();
55              Client = DispatchedClient.DefaultInstance;
56              Client.ForceEditMode = false;
57              Client.Company = WellKnownCompanyID.USMF.ToString();
58              Client.Open();
59          }
```

10. In the **TestSetup** method, uncomment lines in the code titled "for multi-user uncomment following lines", and comment out the following line.

```
Client = DispatchedClient.DefaultInstance;
```

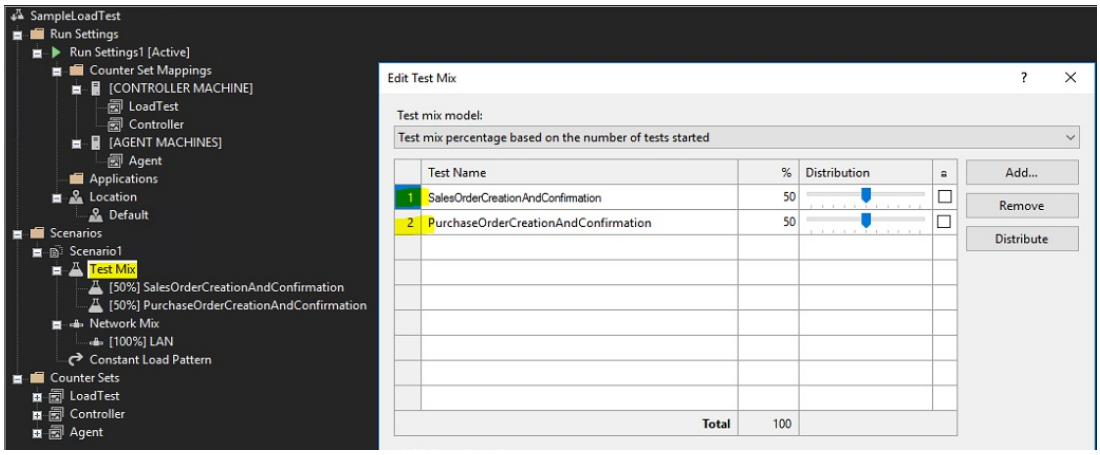
```
public void TestSetup()
{
    var testroot = System.Environment.GetEnvironmentVariable("DeploymentDir");
    if (string.IsNullOrEmpty(testroot))
    {
        testroot = System.IO.Directory.GetCurrentDirectory();
    }
    Environment.SetEnvironmentVariable("testroot", testroot);

    // For multi-user uncomment following lines
    if (this.TestContext != null)
    {
        timerProvider = new TimerProvider(this.TestContext);
    }
    SetupData();
    _userContext = new UserContext(UserManagement.AdminUser);
    // For multi-user testing use this line
    Client = new DispatchedClientHelper().GetClient();
    //Client = DispatchedClient.DefaultInstance;
    Client.ForceEditMode = false;
    Client.Company = WellKnownCompanyID.USMF.ToString();
    Client.Open();
}
```

11. Modify the **TestCleanup** method so that it resembles the following example.

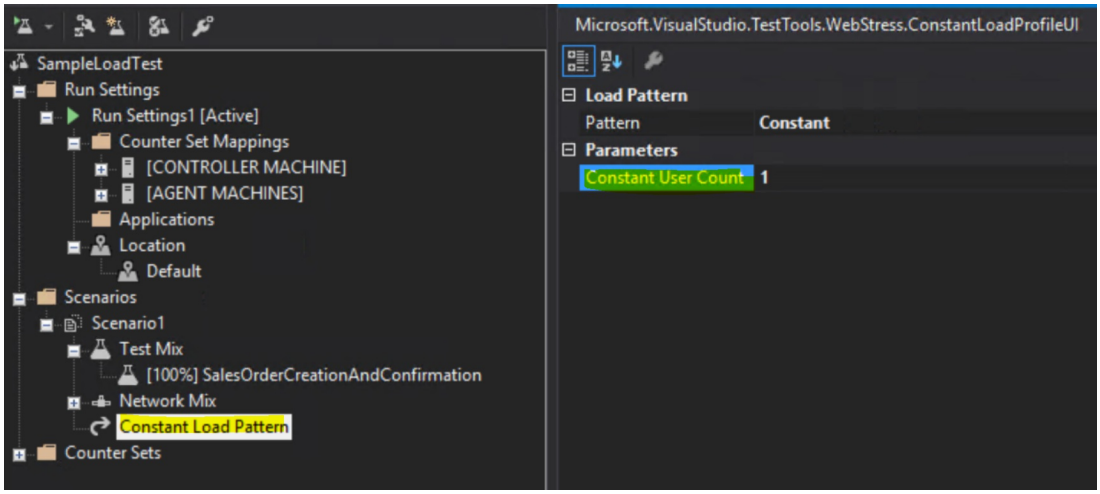
```
public void TestCleanup()
{
    Client.Close();
    Client.Dispose();
    Client = Null;
    //_userContext.Dispose();
}
```

12. Repeat steps 2 through 11 for all the C# performance test classes that you have.
13. When you've finished, build the solution.
14. In **SampleLoadTest.loadtest**, select your test class under **Test Mix**.



15. In **SampleLoadTest.loadtest**, update the **Timing** fields of **Run Settings1 [Active]**. These fields include **Warm-up Duration**, **Run Duration**, and **Cool-down Duration**.

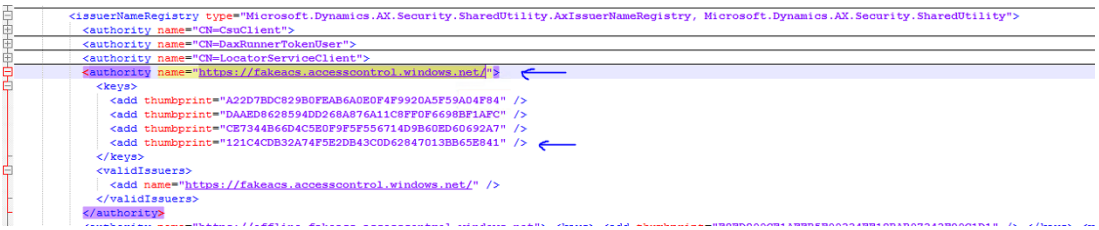
16. In **Constant Load Pattern**, set the **Constant User Count** parameter to the total number of users that you want to use to run the test.



Configure a tier-2 or above sandbox environment for multi-user testing

If your AOS allows Remote Desktop connections

1. Open Microsoft Internet Information Services (IIS) Manager from **Administrative Tools**, and then, under **Sites**, select **AOSService**.
2. On the right, select **Explore in Actions**, and then find the **wif.config** file at the bottom of the window.
3. Add the thumbprint of the **authcert.pfx** certificate to the bottom of the `https://fakeacs.accesscontrol.windows.net/` authority and save the change.



4. Repeat steps 1 through 3 on each AOS computer.

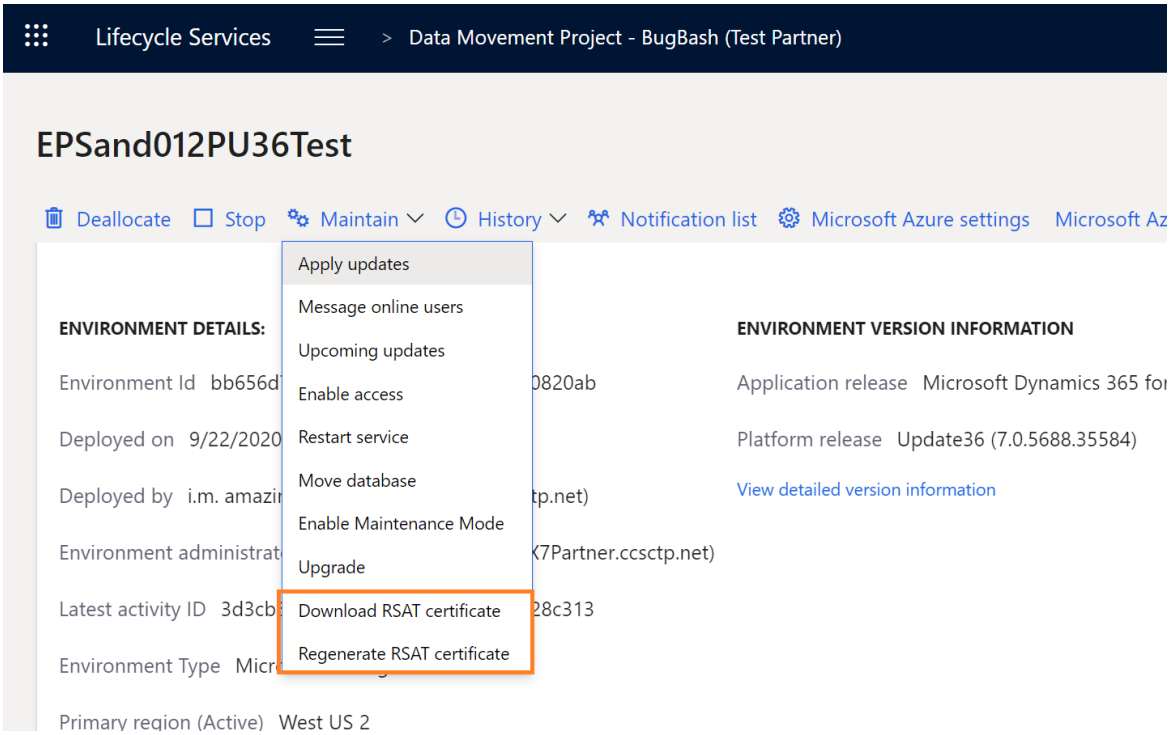
5. Restart all AOS instances.

If you do not have Remote Desktop access to the server

In cases where your Remote Desktop Protocol (RDP) access is removed, such as Microsoft-managed or self-service type sandboxes, Microsoft will generate the certificate for your environment and have it pre-configured. Follow these steps to retrieve the RSAT certificate, which is necessary to use with PerfSDK.

1. Under **Maintain** on your environment details page in Lifecycle Services you'll see two new options.

- Download RSAT certificate
- Regenerate RSAT certificate



Use the **Download** button to retrieve the certificate bundle as a .zip file.

2. You'll receive a warning that a clear-text password will be displayed on your screen. Select **Yes** to continue.

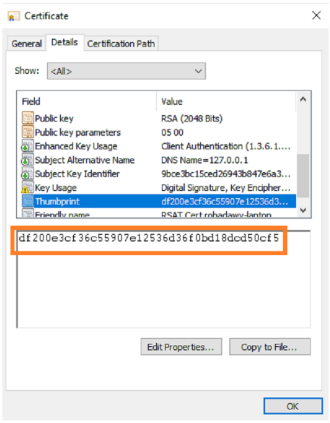
3. Copy the clear-text password for later use. You'll see the .zip file has been downloaded. Inside the .zip file is a certificate (.cer) and a personal information exchange (.pfx) file. Unzip the file.

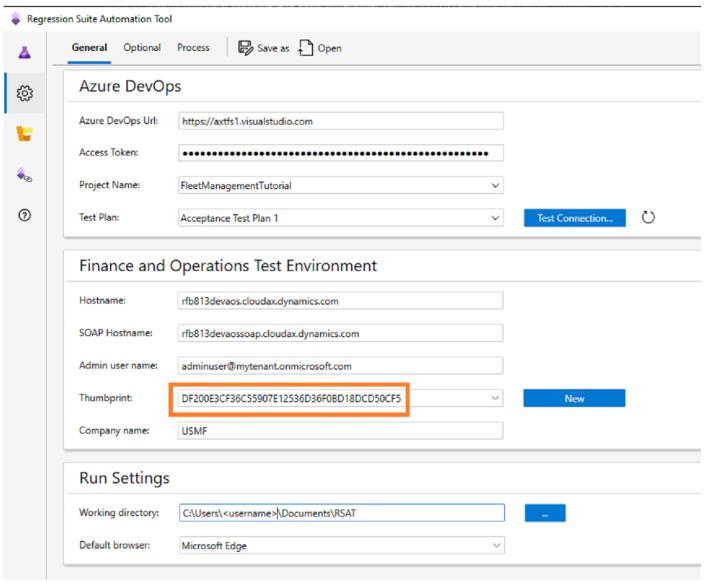
4. Double-click the certificate to open it, and then select **Install**. Install this certificate to your local machine, and then browse to the **Personal** store. Repeat this process for the local machine, and browse specifically to the **Trusted Root Certification Authorities** store.

5. Double-click the personal information exchange (.pfx) file to open it, and select **Install**. Install this certificate to your local machine, enter the password saved in step 2, and browse to the **Personal** store. Repeat this process for the local machine location, enter the password saved in step 2, and browse specifically to the **Trusted Root Certification Authorities** store.

6. Double-click the certificate file to open it. Browse to the **Details** tab, and scroll down until you see the **Thumbprint** section. Select **Thumbprint**, and copy the ID in the text box. Use this thumbprint for RSAT

and to update your PerfSDK **CloudEnvironment.config** thumbprint.

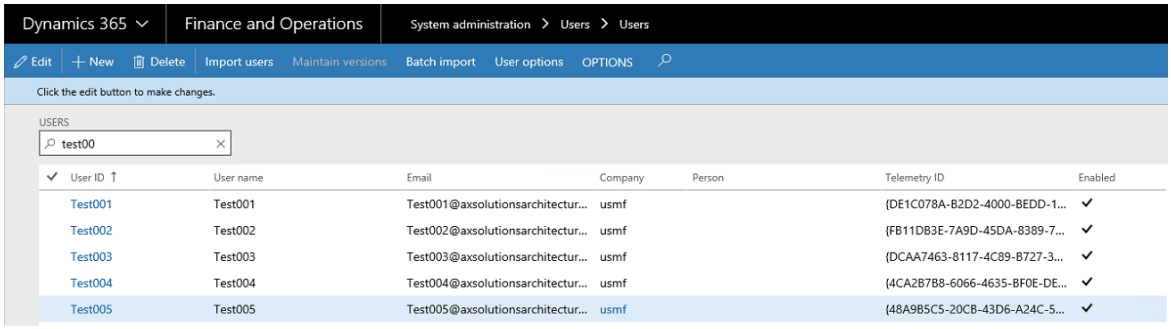




You can now run your tests against the environment using this certificate. The certificate will be auto-rotated by Microsoft before it expires, at which time you will need to download a new version of this certificate starting from step 1 above. For self-service environments this will be rotated every 90 days during a downtime window that is closest to the expiry. These downtime windows include customer initiated package deployment, and database movement operations that target the environment.

Create test users

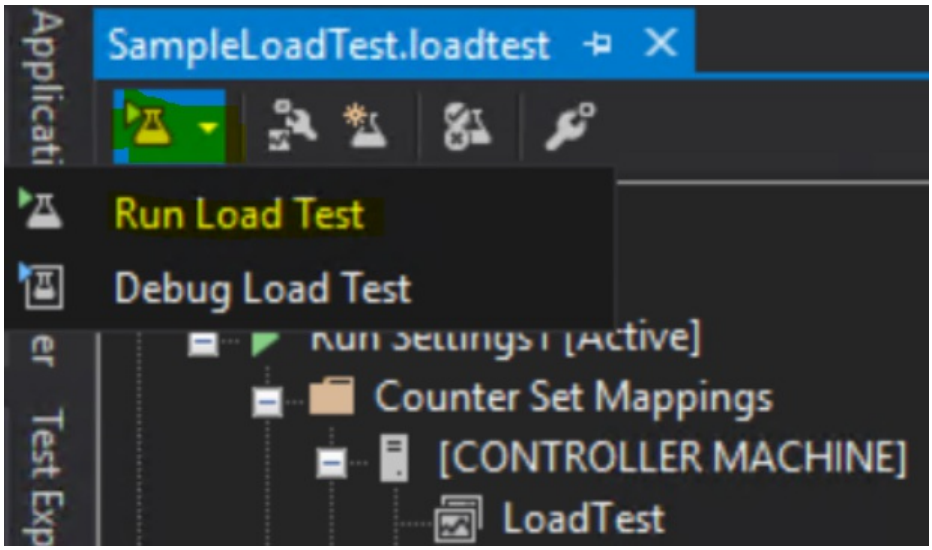
Open the **SampleLoadTest.load** test to create test users and import them into your target environment. Then assign the **System Administrator** security role to each user.



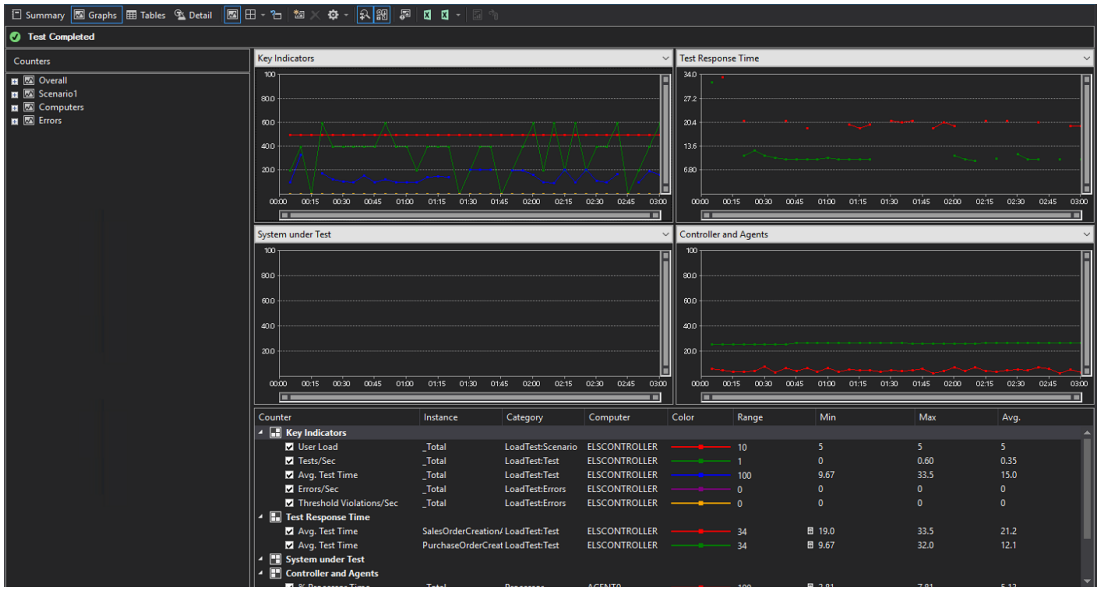
> **NOTE**
>
> You can also create test users by running **MS.Dynamics.Performance.CreateUsers.exe**. In this case, you don't have to use IISRESET.

Run multi-user testing by using a local test controller

1. On the devbox, in Visual Studio, select **Run Load Test**.



2. Review the test output.



Troubleshooting

For information about single-user or multi-user testing that uses the Performance SDK, see Troubleshooting guide for single-user or multi-user testing with the Performance SDK.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Multi-user testing with the Performance SDK and Azure DevOps

2/18/2021 • 2 minutes to read • Edit Online

IMPORTANT

Visual Studio 2019 will be the last version of Visual Studio with web performance and load test features. In the future, we will publish some recommendations for alternative solutions.

- If you are using the Visual Studio and Test Controller/Test Agent for on-premises load testing, Visual Studio 2019 will be the last version. You can continue using it until the end of the support cycle.
- If you are using the cloud-based load testing service, it will continue to run through March 31, 2020. Until then, you can continue to use all of the experiences powered by this service without interruption. Alternatively, you can switch to on-premises load testing. For more information, see Cloud-based load testing service end of life.

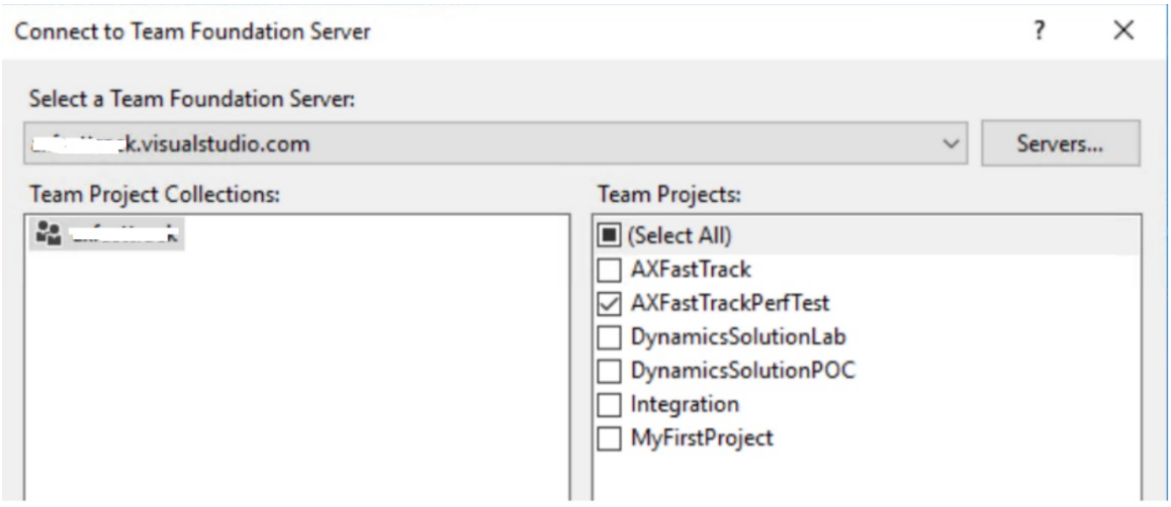
Prerequisites

Before you can complete the steps in this topic, verify that the following prerequisites are met:

- You have a development environment that has platform update 21 or later in own your Microsoft Azure subscription
- You have Microsoft Visual Studio Enterprise edition in a development environment.
- You have a tier-2 or above sandbox that has the same release (application version and platform update) as your development environment.
- You've completed all the steps in Single-user testing with Task recorder and the Performance SDK and all the steps except "Run multi-user testing with local test controller" in Multi-user testing with the Performance SDK and a local test controller.

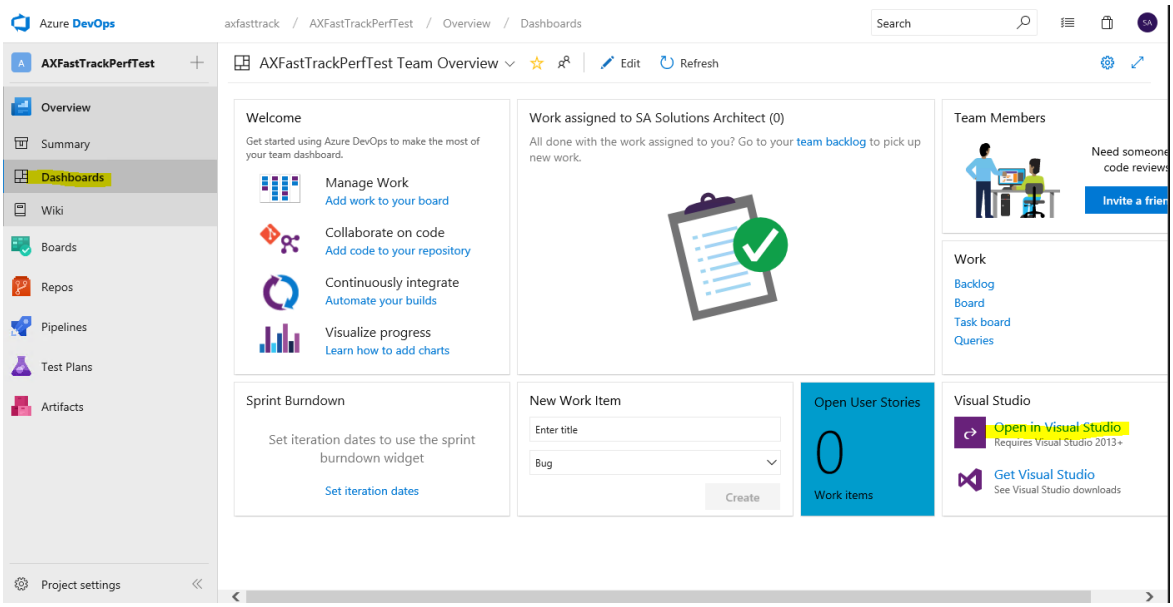
Prepare the PerfSDKSample solution for multi-user testing

1. Connect Microsoft Visual Studio in a development environment to an Azure DevOps project.



> **NOTE**
>
> When you add the Azure DevOps server, use **https://<YourAzureDevOpsAccount>.visualstudio.com**.

2. On your Azure DevOps project dashboard, select **Open in Visual Studio**, and then select **Allow in prompted window**.

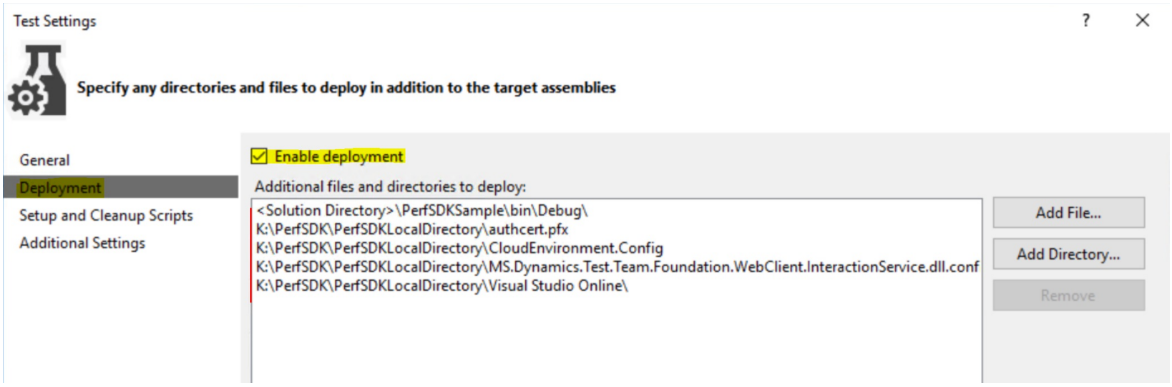


3. After Visual Studio is opened, verify that you're running it as an admin. If you aren't, close it, and then reopen it as an admin.

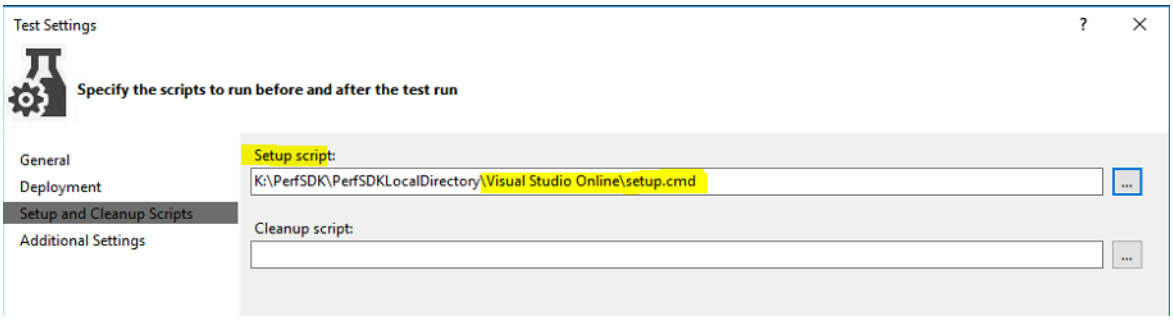
4. Go to K:\PerfSDK\PerfSDKLocalDirectory\SampleProject, and change the file, **vsonline.testsettings**.

5. In the **Test Settings** dialog box, on the **General** tab, select the **Run tests using Visual Studio Team Services** option.

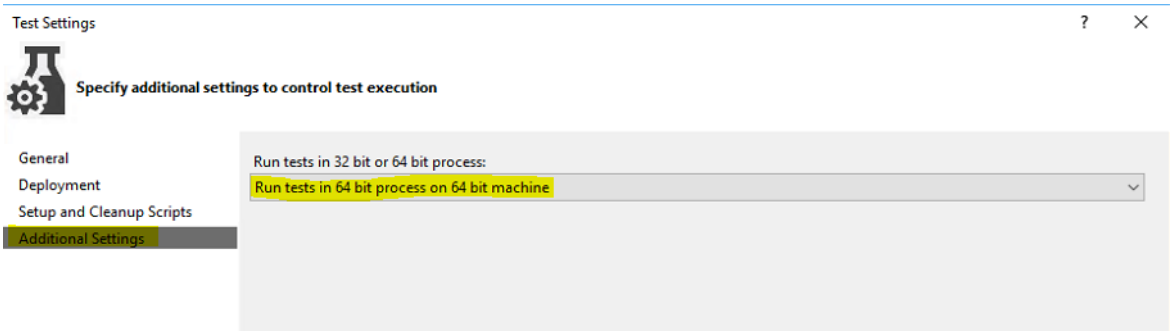
6. On the **Deployment** tab, leave all the fields set as they were set in Multi-user testing with the Performance SDK and a local test controller.



7. On the **Setup and Cleanup Scripts** tab, point the **Setup script** field to the **setup.cmd** file under the **Visual Studio Online** folder.



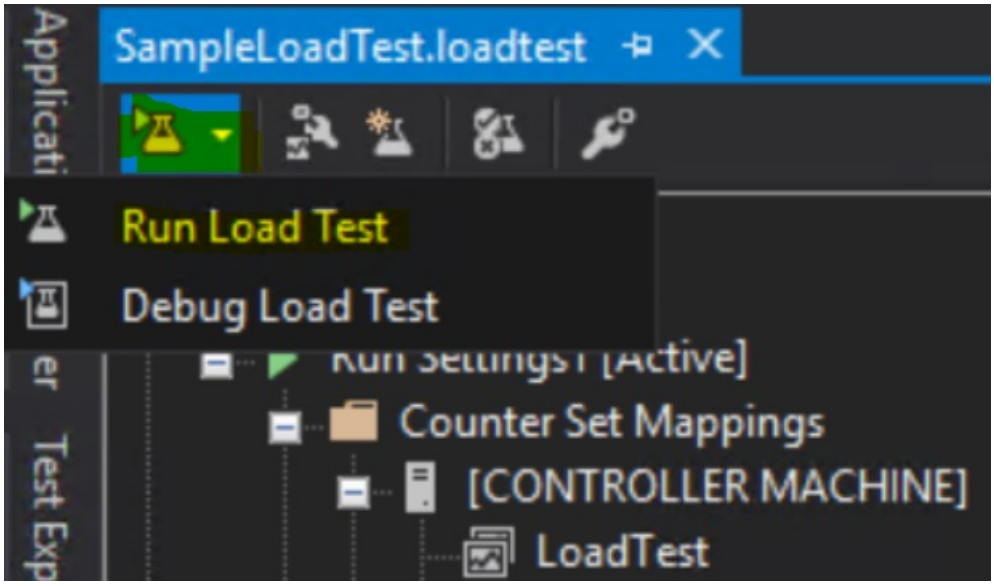
8. On the **Additional Settings** tab, leave all the fields set as they were set in Multi-user testing with the Performance SDK and a local test controller.



9. Select **Apply and Close**.

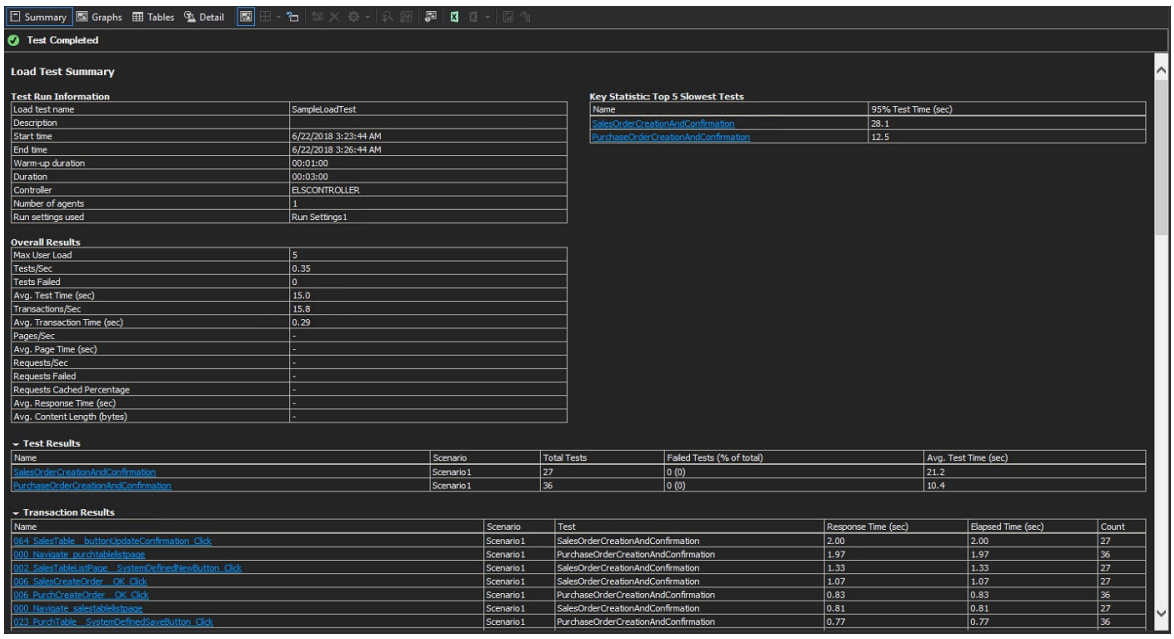
Run multi-user testing by using Azure DevOps

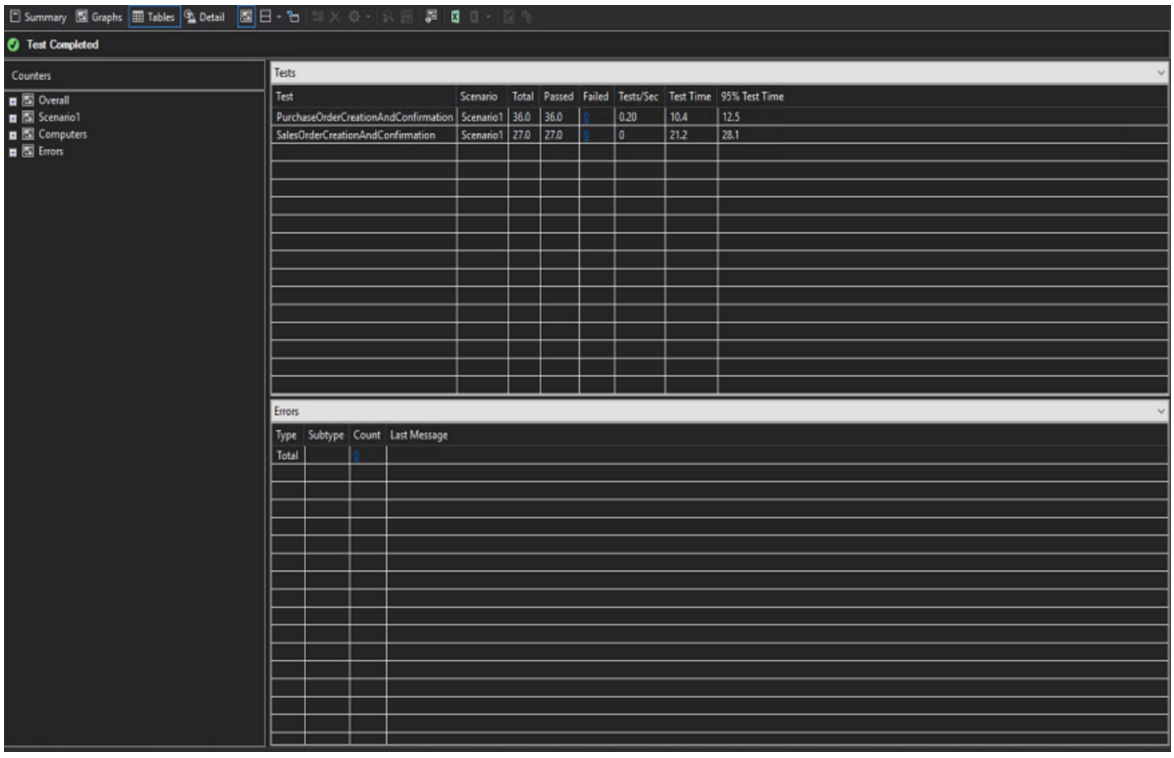
1. In Visual Studio, open **SampleLoadTest.loadtest**.
2. Select **Run Load Test**.

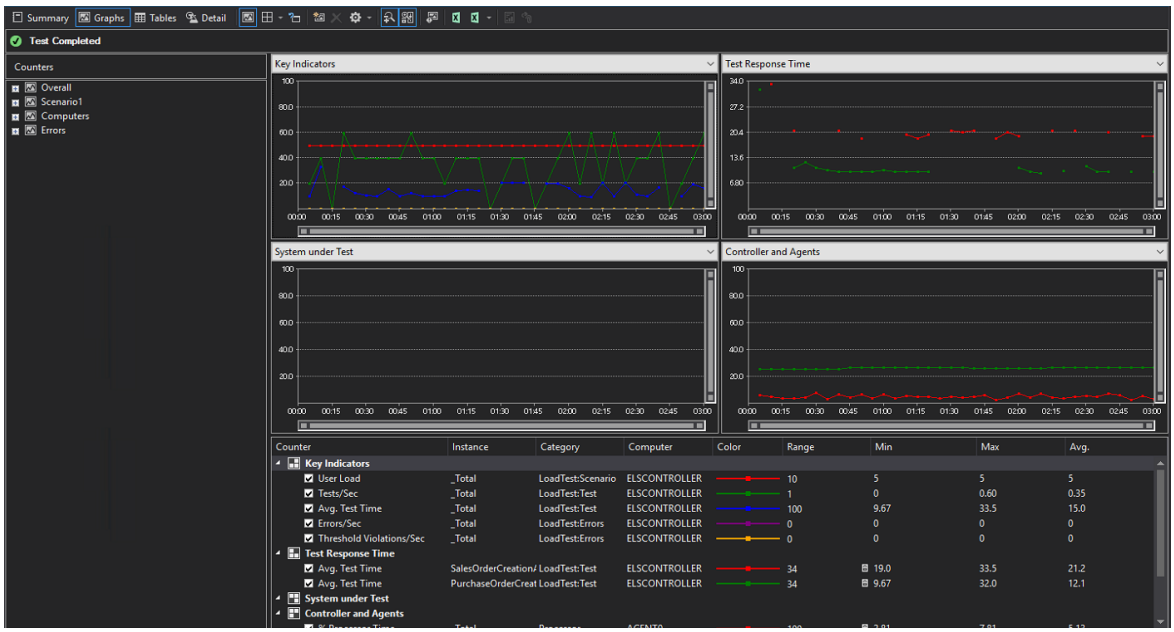


The load test is run, and a chart shows its progress.

3. Review the test output.







Tips

To switch multi-user testing between the local test controller and Azure DevOps, create a copy of **testsettings** for each, and then select the copy that you want as **Active Test Settings**.

Troubleshooting

For information about single-user or multi-user testing that uses the Performance SDK, see Troubleshooting guide for single-user or multi-user testing with the Performance SDK.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Troubleshooting guide for testing with the Performance SDK

2/18/2021 • 13 minutes to read • Edit Online

No client was opened in the time-out period

This issue affects only single-user tests. When the test is running, a web client is opened, but a website is never loaded. Instead, there is an empty web client that has a white background. The following message appears at the top of the page, "This is the initial start page for the WebDriver server." The test eventually times out and fails, and an error message is shown.

Error - No client was opened in the time-out period

> Initialization method <Test class name>.TestSetup threw an exception. System.TimeoutException: System.TimeoutException: No client was opened in the timeout period.

Solution

See Multi-user testing with the Performance SDK and a local test controller. That topic explains how to create a correct certificate for this type of test. It also explains how to add the thumbprint of the certificate to the wif.config file.

Zoom factor

This issue affects only single-user tests.

Error - Zoom factor

> Initialization method <Test class name>.TestSetup threw exception. System.InvalidOperationException: System.InvalidOperationException: Unexpected error launching Internet Explorer. Browser zoom level was set to 200%. It should be set to 100% (NoSuchDriver).

Solution - Zoom factor

In Internet Explorer, you can change the zoom factor to 100 percent by changing the following registry keys:

- Computer\HKEY_CURRENT_USER\SOFTWARE\Microsoft\Internet Explorer\Zoom\ResetZoomOnStartup = 0
- Computer\HKEY_CURRENT_USER\SOFTWARE\Microsoft\Internet Explorer\Zoom\ResetZoomOnStartup2 = 0
- Computer\HKEY_CURRENT_USER\SOFTWARE\Microsoft\Internet Explorer\Zoom\Zoomfactor = 80000

Depending on the version of the local machine that is used, before you start the Remote Desktop Protocol (RDP) session, you might have to select **Change the size of text, apps and other items**. This field is available in **Display settings** in Microsoft Windows.

If those steps don't work, change the size of your remote desktop before you start the RDP session, so that the default zoom level in Internet Explorer is 100 percent.

Certificate thumbprint errors

Error example - Certificate thumbprint errors

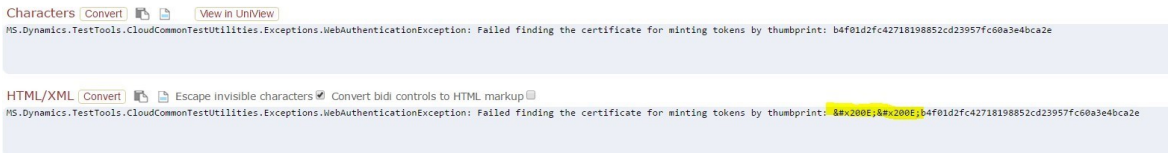
> Initialization method MS.Dynamics.Performance.Application.TaskRecorder.TestRecord1Base.TestSetup**

> threw an exception. System.TypeInitializationException: System.TypeInitializationException: The type initializer for 'MS.Dynamics.TestTools.CloudCommonTestUtilities.Authentication.UserManagement' threw an exception. --> MS.Dynamics.TestTools.CloudCommonTestUtilities.Exceptions.WebAuthenticationException: Failed finding the certificate for minting tokens by thumbprint: b4f01d2fc42718198852cd23957fc60a3e4bca2e.

Solution - Certificate thumbprint errors

You might receive the error message for one of the following reasons:

- The certificate thumbprint that you copied into the CloudEnvironment.Config and wif.config files includes invisible Unicode characters. To determine whether the thumbprint contains invisible Unicode characters, paste it into a Unicode code converter, and see whether extra characters appear in the **HTML/XML** field. For example, you can use the Unicode converter that is available at https://r12a.github.io/apps/conversion/.



- The certificate wasn't installed on the Application Object Server (AOS) machine. To verify that the certificate can be found on the AOS machine, run the following Microsoft Windows PowerShell script.

  ```
  cd Cert:\LocalMachine\My
  Get-ChildItem | Where-Object { $_.Subject -like "CN=<name of your certificate>" }
  ```

  If the thumbprint doesn't appear in the Windows PowerShell console after you run the script, the certificate isn't installed. To fix the issue, copy and install a .cer file on all AOS machines.

- If this issue occurs when you run load tests, the setup scripts might not have installed the corresponding .pfx file correctly. Verify that the password that is specified in the CloudCtuFakeACSInstall.cmd file matches the password that was set when the certificate was created.

No endpoint is listening

This issue can occur when you run single-user or multi-user tests, or when you create users by using MS.Dynamics.Performance.CreateUsers.exe.

Error example - No endpoint is listening

The tests fail, or the user creation process fails, and the following error message is shown:

> System.TypeInitializationException: The type initializer for 'MS.Dynamics.TestTools.CloudCommonTestUtilities.Authentication.UserManagement' threw an exception. ---> System.ServiceModel.EndpointNotFoundException: There was no endpoint listening at \<web address> that could accept the message. This is often caused by an incorrect address or SOAP action.

Solution - No endpoint is listening

This issue occurs when the host that is specified in the CloudEnvironment.Config file can't be accessed from the machine that is trying to run the tests or create users.

In the CloudEnvironment.Config file, review the values that are specified by the following keys:

- \<ExecutionConfigurations Key="HostName" Value="<web address of host>" />
- \<ExecutionConfigurations Key="SoapHostName" Value="<web address of SOAP>" />

The web addresses that are specified by these keys must be in the environment that you're testing. In a web browser on your developer machine, make sure that you can open the web address that is specified by the **HostName** key.

For online load tests, the environment that is specified by the **HostName** key in the CloudEnvironment.Config file must be publicly accessible from any machine. Therefore, if you must test a one-box environment, you won't be able to run the load test by using Microsoft Visual Studio Online, because the endpoint won't be accessible outside the one-box environment.

Users can't be enumerated

This issue can occur when you run multi-user tests, or when you create users by using MS.Dynamics.Performance.CreateUsers.exe.

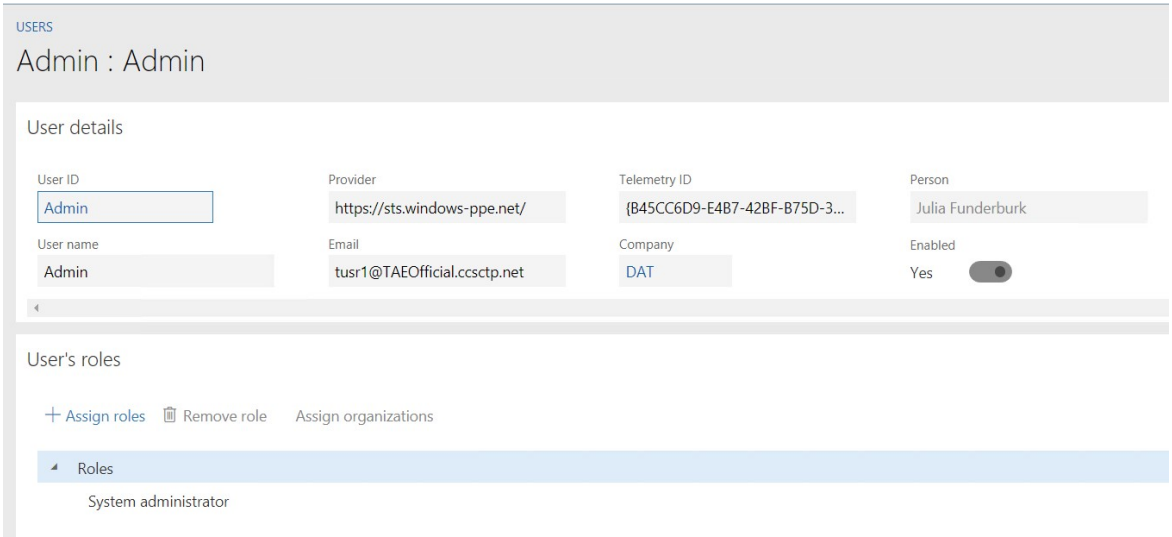
Error example - Users can't be enumerated

> System.TypeInitializationException: The type initializer for 'MS.Dynamics.TestTools.CloudCommonTestUtilities.Authentication.UserManagement' threw an exception. ---> System.InvalidOperationException: Could not enumerate AX users ---> System.ServiceModel.FaultException'1[System.ComponentModel.Win32Exception]: Forbidden

Solution - Users can't be enumerated

Three scenarios can cause this error:

- The System Administrator role isn't assigned to the user who is specified as **SelfMintingAdminUser** in the CloudEnvironment.config file. To verify that you've specified the correct user, sign in to the endpoint, and view the user's roles.



- The user who is specified as **SelfMintingAdminUser** in the CloudEnvironment.config file has a provider other than `https://sts.windows-ppe.net/` or `https://sts.windows.net/`. Sometimes, a company-specific domain is included in the **Provider** field for the admin user.
- If your Finance and Operations apps were deployed in 21Vianet, make sure that you have specified **NetworkDomain="https://sts.chinacloudapi.cn/"** in **SelfMintingSysUser** and **SelfMintingAdminUser**.

To work around this issue, create a user who has any name and email address. Assign the **System Administrator** role to the new user. You don't have to link the user to a real Microsoft Azure Active Directory (Azure AD) user. Specify this new admin user as **SelfMintingAdminUser** in the CloudEnvironment.config file.

The HTTP request was forbidden with client authentication scheme 'Anonymous'

Error example - The HTTP request was forbidden

> Initialization method <Test class name>.TestSetup threw exception. System.ServiceModel.Security.MessageSecurityException: System.ServiceModel.Security.MessageSecurityException: The HTTP request was forbidden with client authentication scheme 'Anonymous'. ---> System.Net.WebException: The remote server returned an error: (403) Forbidden.

Solution - The HTTP request was forbidden

Three known scenarios can cause this error:

- The test users are created by running MS.Dynamics.Performance.CreateUsers.exe without any arguments. For example, if the CreateUsers script is run without any arguments, the email addresses of test users that are created won't be correctly formatted. If these users are used to run the tests, the tests will generate the forbidden request error. You can verify that this scenario is causing the error by viewing the users. The incorrect email addresses of the test users will resemble the email addresses in the following illustration.

User ID	User name	Email	Company
$3D8C	$3D8C	TSTBadFormat_5@TAEOfficial.ccsctp.netAdmin (admin@ax7partner.ccsctp.net)	USMF
$4ED9	$4ED9	TSTBadFormat_10@TAEOfficial.ccsctp.netAdmin (admin@ax7partner.ccsctp.net)	USMF
$6CB7	$6CB7	TSTBadFormat_2@TAEOfficial.ccsctp.netAdmin (admin@ax7partner.ccsctp.net)	USMF
$8CE0	$8CE0	TSTBadFormat_9@TAEOfficial.ccsctp.netAdmin (admin@ax7partner.ccsctp.net)	USMF
$B07A	$B07A	TSTBadFormat_4@TAEOfficial.ccsctp.netAdmin (admin@ax7partner.ccsctp.net)	USMF
$C600	$C600	TSTBadFormat_8@TAEOfficial.ccsctp.netAdmin (admin@ax7partner.ccsctp.net)	USMF
$D14B	$D14B	TSTBadFormat_1@TAEOfficial.ccsctp.netAdmin (admin@ax7partner.ccsctp.net)	USMF
$E412	$E412	TSTBadFormat_3@TAEOfficial.ccsctp.netAdmin (admin@ax7partner.ccsctp.net)	USMF
$EDE4	$EDE4	TSTBadFormat_7@TAEOfficial.ccsctp.netAdmin (admin@ax7partner.ccsctp.net)	USMF
$FA4A	$FA4A	TSTBadFormat_6@TAEOfficial.ccsctp.netAdmin (admin@ax7partner.ccsctp.net)	USMF

  To resolve the issue, delete the test users who have incorrectly formatted email addresses. Rerun the CreateUsers script, and specify the user count and company.

- The number of users that you specify in the **UserCount** field in the CloudEnvironment.Config file exceeds the number of test users that you created by using MS.Dynamics.Performance.CreateUsers.exe. Make sure that you created at least as many test users as you request in the CloudEnvironment.Config file.

CloudEnvironment.Config

```
<?xml version="1.0" encoding="utf-8"?>
<EnvironmentalConfigSettings xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <EnvironmentalConfigSettingsCollection>
    <EnvironmentalConfigSetting ConfigName="DEVFABRIC">
      <!-- NOTE: the HostName value needs to be specified -->
      <ExecutionConfigurations Key="HostName" Value="usnconeboxaxlaos.cloud.onebox.dynamics.com" />
      <ExecutionConfigurations Key="SoapHostName" Value="usnconeboxaxlaos.cloud.onebox.dynamics.com" />
      <ExecutionConfigurations Key="SelfSigningCertificateThumbprint" Value="FDFDEFF031CA36B93B25378EAAAD6210825983F2" />
      <ExecutionConfigurations Key="AdminAuthenticatorConfigurationId" Value="SelfMintingAdminUser" />
      <ExecutionConfigurations Key="DefaultBrowser" Value="InternetExplorer" />
      <ExecutionConfigurations Key="FederationRealm" Value="spn:00000015-0000-0000-c000-000000000000" />
      <ExecutionConfigurations Key="DefaultDispatcher" Value="Microsoft.Dynamics.TestTools.Dispatcher.JsDispatcher, Microsoft.Dynamics.TestTools.Dispatcher.JsDispatcher" />
      <ExecutionConfigurationsNodes ConfigurationName="SVC">
        <ConfigurationSpecificDetails Key="AppConfig" Value="DEVFABRIC.Config" />
      </ExecutionConfigurationsNodes>
      <ExecutionConfigurationsNodes ConfigurationName="PRF">
        <ConfigurationSpecificDetails Key="UserCount" Value="10" />
        <ConfigurationSpecificDetails Key="UserFormat" Value="TST_{0}@TAEOfficial.ccsctp.net" />
        <ConfigurationSpecificDetails Key="UserRole" Value="-SYSADMIN-" />
        <ConfigurationSpecificDetails Key="ThinkTime" Value="0" />
        <ConfigurationSpecificDetails Key="Company" Value="USMF" />
      </ExecutionConfigurationsNodes>
    </EnvironmentalConfigSetting>
  </EnvironmentalConfigSettingsCollection>
  <AuthenticatorConfigurationCollection>
    <AuthenticatorConfiguration Id="SelfMintingRunnerUser" Class="MS.Dynamics.TestTools.CloudCommonTestUtilities.Authentication.SelfMintedTokenAuthenticator">
      <Credentials IsFromKeyVault="false" Username="daxrunneruser@daxmdsrunner.com" NetworkDomain="urn:Microsoft:Dynamics:Cloud:DaxRunner" />
    </AuthenticatorConfiguration>
    <AuthenticatorConfiguration Id="SelfMintingSysUser" Class="MS.Dynamics.TestTools.CloudCommonTestUtilities.Authentication.SelfMintedTokenAuthenticator">
      <Credentials IsFromKeyVault="false" Username="testuser@microsoft.com" />
    </AuthenticatorConfiguration>
    <AuthenticatorConfiguration Id="SelfMintingAdminUser" Class="MS.Dynamics.TestTools.CloudCommonTestUtilities.Authentication.SelfMintedTokenAuthenticator">
      <!-- NOTE: admin username needs to be specified -->
      <Credentials IsFromKeyVault="false" Username="Tusr1@TAEOfficial.ccsctp.net" />
    </AuthenticatorConfiguration>
  </AuthenticatorConfigurationCollection>
</EnvironmentalConfigSettings>
```

- If your Finance and Operations apps were deployed in 21Vianet, make sure that your development and performance testing environments are in Platform Update for 10.0.11 or above.

At least one security token in the message could not be validated

This issue can occur when you run multi-user tests, when you create users by using MS.Dynamics.Performance.CreateUsers.exe, when the AOS machine differs from the developer machine.

Error example - At least one security token in the message could not be validated

> System.TypeInitializationException: The type initializer for 'MS.Dynamics.TestTools.CloudCommonTestUtilities.Authentication.UserManagement' threw an exception. ---> System.ServiceModel.Security.MessageSecurityException: An unsecured or incorrectly secured fault was received from the other party. See the inner FaultException for the fault code and detail. ---> System.ServiceModel.FaultException: At least one security token in the message could not be validated.

Solution - At least one security token in the message could not be validated

This issue occurs when the AOS endpoint can't validate the thumbprint of the certificate that you created. There are two possible causes:

- The certificate wasn't installed on the AOS machine. To fix the issue, copy a .cer file to the AOS machine, and install it.
- The thumbprint of the certificate wasn't added to the wif.config file on the AOS machine. To fix the issue, add the certificate to the wif.config file. Be sure to restart Microsoft Internet Information Services (IIS) after you change the wif.config file.

MS.Dynamics.Test.Team.Foundation.WebClient.InteractionService.dll.config is missing from the deployment items

This issue usually occurs when you run load tests.

Error example - MS.Dynamics.Test.Team.Foundation.WebClient.InteractionService.dll.config is missing

> <Test class name>.TestSetup threw exception. System.InvalidOperationException: System.InvalidOperationException: Could not find endpoint element with name 'ClientCommunicationManager' and contract 'Microsoft.Dynamics.Client.InteractionService.Communication.Reliable.IReliableCommunicationManager' in the ServiceModel client configuration section. This might be because no configuration file was found for your application, or because no endpoint element matching this name could be found in the client element.. at System.ServiceModel.Description.ConfigLoader.LoadChannelBehaviors(ServiceEndpoint serviceEndpoint,

> String configurationName)

Solution - MS.Dynamics.Test.Team.Foundation.WebClient.InteractionService.dll.config is missing

This issue occurs when the system can't find the MS.Dynamics.Test.Team.Foundation.WebClient.InteractionService.dll.config file when the load tests are run, because the file wasn't added as a deployment item. Verify that the MS.Dynamics.Test.Team.Foundation.WebClient.InteractionService.dll.config file is in the Out folder for the test run:

<solution path>\TestResults\<your test run>\Out

If the file is missing, add it to the deployment items in the test settings.

There are two files that have very similar names. The name of one file ends in *.dll, and the name of the other file ends in *.dll.config. The *.dll.config file must be in the deployment items in the test settings.

CloudEnvironment.Config is missing from the deployment items

This issue usually occurs only when you run load tests.

Error example - CloudEnvironment.Config is missing

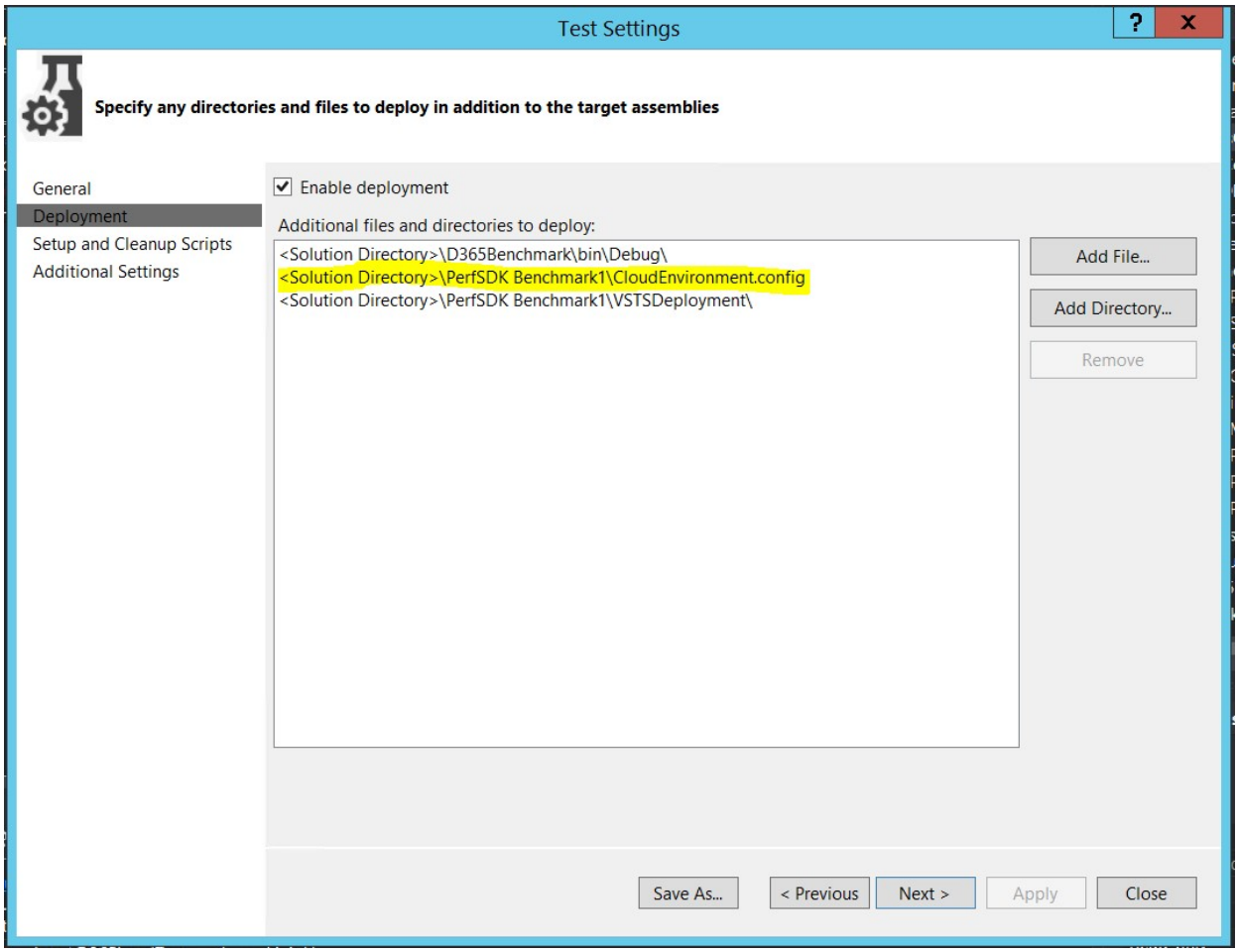
> Initialization method \<Test class name>.TestSetup threw exception. System.TypeInitializationException: System.TypeInitializationException: The type initializer for 'MS.Dynamics.TestTools.CloudCommonTestUtilities.Authentication.UserManagement' threw an exception. ---> MS.Dynamics.TestTools.TestLogging.EvaluateException: Assert.Fail failed. DateTime="10/13/2017 14:42:55" "The type initializer for 'MS.Dynamics.TestTools.CloudCommonTestUtilities.Authentication.SecretSettingsHelper' threw an exception.".

Solution - CloudEnvironment.Config is missing

This issue occurs when the CloudEnvironment.Config file isn't present when the tests are run. It typically occurs when you run load tests and the CloudEnvironment.Config file wasn't added as a deployment item. Verify that the CloudEnvironment.Config file is in the Out folder for the test run:

<solution path>\TestResults\<your test run>\Out

If the file is missing, add it to the deployment items in the test settings.



InteractiveClientId was not specified in the settings

Error example - InteractiveClientId was not specified in the settings

> The type initializer for 'MS.Dynamics.TestTools.CloudCommonTestUtilities.Authentication.SecretSettingsHelper' threw an exception. ---> Microsoft.CE.VaultSDK.SecretProviderException: InteractiveClientId was not specified in settings.

Solution - InteractiveClientId was not specified in the settings

This issue occurs when the **SelfSigningCertificateThumbprint** field is left blank in the CloudEnvironment.Config file. In the CloudEnvironment.Config file, find the following line, and paste in the thumbprint of the certificate that you created and installed.

```
\<ExecutionConfigurations Key="SelfSigningCertificateThumbprint" Value="" />
```

The remote host forcibly closed an existing connection

Error example - The remote host forcibly closed an existing connection

> System.TypeInitializationException: System.TypeInitializationException: The type initializer for 'MS.Dynamics.TestTools.CloudCommonTestUtilities.Authentication.UserManagement' threw an exception. ---> System.ServiceModel.CommunicationException: An error occurred while making the HTTP request to \<Host name>/Services/AxUserManagement/Service.svc/ws2007FedHttp. This could be due to the fact that the server certificate is not configured properly with HTTP.SYS in the HTTPS case. This could also be caused by a mismatch of the security binding between the client and the server. ---> System.Net.WebException: The underlying connection was closed: An unexpected error occurred on a send. ---> System.IO.IOException: Unable to read data from the transport connection: An existing connection was forcibly closed by the remote

> host. ---> System.Net.Sockets.SocketException: An existing connection was forcibly closed by the remote host.

Solution - The remote host forcibly closed an existing connection

Run the following Windows PowerShell script on the development machine.

```
Set-ItemProperty HKLM:\SOFTWARE\Microsoft\.NETFramework\v4.0.30319 -Name SchUseStrongCrypto -Value 1 -Type
dword -Force -Confirm:$false
if ((Test-Path HKLM:\SOFTWARE\Wow6432Node\Microsoft\.NETFramework\v4.0.30319))
{
    Set-ItemProperty HKLM:\SOFTWARE\Wow6432Node\Microsoft\.NETFramework\v4.0.30319 -Name SchUseStrongCrypto
-Value 1 -Type dword -Force -Confirm:$false
}
```

Service w3svc was not found on computer

This error occurs only when you run load tests by using Visual Studio Online.

Error example - Service w3svc was not found on computer

> Test method MS.Dynamics.Performance.Application.GFM.PDLTrend.ProcureToPayTrend.ProcureToPaymentTrend threw exception: System.TypeInitializationException: The type initializer for 'MS.Dynamics.TestTools.CloudCommonTestUtilities.Authentication.UserManagement' threw an exception. ---> System.InvalidOperationException: Service w3svc was not found on computer '.'. ---> System.ComponentModel.Win32Exception: The specified service does not exist as an installed service.

Solution - Service w3svc was not found on computer

A hotfix is available that resolves this issue. The Microsoft Knowledge Base (KB) number is 4095640.

The file IEDriverServer.exe does not exist

This issue affects only single-user tests.

Error example - The file IEDriverServer.exe does not exist

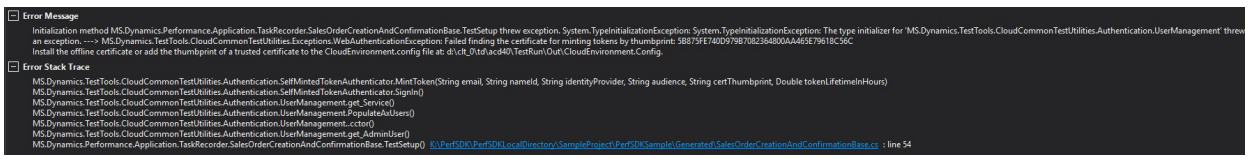
> The file K:\perfSDK\PerfSDKLocalDirectory\SampleProject\TestResults\Admin501201994c_devae648d1909-1 2018-06-25 03_40_51\Out\Common\External\Selenium\IEDriverServer.exe does not exist. The driver can be downloaded at `https://selenium-release.storage.googleapis.com/index.html`.

Solution - The file IEDriverServer.exe does not exist

Copy the **Common\External\Selenium** folder under **<Your_PerfSDK_Folder>** to the **<Your_PerfSDK_Folder>\SampleProject\ PerfSDKSample\bin\Debug** folder.

Failed finding the certificate for minting tokens by thumbprint: <your certificate thumbprint>

Error example - Failed finding the certificate for minting tokens by thumbprint



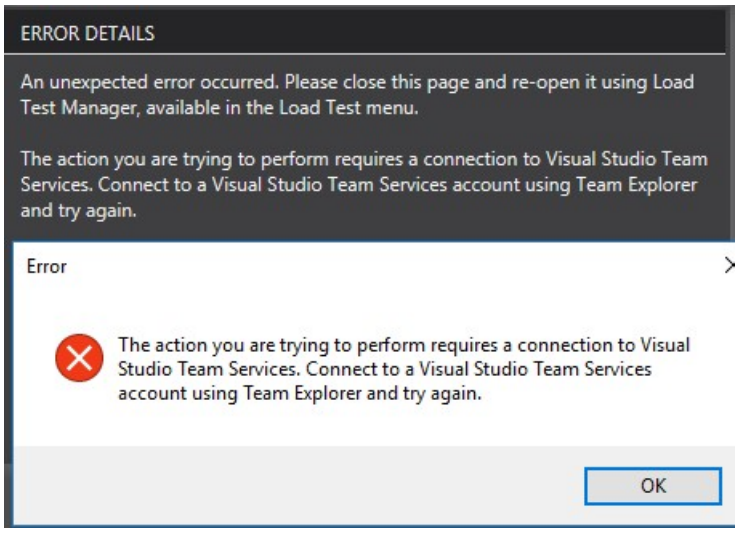
Solution - Failed finding the certificate for minting tokens by thumbprint

Make sure that you install the generated certificate on each AOS machine in your sandbox environment.

The action you are trying to perform requires a connection to Visual Studio Team Services

This issue affects only multi-user tests.

Error example - The action you are trying to perform requires a connection to Visual Studio Team Services



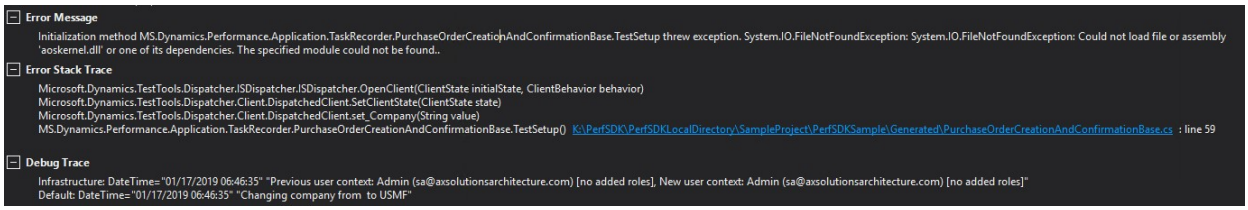
Solution - The action you are trying to perform requires a connection to Visual Studio Team Services

When you connect to Azure DevOps, use the old URI format (**<Azure_DevOps_Account>.visualstudio.com**) instead of **dev.azure.com/<Azure_DevOps_Account>**. Additionally, open Azure DevOps by using the old URI, and then select **Open in Visual Studio**.

Could not load file or assembly 'aoskernel.dll' or one of its dependencies

This error affects only multi-user tests.

Error example - Could not load file or assembly 'aoskernel.dll'



Solution - Could not load file or assembly 'aoskernel.dll'

Make sure that you're using Open Database Connectivity (ODBC) Driver 17 in an environment that has Platform update 20 or later.

AzureActiveDirectoryConfiguration node is missing in CloudEnvironment.config

Error example - AzureActiveDirectoryConfiguration node is missing

> Initialization method MS.Dynamics.Performance.Application.TaskRecorder.SalesOrderCreationAndConfirmationBase.TestSetup threw exception. System.TypeInitializationException: System.TypeInitializationException: The type initializer

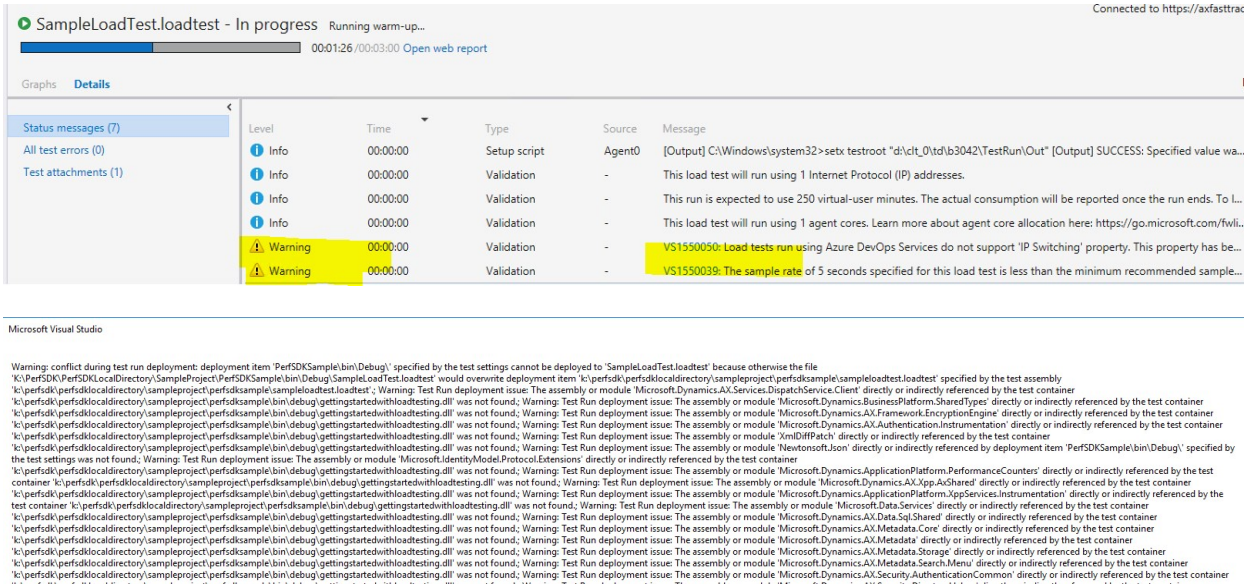
> for 'MS.Dynamics.TestTools.CloudCommonTestUtilities.Authentication.UserManagement' threw an exception. ---> System.Reflection.TargetInvocationException: Exception has been thrown by the target of an invocation. ---> System.MissingFieldException: AzureActiveDirectoryConfiguration node is missing in CloudEnvironment.config.

Solution - AzureActiveDirectoryConfiguration node is missing

Replace all instances of **"MS.Dynamics.TestTools.CloudCommonTestUtilities.Authentication.AadAuthenticator"** with **"MS.Dynamics.TestTools.CloudCommonTestUtilities.Authentication.SelfMintedTokenAuthenticator"** in the **AuthenticatorConfigurationCollection** section of the CloudEnvironment.config file.

Multiple warning messages before and after multi-user testing that uses Azure DevOps

Error example - Multiple warning messages before and after multi-user testing



]

Solution - Multiple warning messages before and after multi-user testing

There is no impact, and the messages can be ignored.

The type or namespace name 'xxxx' could not be found (are you missing a using directive or an assembly reference?)

Error example - The type or namespace name 'xxxx' could not be found

The type or namespace name 'InventTransferOrders' could not be found (are you missing a using directive or an assembly reference?)

Solution - The type or namespace name 'xxxx' could not be found

The sample solution shipped with the perfSDK was previously prepared and wasn't updated after the packages split. To resolve the issue, add the assembly **MS.Dynamics.TestTools.DirectoryProxyLibrary.dll** under <Service volume>:\PerfSDK\PerfSDKLocalDirectory as a reference.

Assembly was built against the ".NETFramework,Version=v4.6" framework

Error example - Assembly was built against the ".NETFramework,Version=v4.6" framework

The primary reference "MS.Dynamics.TestTools.ApplicationSuiteProxyLibrary" could not be resolved because it has an indirect dependency on the assembly "MS.Dynamics.TestTools.DirectoryProxyLibrary, Version=7.0.0.0, Culture=neutral, PublicKeyToken=a7cf325ee2c8a9ff" which was built against the ".NETFramework,Version=v4.6" framework. This is a higher version than the currently targeted framework ".NETFramework,Version=v4.5".

Solution- Assembly was built against the ".NETFramework,Version=v4.6" framework

Change the **Target framework** property in the properties window of PerfSDKSample to .Net Framework 4.6.

Performance SDK and multiuser testing in on-premises environments

2/18/2021 • 17 minutes to read • Edit Online

This topic explains how to use the Performance software development kit (SDK) to do multiuser load testing in an on-premises environment.

> **IMPORTANT**
>
> Visual Studio 2019 will be the last version of Visual Studio with web performance and load test features. In the future, we will be publishing some recommendations for alternative solutions.
>
> - If you are using the Visual Studio and Test Controller/Test Agent for on-premises load testing, Visual Studio 2019 will be the last version. You can continue using it until the end of support cycle.
> - If you are using the cloud-based load testing service, the cloud-based load testing service will continue to run through March 31, 2020. Until then, you can continue to use all of the experiences powered by this service without interruption. Alternatively, you can switch to on-premises load testing.
>
> For more information, see Cloud-based load testing service end of life.

Prerequisites

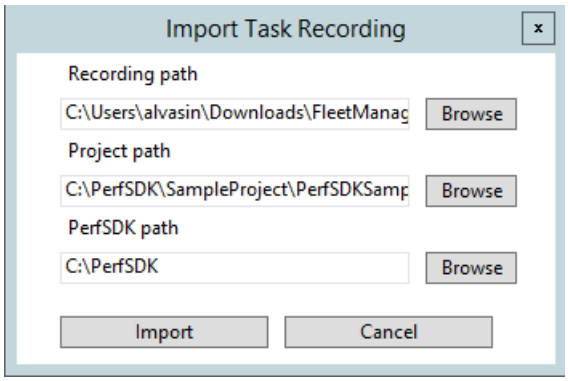
- An on-premises environment that has volume data
- A development environment that has the following characteristics:
  - Microsoft Visual Studio Enterprise or a later version is installed.
  - The Performance SDK is installed. (The SDK will likely be in K:\PerfSDK\PerfSDKLocalDirectory. However, depending on your environment, it might be in another location, such as C:\PerfSDK.)
  - The on-premises environment can be accessed in a web browser. (The development virtual machine [VM] might be in the same domain as the on-premises environment, or the on-premises environment might have a publicly registered domain name.)

Create a single-user C# test from an XML recording

1. Use Task recorder to create a recording of the scenario that you want to test.

   > **IMPORTANT**
   >
   > Your task recording must start on the default dashboard page. Otherwise, the test won't be able to run.

2. Start Microsoft Visual Studio as an administrator, and build the **PerfSDKSample** project. This project is in the **PerfSDK** folder. If you've already built the project, skip this step.
3. Select **Dynamics 365** > **Addins** > **Create C# perf test from recording**.
4. In the **Import Task Recording** dialog box, enter the required details, and then select **Import**.



A C# test is generated in the Generated folder for the project that you selected.

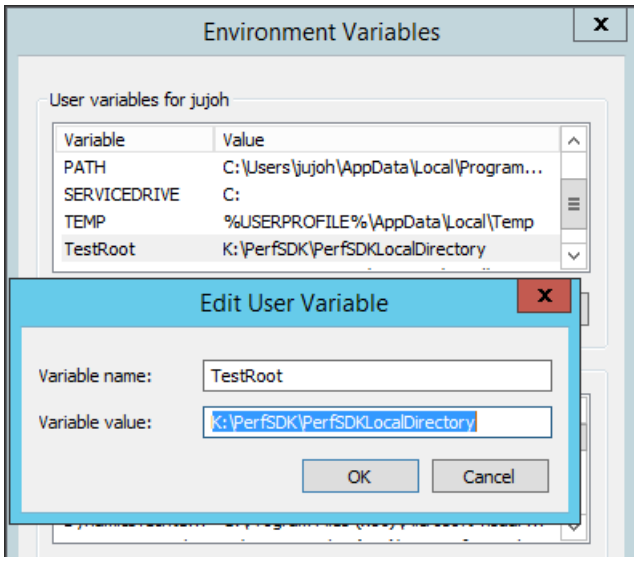
> **NOTE**
> The test that is generated might have to be edited to resolve any compilation issues.

Run a single-user test by using the Performance SDK

Prepare the development environment

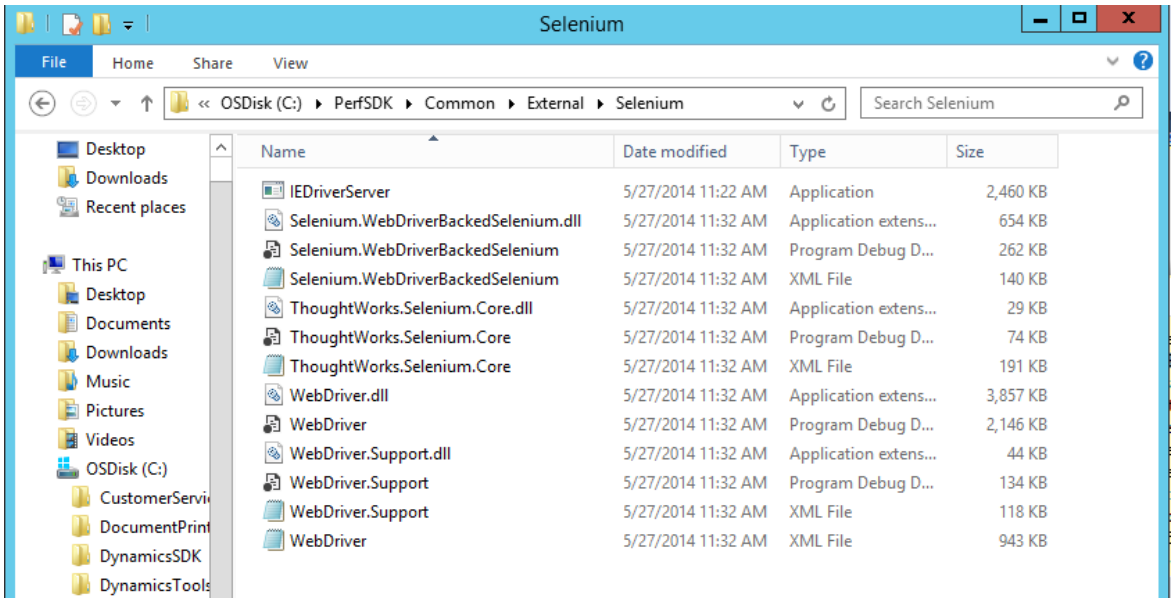
Follow these steps in the development environment.

1. In Control Panel in Microsoft Windows, select **System and Security** > **System** > **Advanced System Settings**. Verify that the **TestRoot** environment variable is set to the path of the PerfSDK folder.



2. Download the **selenium-dotnet-strongnamed-2.42.0.zip** and **IEDriverServer_Win32_2.42.0.zip** files from https://selenium-release.storage.googleapis.com/index.html?path=2.42/, and extract the files.

3. Copy the dynamic-link libraries (DLLs) from the **selenium-dotnet-strongnamed-2.42.0.zip\net40** folder to the **PerfSDK\Common\External\Selenium** folder. Also copy the **IEDriverServer.exe** from the **IEDriverServer_Win32_2.42.0.zip** to the **PerfSDK\Common\External\Selenium** folder.



4. Generate a certificate to use for authentication for the tests. To generate a certificate file, open a Command Prompt window as an administrator, and run the following commands. When you're prompted for a private key password, select **None**.

   ```
   "C:\Program Files (x86)\Windows Kits\8.1\bin\x64\makecert" -n "CN=127.0.0.1" -ss Root -sr
   LocalMachine -a sha256 -len 2048 -cy end -r -eku 1.3.6.1.5.5.7.3.1 -sv c:\temp\authcert.pvk
   c:\temp\authcert.cer

   "c:\Program Files (x86)\Windows Kits\8.1\bin\x64\pvk2pfx" -pvk c:\temp\authCert.pvk -spc
   c:\temp\authcert.cer -pfx c:\temp\authcert.pfx
   ```

   Note the following elements in these commands:

   - **-n "CN=127.0.0.1"** gives a human-readable name to the certificate. The name of this certificate must be **127.0.0.1**. Otherwise, the single-user tests won't be able to run.
   - **-eku 1.3.6.1.5.5.7.3.1** gives the purpose of the certificate. It indicates that the certificate can be used as a Secure Sockets Layer (SSL) server certificate.

   After the script has finished running, you should see the following files in **C:\Temp**:

   - authcert.pfx
   - authcert.cer
   - authcert.pvk
5. Install the **authcert.pfx** certificate file. When you install the file, make sure that you select **Local Machine**.
6. Copy the **authcert.pfx** file to the **PerfSDK** folder.
7. Open a Microsoft Windows PowerShell window as an administrator, and run the following commands to get the thumbprint of the installed certificate.

   ```
   cd Cert:\LocalMachine\My

   Get-ChildItem | Where-Object { $_.Subject -like "CN=127.0.0.1" }
   ```
8. In Visual Studio, open the **PerfSDKSample** project that is in the **PerfSDK** folder.
9. In the Visual Studio project, add a reference to the **WebDriver.dll** file in the **PerfSDK\Common\External\Selenium** folder.

10. Open the **CloudEnvironment.Config** file, and replace the contents with the following template.

```
<?xml version="1.0" encoding="utf-8"?>
<EnvironmentalConfigSettings xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <EnvironmentalConfigSettingsCollection>
        <EnvironmentalConfigSetting ConfigName="DEVFABRIC">
            <!-- NOTE: the HostName value needs to be specified -->
            <ExecutionConfigurations Key="HostName" Value="[yourD365FOdomain]/namespaces/AXSF" />
            <ExecutionConfigurations Key="SoapHostName" Value="[yourD365FOdomain]/namespaces/AXSF" />
            <ExecutionConfigurations Key="SelfSigningCertificateThumbprint" Value="
[ThumbprintFromPowerShell]" />
            <ExecutionConfigurations Key="AdminAuthenticatorConfigurationId"
Value="SelfMintingAdminUser" />
            <ExecutionConfigurations Key="DefaultBrowser" Value="InternetExplorer" />
            <ExecutionConfigurations Key="FederationRealm" Value="spn:00000015-0000-0000-c000-
000000000000" />
            <ExecutionConfigurations Key="DefaultDispatcher"
Value="Microsoft.Dynamics.TestTools.Dispatcher.JsDispatcher,
Microsoft.Dynamics.TestTools.Dispatcher.JsDispatcher" />
            <ExecutionConfigurationsNodes ConfigurationName="SVC">
                <ConfigurationSpecificDetails Key="AppConfig" Value="DEVFABRIC.Config" />
            </ExecutionConfigurationsNodes>
            <ExecutionConfigurationsNodes ConfigurationName="PRF">
                <ConfigurationSpecificDetails Key="IsAdfs" Value="True" />
                <ConfigurationSpecificDetails Key="UserCount" Value="2" />
                <ConfigurationSpecificDetails Key="UserFormat" Value="[AdminUserEmail]" />
                <ConfigurationSpecificDetails Key="UserPassword" Value="[AdminUserPassword]" />
                <ConfigurationSpecificDetails Key="UserRole" Value="-SYSADMIN-" />
                <ConfigurationSpecificDetails Key="ThinkTime" Value="0" />
                <ConfigurationSpecificDetails Key="Company" Value="USMF" />
            </ExecutionConfigurationsNodes>
        </EnvironmentalConfigSetting>
    </EnvironmentalConfigSettingsCollection>
    <AuthenticatorConfigurationCollection>
        <AuthenticatorConfiguration Id="SelfMintingRunnerUser"
Class="MS.Dynamics.TestTools.CloudCommonTestUtilities.Authentication.SelfMintedTokenAuthenticator">
            <Credentials IsFromKeyVault="false" Username="daxrunneruser@daxmdsrunner.com"
NetworkDomain="urn:Microsoft:Dynamics:Cloud:DaxRunner" />
        </AuthenticatorConfiguration>
        <AuthenticatorConfiguration Id="SelfMintingSysUser"
Class="MS.Dynamics.TestTools.CloudCommonTestUtilities.Authentication.SelfMintedTokenAuthenticator">
            <Credentials IsFromKeyVault="false" Username="testuser@microsoft.com" />
        </AuthenticatorConfiguration>
        <AuthenticatorConfiguration Id="SelfMintingAdminUser"
Class="MS.Dynamics.TestTools.CloudCommonTestUtilities.Authentication.SelfMintedTokenAuthenticator">
            <!-- NOTE: admin username needs to be specified -->
            <Credentials IsFromKeyVault="false" Username="[AdminUserEmail]" NetworkDomain="[AdfsUrl]"
/>
        </AuthenticatorConfiguration>
    </AuthenticatorConfigurationCollection>
</EnvironmentalConfigSettings
```

11. In the **CloudEnvironment.Config** file, specify values for the following keys. These values replace the placeholder values in square brackets in the template.

    - **HostName** – Specify the URL that is used to access your on-premises environment. The URL should be **[yourD365FOdomain]/namespaces/AXSF**.
    - **SoapHostName** – Specify the same URL that you specified for **HostName**.
    - **SelfSigningCertificateThumbprint** – Specify the thumbprint that you retrieved from Windows PowerShell in step 7.
    - **UserFormat** – Specify the email address of a user who has the System Administrator role in your

on-premises environment. The user must be an Active Directory Federation Services (AD FS) user.

- **UserPassword** – Specify the password of the user whose email address you specified for **UserFormat**.
- **Username** – Specify the same email address that you specified for **UserFormat**.
- **NetworkDomain** – Specify the URL of the AD FS identity provider. You can find this value by running the following SQL query against the **AXDB** database of your on-premises deployment.

```
select NETWORKDOMAIN, NETWORKALIAS from USERINFO where NETWORKALIAS='[AdminUserEmail]'
```

Prepare the on-premises environment

Follow these steps on each Application Object Server (AOS) VM in the on-premises deployment.

1. Copy the **authcert.cer** file that you created in the Prepare the development environment section of this topic to the AOS VM.
2. Install the **authcert.cer** certificate file. When you install the certificate, make sure that you select **Local Machine**. Also make sure that you put the certificate in the **Trusted Root Certification Authorities** store.
3. Open the **wif.config** file in a text editor. The path of the file will resemble **C:\ProgramData\SF\AOS1\Fabric\work\Applications\AXSFType_App19\AXSF.Code.1.0.20180717001108**.

> **NOTE**
> In the file path, the AOS number (**AOS1** in this example) will vary, depending on the AOS node that you're on. Additionally, the **ProgramData** folder is a hidden folder. Therefore, to see the folder in File Explorer, you must enable hidden items.

4. In the **wif.config** file, find the authority that is named `https://fakeacs.accesscontrol.windows.net`. In the list of thumbprints for this authority, add the thumbprint of the certificate that you created in the Prepare the development environment section. In the following example, the fourth thumbprint has been added to the `https://fakeacs.accesscontrol.windows.net` authority.

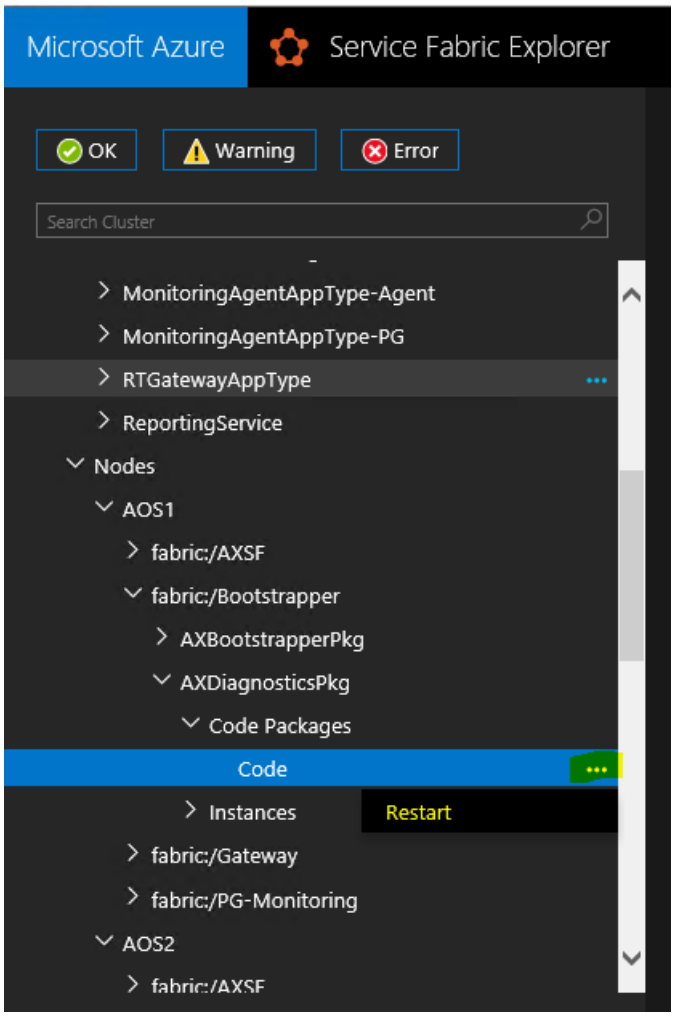
```
<authority name="https://fakeacs.accesscontrol.windows.net/">
    <keys>
        <add thumbprint="9567B0F32F4B312FEE44ACE88A9CAAA13B7FBB86" />
        <add thumbprint="B8F659E2FDD2A6811CD72BE753FF7ACB64351AC1" />
        <add thumbprint="B51C394E6EE9578D8C54AE0A9927D8B6DCEFF84B" />
        <add thumbprint="F9112DE3D4DE65CBE39601EBDC7FBB1F8F525DED" />
    </keys>
    <validIssuers>
        <add name="https://fakeacs.accesscontrol.windows.net/" />
    </validIssuers>
</authority>
```

5. Open the **AXService.exe.config** file in a text editor. You can find this file in the same directory as the **wif.config** file.
6. Search the **AXService.exe.config** file for **"Aos.AosRole"**. Replace the line that contains **"Aos.AosRole"** with the following line.

```
<add key="Aos.AosRole" value="AosRoleUnknown" />
```

> **NOTE**
>
> Setting the **AosRole** to **AosRoleUnknown** disables the limit on the number of web sessions per user. This is necessary to complete load testing with a large user load because the load test will create many sessions for a single user. Once you are finished with your load testing, reset this value to **"AosRoleWeb"**.

7. In Service Fabric Explorer, find the **Code** package for the AOS node, select the ellipse button (...), and then select **Restart** to restart the application.



Run the single-user test

1. In the **PerfSDKSample** project, find the **PurchaseReq.cs** file. This file is a sample single-user test. In the file, comment out the following lines.

```
if (this.TestContext !=null)
{
    timerProvider = new TimerProvider(this.TestContext);
}
```

```
private TimerProvider timerProvider;
[TestInitialize]
0 references
public void TestSetup()
{
    //if (this.TestContext != null)
    //{
    //    timerProvider = new TimerProvider(this.TestContext);
    //}
```

2. Select **Test** > **Test settings**, set the **Default processor architecture** field to **x64**, and then build the solution.
3. Select **Test** > **Windows** > **Test Explorer** to view the list of tests.

> **NOTE**
> Sometimes, Visual Studio might not update the list of tests after you create a test script from a task recording. In this case, restart Visual Studio, and then reopen Test Explorer.

4. Run the sample single-user test by right-clicking **CreatePurchReq**. Alternatively, you can run the test that you created from your task recording. When you run the test, Internet Explorer should be started, and it should replay the scenario that you recorded.

Run a multiuser load test by using the Performance SDK

Create a multiuser test from a single-user test

After you create a single-user test by using the information earlier in this topic, you can convert it to a multiuser test. Add **MS.Dynamics.TestTools.UIHelpers.Core;** to your test script, and find the following line in the **TestSetup** method.

```
Client = DispatchedClient.DefaultInstance;
```

Replace that line with the following lines.

```
DispatchedClientHelper helper = new DispatchedClientHelper();
Client = helper.GetClient();
```

The test script that was generated by the Task Importer might contain a line that resembles the following line.

```
UserContextRole _context = new UserContextRole(UserManagement.AdminUser);
```

Remove this line from any tests that will be run as load tests. This code is required only for single-user tests and has a negative effect on the performance of load tests.

Make sure that the values that you entered when you made the task recording are randomized.

Run the multiuser load test

1. In the Visual Studio editor, open the **ProcureToPay.cs** file, and append the following lines in the **TestSetup** method.

```
var testroot = System.Environment.GetEnvironmentVariable("DeploymentDir");
if (string.IsNullOrEmpty(testroot))
{
    testroot = System.IO.Directory.GetCurrentDirectory();
}
Environment.SetEnvironmentVariable("testroot", testroot);
```

2. Download the installer (.msi) file for Microsoft ODBC Driver 13 for SQL Server from https://www.microsoft.com/download/details.aspx?id=50420. (Select the 64-bit version of the .msi file.) Put the file in the **Visual Studio Online** folder in the **PerfSDK** directory.

3. Modify the contents of the **setup.cmd** file in the **Visual Studio Online** folder so that they match the following code.

```
setx testroot "%DeploymentDirectory%"
ECHO Installing D365 prerequisites
ECHO MSIEXEC /a %DeploymentDirectory%\msodbcsql /passive /norestart IACCEPTMSODBCSQLLICENSETERMS=YES
MSIEXEC /a %DeploymentDirectory%\msodbcsql /passive /norestart IACCEPTMSODBCSQLLICENSETERMS=YES
%windir%\sysnative\windowspowershell\v1.0\powershell.exe -File %DeploymentDirectory%\install-wif.ps1
Md %DeploymentDirectory%\Common\Team\Foundation\Performance\Framework
%DeploymentDirectory%\CloudCtuFakeACSInstall.cmd %DeploymentDirectory%\authcert.pfx
```

4. Modify the contents of the **CloudCtuFakeACSInstall.cmd** file so that the **Import** command has an empty string instead of '**password**'. The third line of the script should resemble the following line.

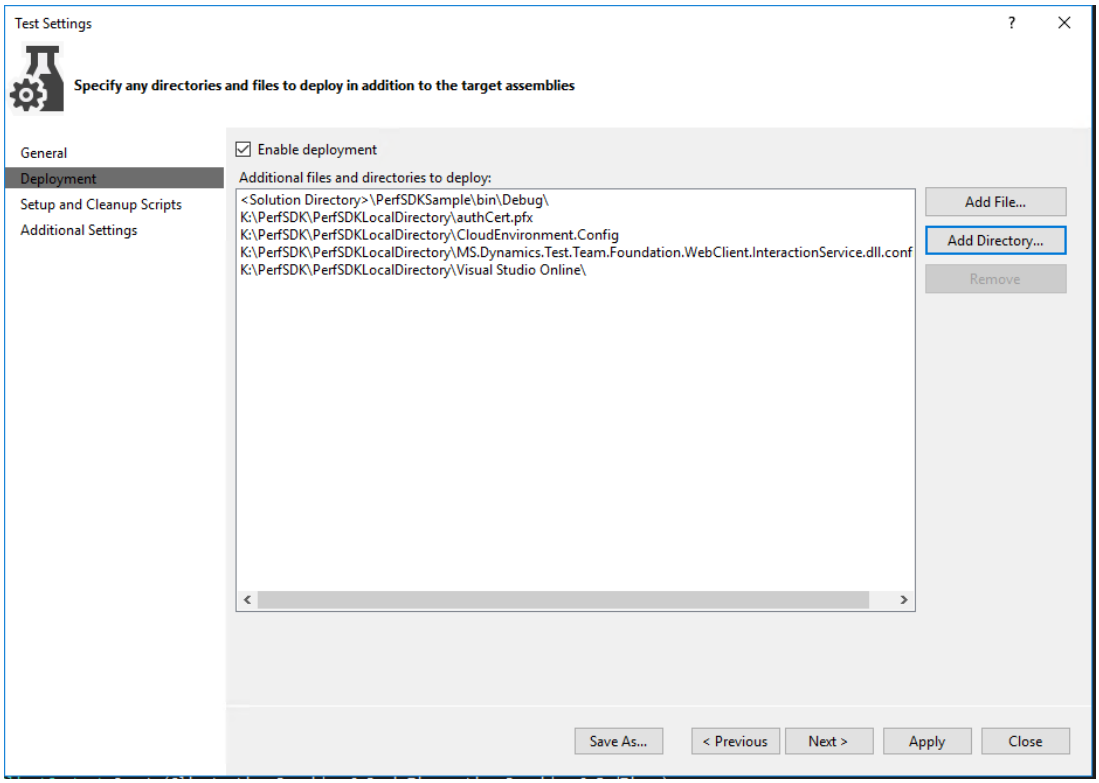
```
set MyStoreInstallCmd= .... $pfxcert.Import('%TestCertPath%', '', 'Exportable,PersistKeySet').... 
```

5. In your solution files, double-click the **vsonline.testsettings** file to modify the test settings.

6. In the **Test Settings** dialog box, on the **General** tab, set the **Test run location** field to **Run tests using local computer or a test controller**.

7. On the **Deployment** tab, use the following settings:

   - Select the **Enable deployment** check box.
   - In the **Additional files and directories to deploy** field, make sure that the following files and directories are listed:
     - <Solution Directory>\PerfSDKSample\bin\Debug\
     - C:\PerfSDK\CloudEnvironment.Config
     - C:\PerfSDK\authcert.pfx
     - C:\PerfSDK\MS.Dynamics.Test.Team.Foundation.WebClient.InteractionService.dll.config
     - C:\PerfSDK\Visual Studio Online\

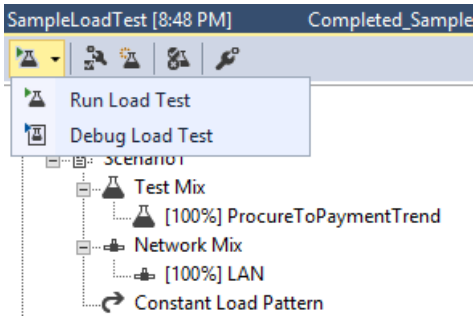


> **NOTE**
>
> Your PerfSDK folder might differ.

8. On the **Setup and Cleanup Scripts** tab, select the **setup.cmd** file that is in the **Visual Studio Online** folder in the **PerfSDK** directory.

9. On the **Hosts** tab, select **Run tests in 64 bit process on 64 bit machine**.

10. To run the test, open the **SampleLoadTest.loadtest** file, and select **Run Load Test**.



When the test has finished running, you should see a summary that shows transaction results. Here is an example.

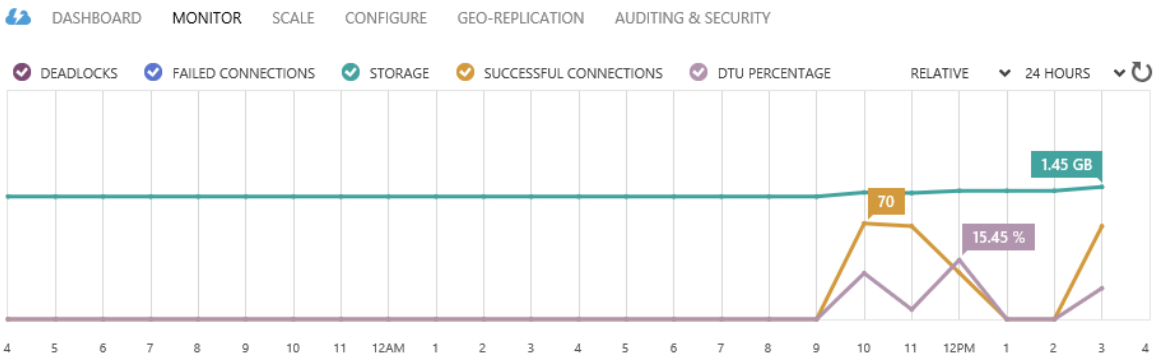
Test Results

Name	Scenario	Total Tests	Failed Tests (% of total)	Avg. Test
ProcureToPaymentTrend	Scenario1	10	10 (100)	56.6

Transaction Results

Name	Scenario	Test	Response Time (sec)
000_Navigate_purchtablelistpage	Scenario1	ProcureToPaymentTrend	1.56
002_PurchTable_SystemDefinedNewButton_Click	Scenario1	ProcureToPaymentTrend	0.44
004_PurchCreateOrder_VendorTab_Dispatcher_ActivateTab	Scenario1	ProcureToPaymentTrend	0.33
028_PurchTable_LineViewLines_Dispatcher_ActivateTab	Scenario1	ProcureToPaymentTrend	0.22
026_PurchTable_TabPageDetails_Dispatcher_ActivateTab	Scenario1	ProcureToPaymentTrend	0.22
014_PurchTable_TabPageDetails_Dispatcher_ActivateTab	Scenario1	ProcureToPaymentTrend	0.22
027_PurchTable_LineView_Dispatcher_ActivateTab	Scenario1	ProcureToPaymentTrend	0.11

11. To view various indicators for the test controller and test scenario, you can switch to the **Graphs** view.



> **NOTE**
>
> While tests are being run, information about your system isn't available in this view. To access this information, you must use Microsoft Dynamics Lifecycle Services (LCS) to monitor the CPU and memory usage of your AOS machine. Alternatively, you can set up perfmon directly on the AOS machine and set up the Microsoft Azure portal to monitor Microsoft SQL Server usage of Database Transaction Units (DTUs).

Troubleshooting

Zoom factor

This issue affects only single-user tests.

Error example

```
Initialization method <Test class name>.TestSetup threw exception. System.InvalidOperationException:
System.InvalidOperationException: Unexpected error launching Internet Explorer. Browser zoom level was set
to 200%. It should be set to 100% (NoSuchDriver).
```

Solution

In Internet Explorer, you can change the zoom factor to 100 percent by changing the following registry keys:

- Computer\HKEY_CURRENT_USER\SOFTWARE\Microsoft\Internet Explorer\Zoom\ResetZoomOnStartup = 0
- Computer\HKEY_CURRENT_USER\SOFTWARE\Microsoft\Internet Explorer\Zoom\ResetZoomOnStartup2 = 0
- Computer\HKEY_CURRENT_USER\SOFTWARE\Microsoft\Internet Explorer\Zoom\Zoomfactor = 80000

Depending on the version of the local machine that is used, before you start the Remote Desktop Protocol (RDP) session, you might have to select **Change the size of text, apps and other items**. This field is available in **Display settings** in Windows.

If those steps don't work, try to change the size of your remote desktop before you start the RDP session, so that the default zoom level in Internet Explorer is 100 percent.

Certificate thumbprint errors

Error example

```
Initialization method MS.Dynamics.Performance.Application.TaskRecorder.TestRecord1Base.TestSetup threw
exception.
System.TypeInitializationException: System.TypeInitializationException: The type initializer for
'MS.Dynamics.TestTools.CloudCommonTestUtilities.Authentication.UserManagement' threw an exception. -->
MS.Dynamics.TestTools.CloudCommonTestUtilities.Exceptions.WebAuthenticationException:
Failed finding the certificate for minting tokens by thumbprint: b4f01d2fc42718198852cd23957fc60a3e4bca2e
```

Solution

You might receive the error message for several reasons:

- The certificate thumbprint that you copied into the CloudEnvironment.Config and wif.config files includes invisible Unicode characters. To determine whether the thumbprint contains invisible Unicode characters, paste it into a Unicode code converter, and see whether extra characters appear in the **HTML/XML** field. For example, you can use the Unicode converter that is available at https://r12a.github.io/apps/conversion/.

- The certificate wasn't installed correctly on the AOS machine. To verify that the certificate can be found on the AOS machine, run the following Windows PowerShell script.

```
cd Cert:\LocalMachine\My
Get-ChildItem | Where-Object { $_.Subject -like "CN=127.0.0.1" }
```

  If the thumbprint doesn't appear in the Windows PowerShell console after you run the script, the certificate can't be found. To fix the issue, copy and install the .cer file that you created earlier in this topic to the AOS machine.

- If this issue occurs when you run load tests, the setup scripts might not have installed the corresponding .pfx file correctly. Verify that the password that is specified in the CloudCtuFakeACSInstall.cmd file matches the password that was set when the certificate was created.

No endpoint is listening

Error example

The tests process fails, and the following error message is shown.

```
System.TypeInitializationException: The type initializer for
'MS.Dynamics.TestTools.CloudCommonTestUtilities.Authentication.UserManagement' threw an exception. --->
System.ServiceModel.EndpointNotFoundException: There was no endpoint listening at <web address> that could
accept the message. This is often caused by an incorrect address or SOAP action.
```

Solution

This issue occurs when the host that is specified in the CloudEnvironment.Config file can't be accessed from the machine that is trying to run the tests or create users.

In the CloudEnvironment.Config file, review the values that are specified for the following keys:

- <ExecutionConfigurations Key="HostName" Value="<web address of host>" />
- <ExecutionConfigurations Key="SoapHostName" Value="<web address of SOAP>" />

The web addresses that are specified by these keys must be the environment that you're testing. In a web browser on your developer machine, make sure that you can open the web address that is specified for the **HostName** key.

Users can't be enumerated

This issue can occur when you run multiuser tests, or when you create users by using MS.Dynamics.Performance.CreateUsers.exe.

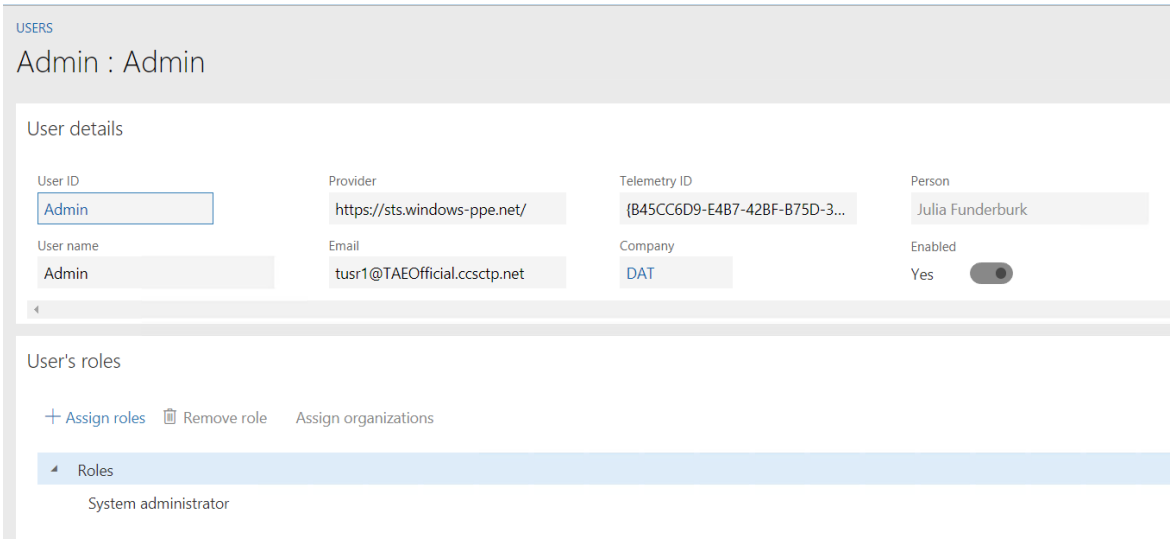
Error example

```
System.TypeInitializationException: The type initializer for
'MS.Dynamics.TestTools.CloudCommonTestUtilities.Authentication.UserManagement' threw an exception. --->
System.InvalidOperationException: Could not enumerate AX users --->
System.ServiceModel.FaultException'1[System.ComponentModel.Win32Exception]: Forbidden
```

Solution

Two scenarios can cause this error:

- The user who is specified as **SelfMintingAdminUser** in the CloudEnvironment.Config file must have the System Administrator role. This issue occurs when the System Administrator role isn't assigned to the user who is specified as **SelfMintingAdminUser**. To verify that you've specified the correct user, you can sign in to the endpoint and view the user's roles.



- An incorrect **NetworkDomain** value was specified for the user who is specified as **SelfMintingAdminUser** in the CloudEnvironment.Config file. You can find the correct value by running the following SQL query against the **AXDB** database of your on-premises deployment.

```
select NETWORKDOMAIN, NETWORKALIAS from USERINFO where NETWORKALIAS='[AdminUserEmail]'
```

At least one security token in the message could not be validated

This issue can occur when you run multiuser tests, or when you create users by using MS.Dynamics.Performance.CreateUsers.exe. It tends to occur when the AOS machine differs from the developer machine.

Error example

```
System.TypeInitializationException: The type initializer for
'MS.Dynamics.TestTools.CloudCommonTestUtilities.Authentication.UserManagement' threw an exception. --->
System.ServiceModel.Security.MessageSecurityException: An unsecured or incorrectly secured fault was
received from the other party. See the inner FaultException for the fault code and detail. --->
System.ServiceModel.FaultException: At least one security token in the message could not be validated.
```

Solution

This issue occurs when the AOS endpoint can't validate the thumbprint of the certificate that you created. There are three possible causes:

- In the CloudEnvironment.Config file, either a value isn't specified for the **IsAdfs** key, or the value is set to **False**. Make sure that the value for the **IsAdfs** key is set to **True**.
- The certificate wasn't installed on the AOS machine. To fix the issue, copy the .cer file that you created earlier

in this topic to the AOS machine, and install the certificate.

- The thumbprint of the certificate wasn't added to the wif.config file on the AOS machine. To fix the issue, see step 8 in the Run a single-user test by using the Performance SDK section for information about how to add the certificate to the wif.config file. After you modify the wif.config file, be sure to restart the application through Service Fabric Explorer.

MS.Dynamics.Test.Team.Foundation.WebClient.InteractionService.dll.config is missing from the deployment items

This issue usually occurs only when you run load tests.

Error example

```
<Test class name>.TestSetup threw exception. System.InvalidOperationException:
System.InvalidOperationException: Could not find endpoint element with name 'ClientCommunicationManager' and
contract 'Microsoft.Dynamics.Client.InteractionService.Communication.Reliable.IReliableCommunicationManager'
in the ServiceModel client configuration section. This might be because no configuration file was found for
your application, or because no endpoint element matching this name could be found in the client element..
at System.ServiceModel.Description.ConfigLoader.LoadChannelBehaviors(ServiceEndpoint serviceEndpoint, String
configurationName)
```

Solution

This issue occurs when the system can't find the MS.Dynamics.Test.Team.Foundation.WebClient.InteractionService.dll.config file when the load tests are run, because the file wasn't added as a deployment item. Verify that the MS.Dynamics.Test.Team.Foundation.WebClient.InteractionService.dll.config file is in the Out folder for the test run:

<solution path>\TestResults\<your test run>\Out

If the file is missing, add it to the deployment items in the test settings.

> **IMPORTANT**
> There are two files that have very similar names. The name of one file ends in ***.dll**, and the name of the other file ends in ***.dll.config**. The ***.dll.config** file must be in the deployment items in the test settings.

CloudEnvironment.Config is missing from the deployment items

This issue usually occurs only when you run load tests.

Error example

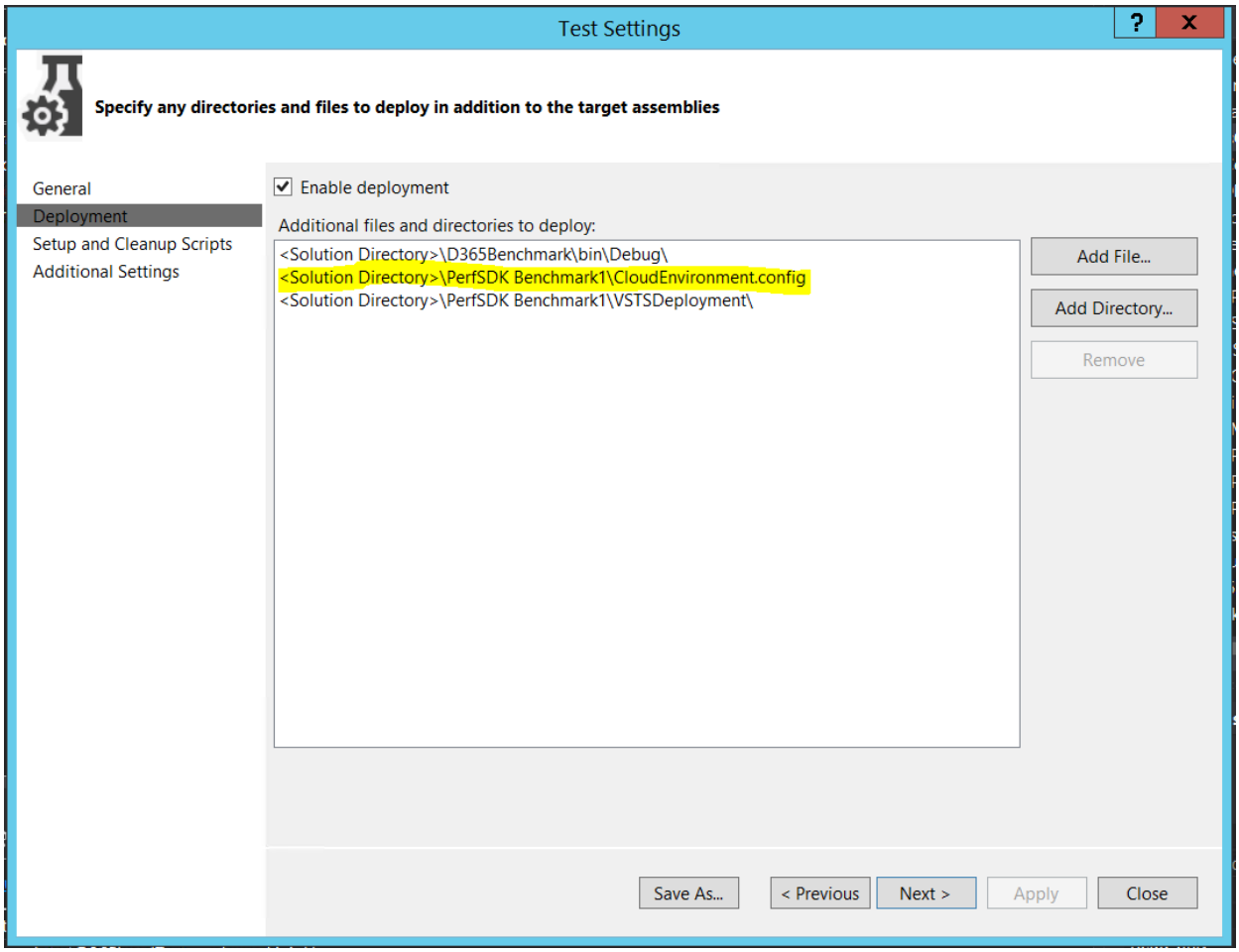
```
Initialization method <Test class name>.TestSetup threw exception.
System.TypeInitializationException: System.TypeInitializationException: The type initializer for
'MS.Dynamics.TestTools.CloudCommonTestUtilities.Authentication.UserManagement' threw an exception. --->
MS.Dynamics.TestTools.TestLogging.EvaluateException: Assert.Fail failed. DateTime="10/13/2017 14:42:55" "The
type initializer for 'MS.Dynamics.TestTools.CloudCommonTestUtilities.Authentication.SecretSettingsHelper'
threw an exception.".
```

Solution

This issue occurs when the CloudEnvironment.Config file isn't present when the tests are run. The issue typically occurs when you run load tests and the CloudEnvironment.Config file wasn't added as a deployment item. Verify that the CloudEnvironment.Config file is in the Out folder for the test run:

<solution path>\TestResults\<your test run>\Out

If the file is missing, add it to the deployment items in the test settings.



InteractiveClientId wasn't specified in the settings

Error example

```
The type initializer for
'MS.Dynamics.TestTools.CloudCommonTestUtilities.Authentication.SecretSettingsHelper' threw an exception. ---
>
Microsoft.CE.VaultSDK.SecretProviderException: InteractiveClientId was not specified in settings
```

Solution

This issue occurs when no value is specified for the **SelfSigningCertificateThumbprint** key in the CloudEnvironment.Config file. In the CloudEnvironment.Config file, find the following line, and paste in the thumbprint of the certificate that you created and installed.

```
<ExecutionConfigurations Key="SelfSigningCertificateThumbprint" Value="" />
```

The remote host forcibly closed an existing connection

Error example

```
System.TypeInitializationException: System.TypeInitializationException: The type initializer for
'MS.Dynamics.TestTools.CloudCommonTestUtilities.Authentication.UserManagement' threw an exception. --->
System.ServiceModel.CommunicationException: An error occurred while making the HTTP request to
<Host name>/Services/AxUserManagement/Service.svc/ws2007FedHttp. This could be due to the fact
that the server certificate is not configured properly with HTTP.SYS in the HTTPS case. This could also be
caused
by a mismatch of the security binding between the client and the server.** ---> System.Net.WebException:
The underlying connection was closed: An unexpected error occurred on a send. ---> System.IO.IOException:
Unable to read data from the transport connection: An existing connection was forcibly closed by the remote
host. --->
System.Net.Sockets.SocketException: An existing connection was forcibly closed by the remote host.
```

Solution

Run the following Windows PowerShell script on the development machine.

```
Set-ItemProperty HKLM:\SOFTWARE\Microsoft\.NETFramework\v4.0.30319 -Name SchUseStrongCrypto -Value 1 -Type
dword -Force -Confirm:$false
if ((Test-Path HKLM:\SOFTWARE\Wow6432Node\Microsoft\.NETFramework\v4.0.30319))
{
    Set-ItemProperty HKLM:\SOFTWARE\Wow6432Node\Microsoft\.NETFramework\v4.0.30319 -Name SchUseStrongCrypto
-Value 1 -Type dword -Force -Confirm:$false
}
```

The w3svc service wasn't found on the computer

This error only occurs when you run load tests by using Microsoft Visual Studio Online.

Error example

```
Test method MS.Dynamics.Performance.Application.GFM.PDLTrend.ProcureToPayTrend.ProcureToPaymentTrend threw
exception:
System.TypeInitializationException: The type initializer for
'MS.Dynamics.TestTools.CloudCommonTestUtilities.Authentication.UserManagement' threw an exception. --->
System.InvalidOperationException: Service w3svc was not found on computer '.'. --->
System.ComponentModel.Win32Exception: The specified service does not exist as an installed service
```

Solution

A hotfix is available that resolves this issue. The Microsoft Knowledge Base (KB) number is 4095640.

Diagnose issues and analyze performance by using Trace parser

2/18/2021 • 2 minutes to read • Edit Online

This topic explains how you can use the Trace parser to consume traces and analyze performance in your deployment. You can use the Trace Parser to find and diagnose various types of errors. You can also use the tool to visualize execution of X++ methods, as well as the execution call tree.

> **NOTE**
>
> There are many more features in the Trace parser are similar to Microsoft Dynamics AX 2012. See the Dynamics Ax Performance Team Blog for more information.

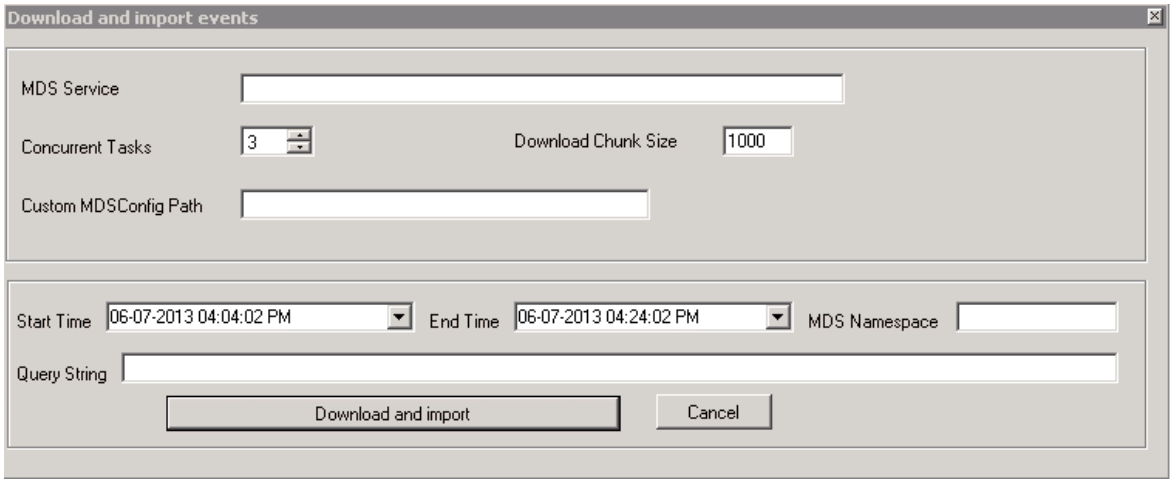
Finding the Trace parser

Trace parser should be preinstalled with your developer deployment or VHD. The install location is here: **C:\Program Files (x86)\Microsoft Dynamics Trace Parser**. If it's not installed, you can run the installer from **C:\PerfSDK\PerfTools\traceparser.msi**.

Capturing events

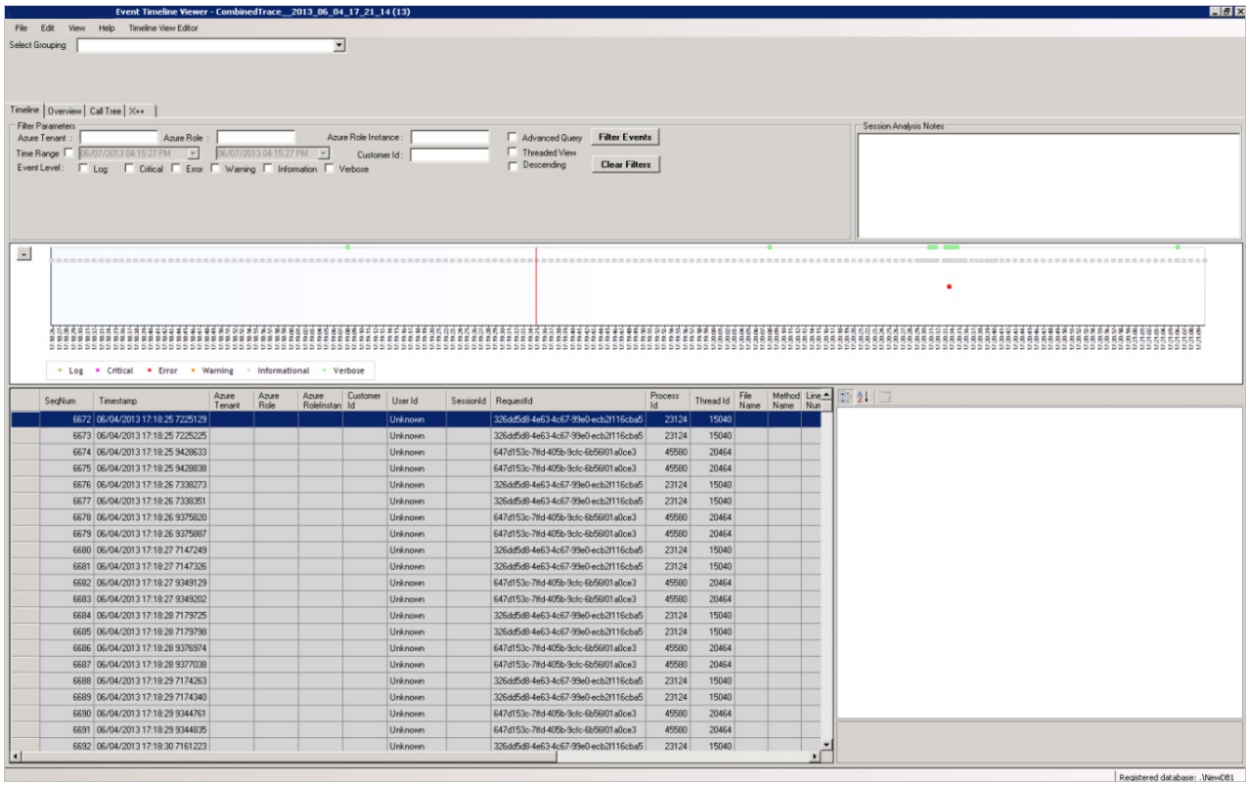
There are two ways that you can obtain the data that you will analyze in the Trace parser. They include:

- Capture events from the local installation.
    - If the **Select Trace** window isn't already open, go to the **File** menu and click **Open trace**. In the **Select Trace** window, click **Capture Events**. After selecting your providers, click **Start**. The Trace Parser tool will start listening to all the providers and capturing the events. Capturing stops when you click **Stop and Import**.
- Open an existing ETL (Windows Event) file that was captured using tools such as Logman.



Viewing traces

Timeline view The Timeline tab is the first tab that you see after you import a trace into the Trace Parser. This tab is shown in the following illustration.



The **Timeline** tab has the following major components:

- The **Select Grouping** drop-down allows you to group based on a variety of categories, such as Customer ID, Username, Session Name, etc. Groupings will display maximum and minimum timestamp of events, total number of events, and lowest event level within the grouping.
- List of all events in a threaded or unthreaded view.
- Property grid displayed for the selected event.
- Timeline chart for all the selected events.
- Filtering of events.
- Session analysis notes.

Call tree view By selecting the **Call Tree** tab, you can see the call tree for all X++ methods. The tab is shown below.

Example of information shown in the Call Tree tab](./media/3_desktop.png)

Similarly, you can display the **X++** tab to view a list of all the X++ methods. They will be sorted by fields such as Inclusive/Exclusive durations, RPC, or Database calls. Note that these are similar to the corresponding tabs in Trace Parser and have the same behavior.

Additional resources

Develop and customize home page

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

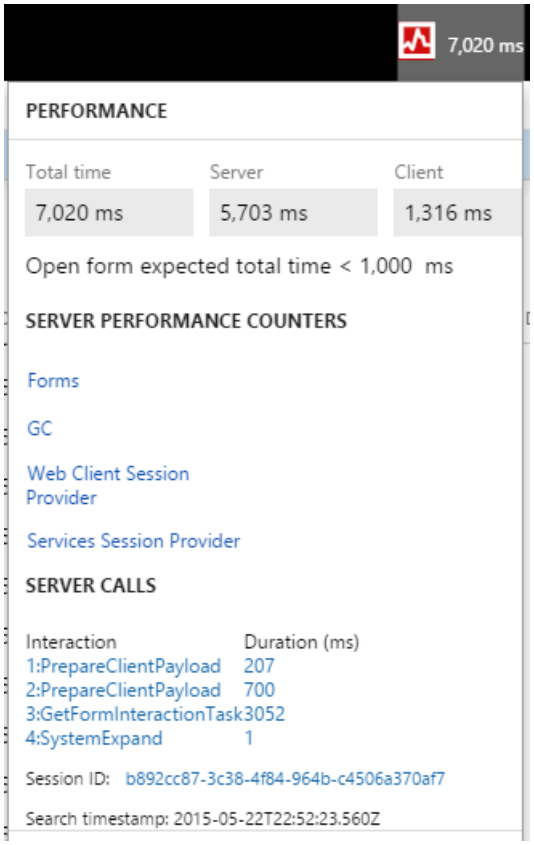
Performance timer

2/18/2021 • 2 minutes to read • Edit Online

This topic provides an overview of the Performance timer, which is a tool that helps you to determine why your system's performance might be slow.

To open the Performance timer, open your webpage with the added parameter debug=develop: https://*yoursite*.cloud.test.dynamics.com/en/?cmp=USMF&debug=develop Note: When you run in debug mode you will notice slower performance. You can quickly get an overview of most performance issues by pressing F12 and working with the debugging tools that are available in your browser. The timer will show up here.

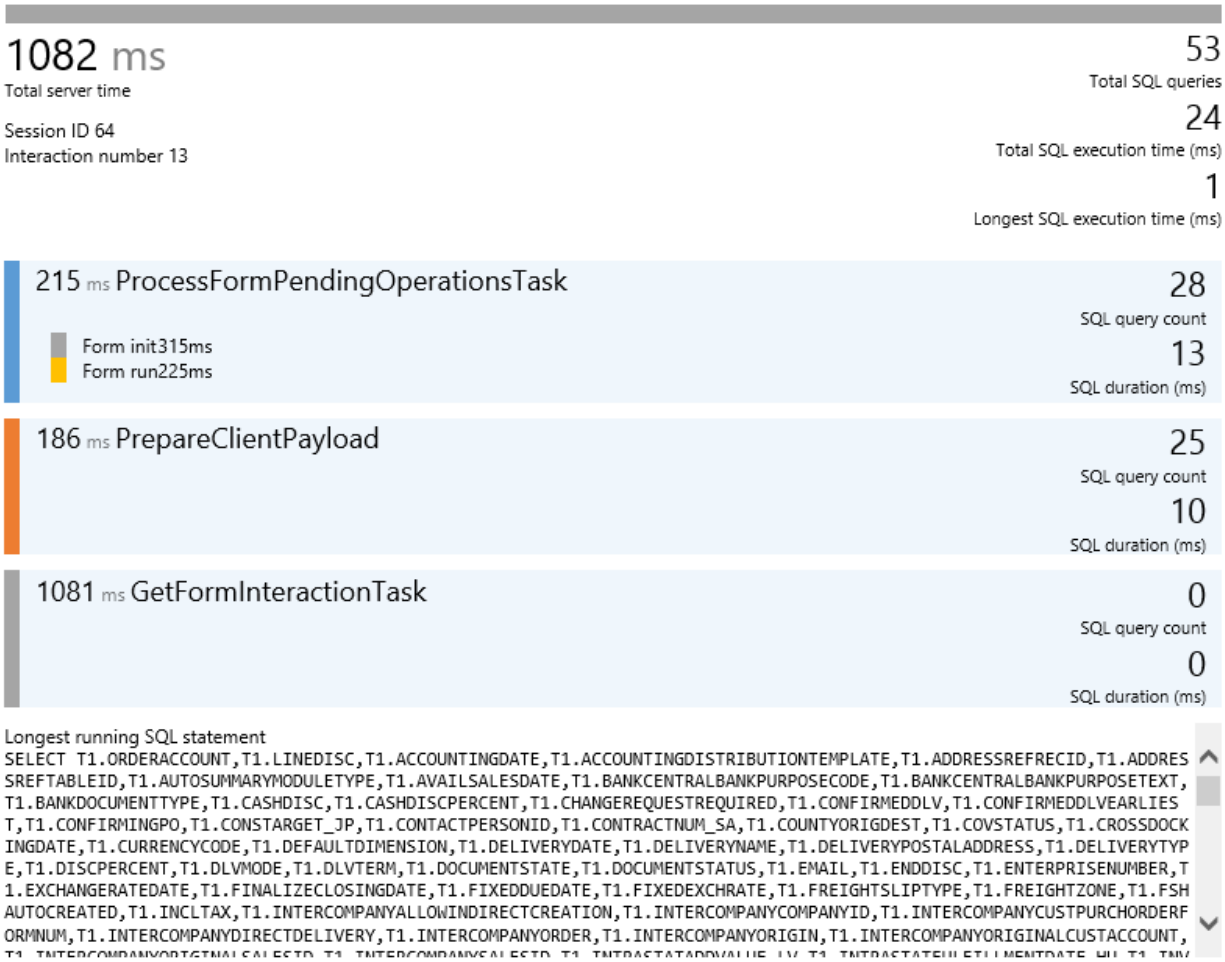
To open a list page, for example, such as the purchase order list page, click the Performance timer. The following screenshot shows the separation between client time and server time, and the total time. Additionally, you can see a set of performance counters and expensive server calls.



For more information about the server performance counters, click on any of the links.

- **Forms** - Forms will show how many forms are currently open, plus the rate at which they opened and closed (per second), and a set of counters, such as the total amount of created or closed forms.
- **GC** - This is information about the garbage collection processes on the server.
- **Web client session** - This shows how many web client sessions you currently have and how many are in use.
- **Services Session provider** - This is the total number of sessions created.

For more information, click a link. In the next screen, you can see how many SQL queries were triggered by this individual call and which SQL query was the most expensive.



This information can help you to understand what to trace and where to start troubleshooting.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Testing and validations

2/18/2021 • 6 minutes to read • Edit Online

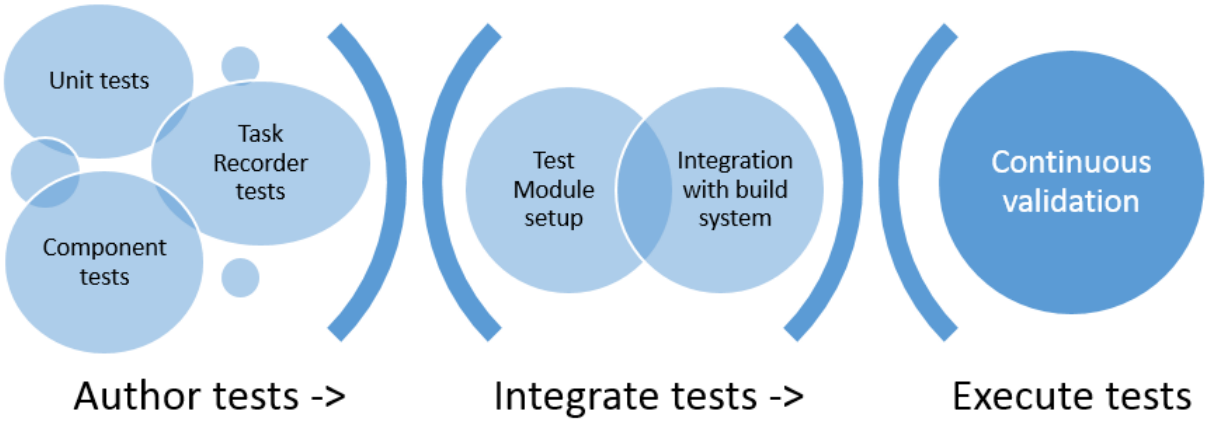
This tutorial shows you how to create and run test cases.

Prerequisites

You will need to deploy Developer Topology with Developer and Build VM.

Key concepts

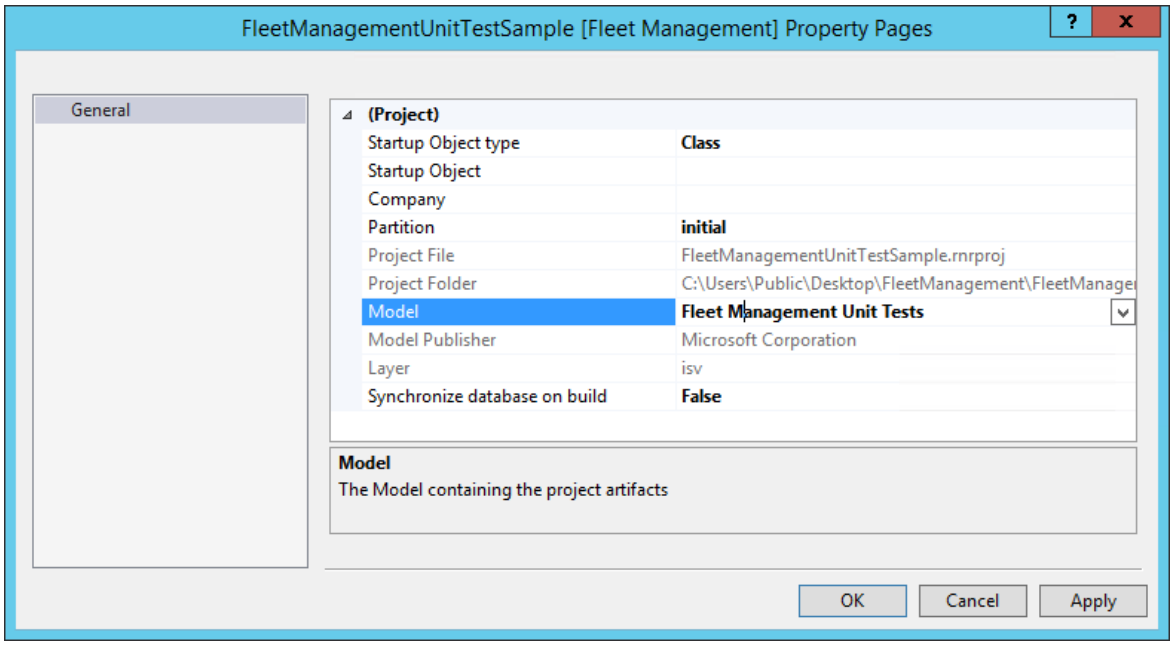
- Use SysTest Framework to author unit/component test code.
- Test isolation
- Test module creation to manage test code and FormAdaptors.
- Import Task Recorder recordings into Visual Studio to generate test code.
- Integrate a Test module with a build machine.



Use SysTest Framework to author unit/component test code

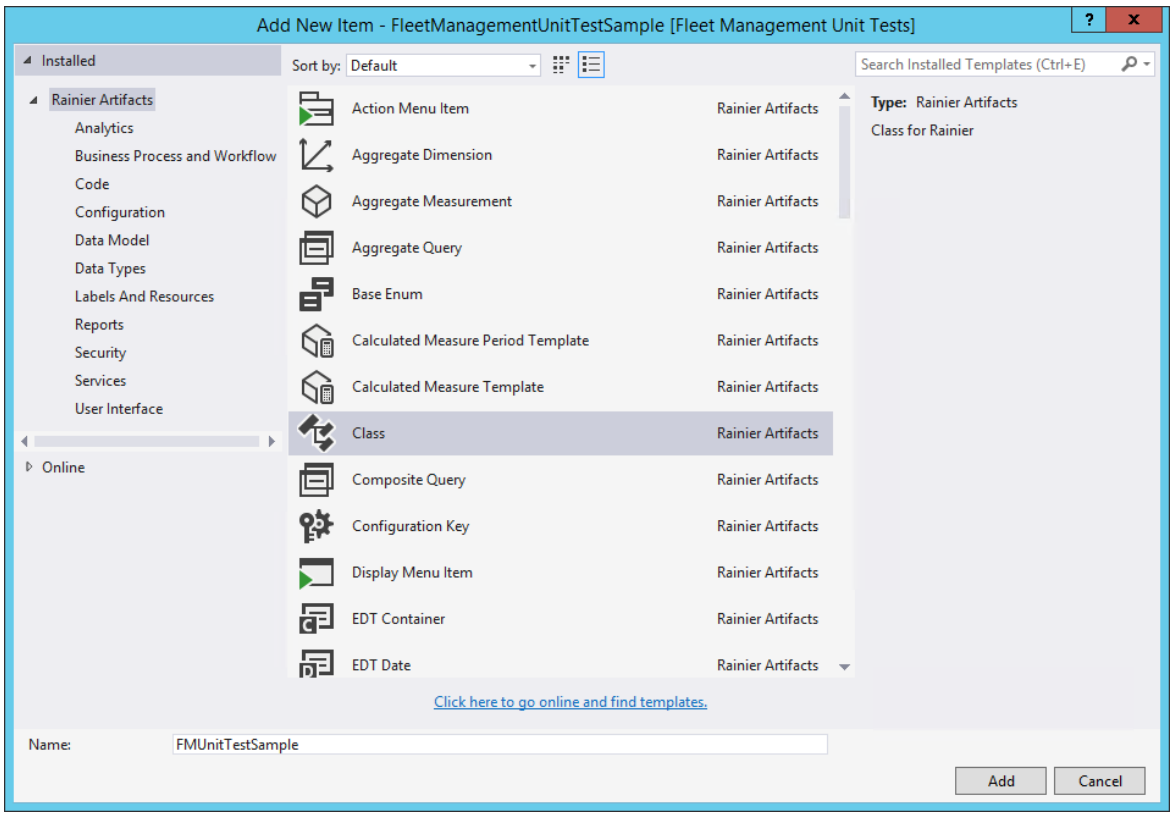
You can create new test cases to test the functionality in an application.

1. Open Visual Studio as an administrator.
2. On the **File** menu, click **Open** > **Project/Solution**, and then select **FleetManagement solution** from the desktop folder. If the solution file is not on your computer, the steps to create it are listed in Tutorial: Create a Fleet Management solution file out of the Fleet Management models in the AOT.
3. In **Solution Explorer**, right-click the **Fleet Management** solution, point to **Add**, and then click **New Project**.
4. Choose **Finance and Operations** as the project type to create.
5. Name this new project *FleetManagementUnitTestSample*, specify the FleetManagement folder on the desktop (C:\Users\Public\Desktop\FleetManagement) as the location, and then click **OK**.
6. In **Solution Explorer**, right-click the new project, and then click **Properties**.
7. Set the **Model** property to **FleetManagementUnitTests**, and then click **OK**.



8. Right-click the FleetManagementUnitTestSample project, point to **Add**, and then click **New Item**.

9. In the **Add New Item** window, select **Class** as the type of element to add. Name the new class FMUnitTestSample, and then click **Add**.



10. In the first line of the code for the new class, indicate that the class extends the SysTestCase class.

11. Add the following code to define the methods for the class. These methods define two additional tests.

```
class FMUnitTestSample extends SysTestCase
{
    public void setup()
    {
        // Reset the test data to be sure things are clean
        FMDataHelper::main(null);
    }

    [SysTestMethodAttribute]
    public void testFMTotalsEngine()
    {
        FMRental rental;
        FMTotalsEngine fmTotals;
        FMRentalTotal fmRentalTotal;
        FMRentalCharge rentalCharge;
        FMRentalTotal expectedtotal;
        str rentalID = '000022';

        // Find a known rental
        rental = FMRental::find(rentalID);

        // Get the rental charges associated with the rental
        // Data is seeded randomly, so this will change for each run
        select sum(ExtendedAmount) from rentalCharge
                where rentalCharge.RentalId == rental.RentalId;

        fmTotals = FMTotalsEngine::construct();
        fmTotals.calculateRentalVehicleRate(rental);

        // Get the totals from the engine
        fmRentalTotal = fmTotals.totals(rental);

        // Set the expected amount
        expectedTotal = rental.VehicleRateTotal + rentalCharge.ExtendedAmount;

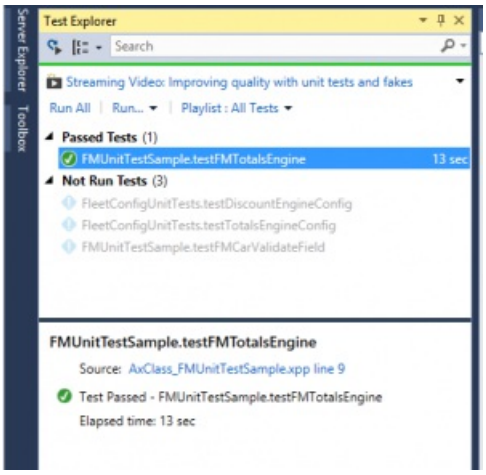
        this.assertEquals(expectedTotal,fmRentalTotal);
    }

    [SysTestMethodAttribute]
    public void testFMCarValidateField()
    {
        FMCarClass fmCar;

        fmCar.NumberOfDoors = -1;
        this.assertFalse(fmCar.validateField(Fieldnum("FMCarClass", "NumberOfDoors")));

        fmCar.NumberOfDoors = 4;
        this.assertTrue(fmCar.validateField(Fieldnum("FMCarClass", "NumberOfDoors")));
    }
}
```

12. Save the new class. After the save is complete, you will see the additional two test cases in **Test Explorer**. Right-click on the FleetManagementUnitTestSample project in **Solution Explorer**, and then click **Build.**
13. On the **View** menu, open **Test Explorer**.
14. Click **Run selected test** to execute specific test case.
15. Test Explorer will show the results of test after it is complete.



Test isolation

For a test to be of high value it must be reliable. A test will pass or fail consistently, independent of other factors such as other tests. One typical cause of unreliable tests is leaking state, such as data left behind in the data base that influences downstream tests. To prevent this type of issue, you can use the `SysTestTransaction` attribute.

TESTTRANSACTIONMODE	DESCRIPTION
AutoRollback	**Default**. This provides the best isolation. All transactions are rolled back using SQL save points, and all database statements are routed to the main connection, including user connections. No data will be persisted.
LegacyRollback	All insert statements are tracked and deleted during clean-up. All insert statements are downgraded to row-by-row. One typical use case is when testing user connections or concurrency scenarios. This isolation level will clean up setup data, and the recommendation is to wrap each test method in a ttsBegin and ttsAbort.
LegacyRollbackWithUpdateTracking	All update, delete, and insert statements are tracked and reverted during cleanup. All insert, update, and delete statements are tracked and downgraded to row-by-row. This is the slowest isolation level.
None	**Only use for debugging**. This provides no isolation. This setting can be useful to temporarily to debug a test, as it allows you to use the regular user interface to navigate the data that the test created.

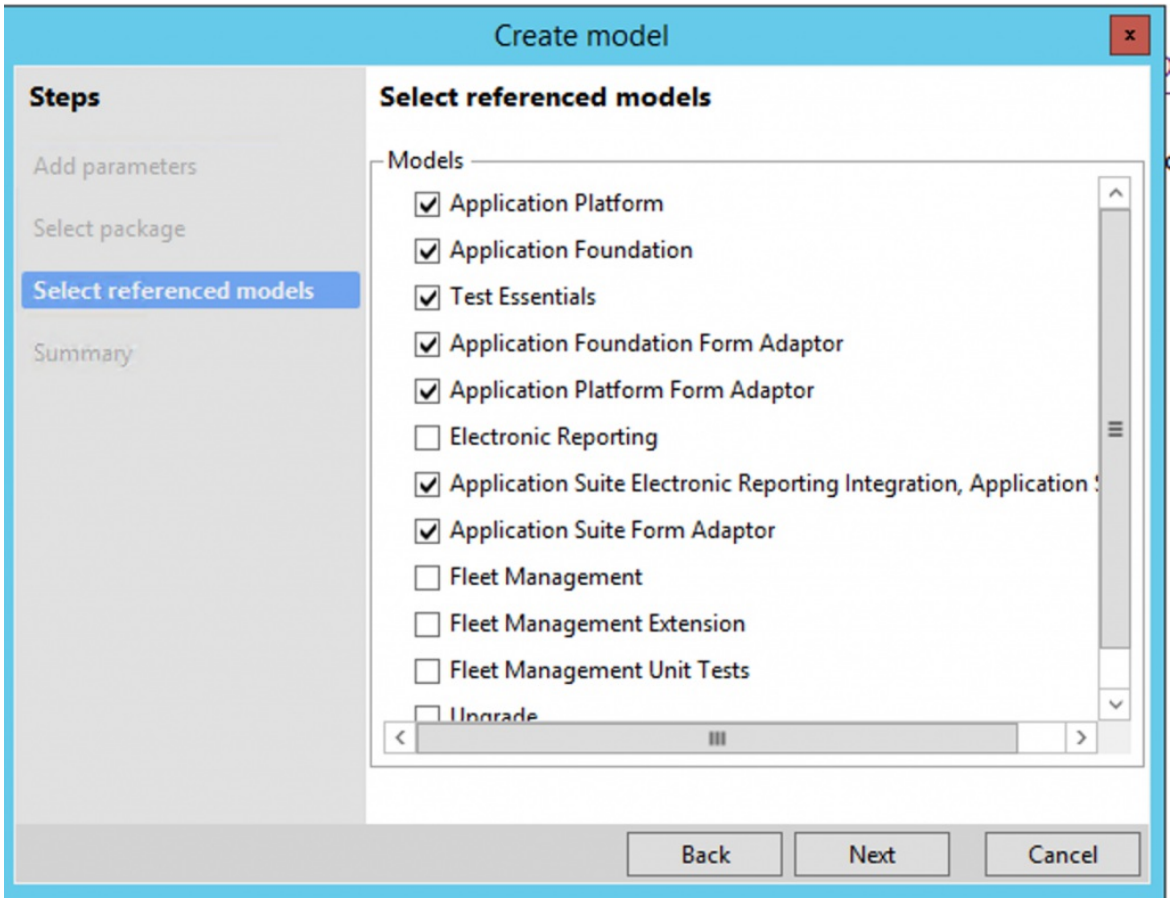
Example:

```
[SysTestTransaction(TestTransactionMode::LegacyRollback)]
class MyTestSample extends SysTestCase
```

Test module creation to manage test code and FormAdaptors

Creating a test specific module helps to keep test code together and manageable.

1. Open **Visual Studio** and go to **Finance and Operations** > **Model Management** > **Create model**.
2. Enter the model name, select the layer, and then enter any additional details. Note that it's a good idea to include the word **Test** in the name of the test module. The default build definition is configured to discover all test modules that contain the word **Test**.
3. Because this model holds forms from the Application Platform/Foundation, add references to models shown below.

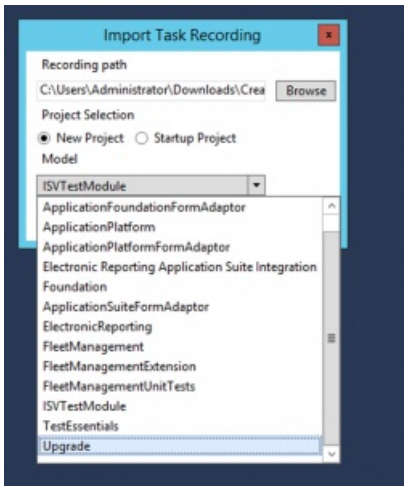


After the base test module is in place, you can import a Task Recorder recording to generate test code. When you import a Task Recorder recording XML, test code is generated using FormAdaptors. Form adaptors are wrapper classes over forms which provide strongly typed API that can be used to test form functionality. We have included pre-generated FormAdapters for each package for built-in forms. In the test module, add a reference to the corresponding Form Adaptor for packages and Test Essentials, which has helper methods to execute test code.

Import a Task Recorder recording into Visual Studio to generate test code

You can generate test code from Task Recorder recording to execute headless (non-UI) test.

1. Record a scenario in by using Task Recorder.
2. To import a Task Recording, in Visual Studio, click **Finance and Operations** > **Addins** > **Import Task Recording**.
3. In the **Import Task Recording** dialog, select the Test Module (ISVTestModule) under which you want to import task recording, and browse to recording xml file.

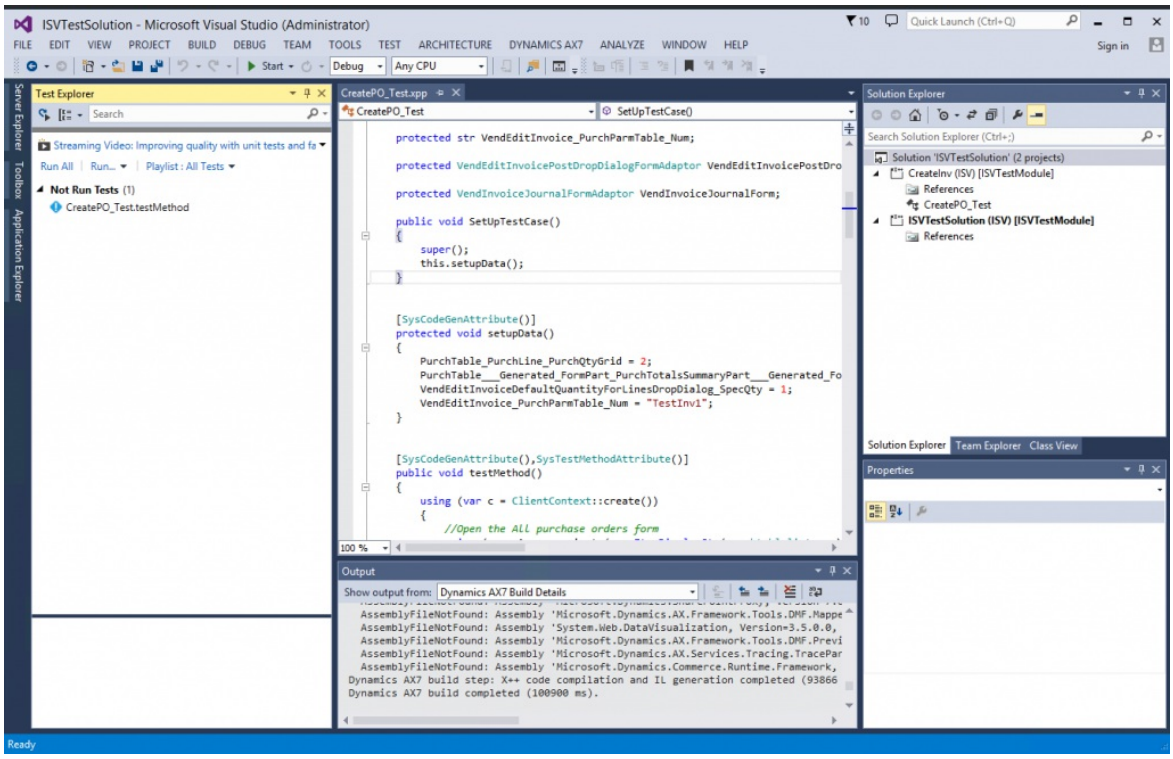


4. The task recording import process generates test code that is based on the SysTestAdapter and FormAdaptor which can be viewed in Visual Studio IDE. We do not expect you to change any test source code that is generated as part of this step.

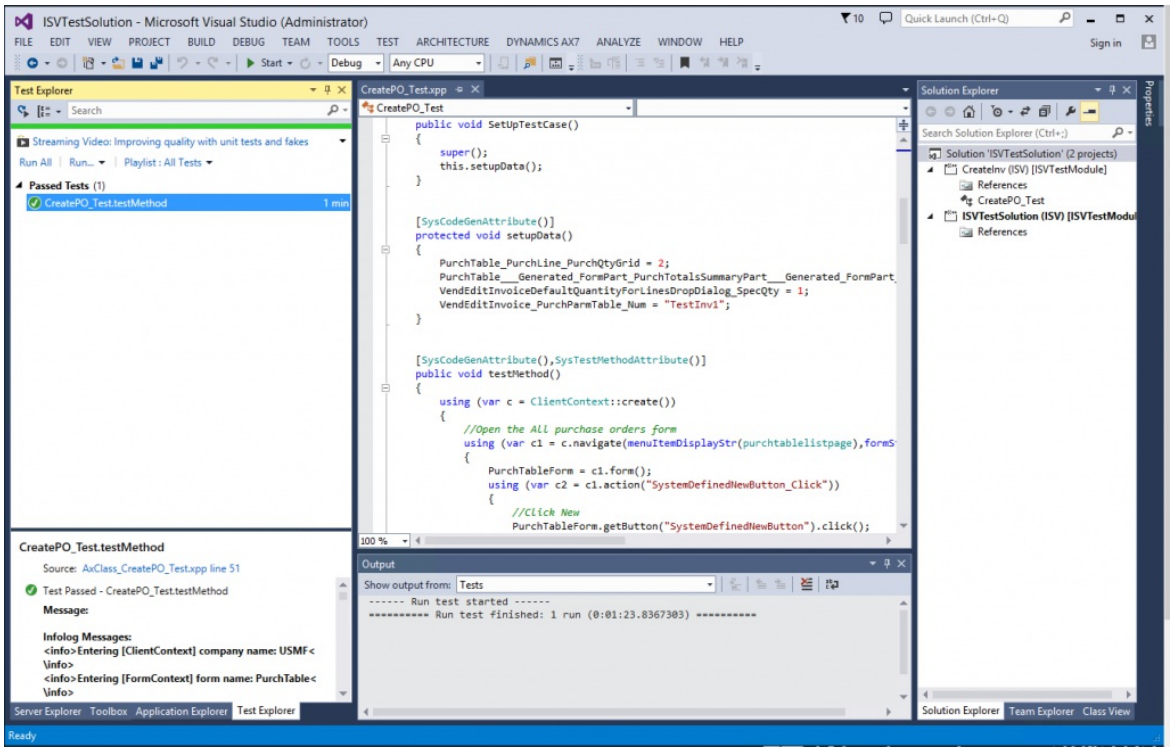
5. After the test code is generated, set up Visual Studio options for test discovery and execution:

   - If you have a 64-bit machine, you can run unit tests and capture code coverage information as a 64-bit process.
   - To configure this, select **Test** > **Test Settings** > **Default Processor Architecture**, and then select **X64**.
   - You might run into a situation in which the test execution engine opens and locks an assembly in your test project. When this happens, you can't for example, save changes to the assembly. To fix this, select **Test** > **Test Settings**, and then select **Keep Test Execution Engine Running**.
   - Now that you have test code generated in Visual Studio IDE, it's time to discover the test and try executing them locally.

6. From menu options, select **Test** > **Windows**, and then click **Test Explorer**. After the Test Explorer window is open, it will try to discover test from test code and list all the available tests as shown below.



7. Select the test and then click **Run** > **Execute selected**. This will execute test against the locally deployed environment.



Integration of the test module with build process

After the test module is a part of source control, the build process template will discover all test modules, which contain the word **Test** in the name. The following illustration shows build and test execution as part of Visual Studio Online.

CTP5[illegible]_03.18.15_18.11.18_20150318.1 - Build succeeded

View Summary | View Log – Open Drop Folder | Diagnostics ▾ | <No Quality Assigned> ▾ | Actions ▾

[illegible] triggered CTP5.[illegible]_03.18.15_18.11.18 (CTP5.[illegible]) for changeset 26

Ran for 23.5 minutes (rdxp007C62rains - Controller), completed 4 seconds ago

Latest Activity

Build last modified by [illegible] 4 seconds ago.

Request Summary

Request 1, requested by [illegible] 23.7 minutes ago, Completed

Summary

Default Configuration and Platform

0 error(s), 0 warning(s)

▷ $/CTP5. ./Metadata/AXModulesBuild.proj compiled

◢ 2 test runs completed - 87% average pass rate (80% total pass rate)

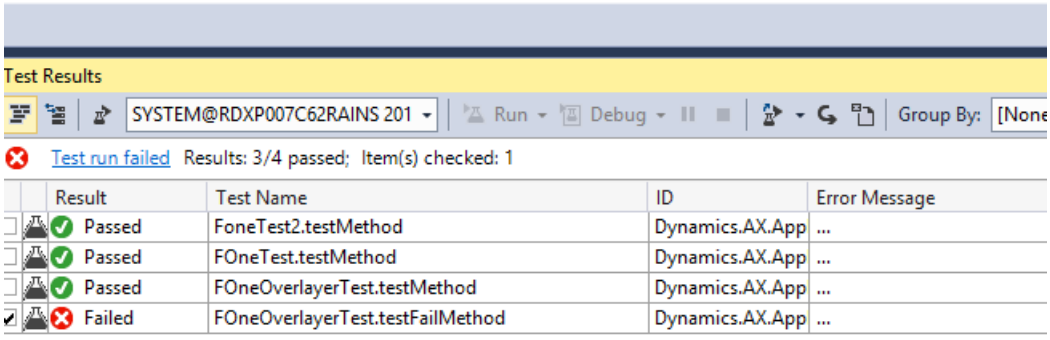
SYSTEM@RDXP007C62RAINS 2015-03-18 18:41:11, 1 of 1 test(s) passed

▷ SYSTEM@RDXP007C62RAINS 2015-03-18 18:41:34, 3 of 4 test(s) passed

No Code Coverage Results

Impacted Tests

No tests were impacted

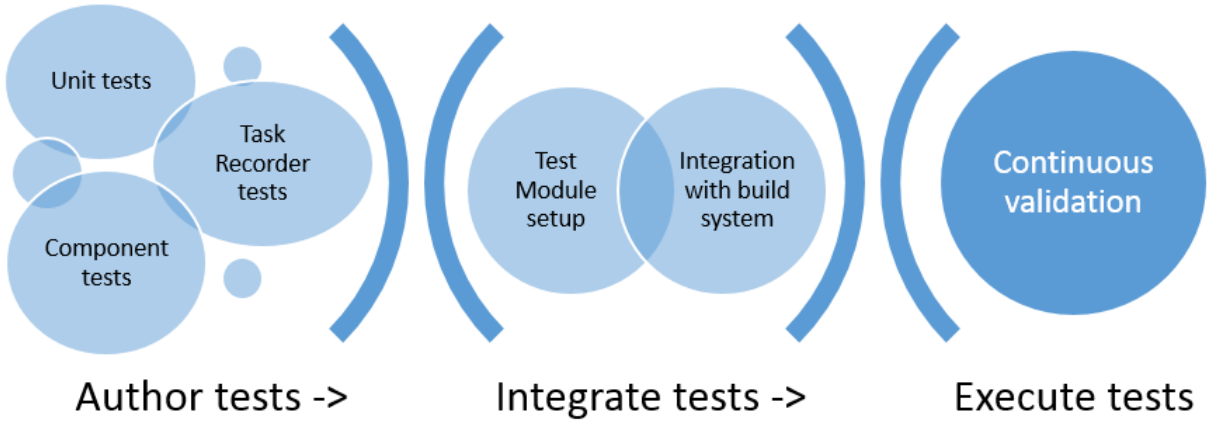


Test projects in Visual Studio

2/18/2021 • 2 minutes to read • Edit Online

This topic describes the options for testing in Visual Studio.

A custom unit test adapter is available in Visual Studio. This adapter lets test authors use the standard **Test Explorer** window in Visual Studio to schedule X++ tests and analyze test results. Developers can author tests by using **SysTestAdaptor**. They can also generate test code from Task Recorder recordings. These test cases can then be added to build systems for validations.

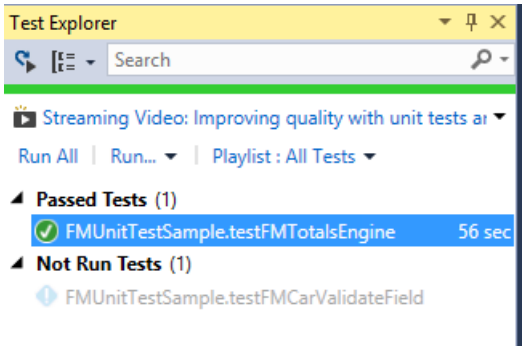


Author unit/component test code by using the SysTest Framework

When you create a project in Visual Studio, you can add an X++ unit test. You extend the class with **SysTestCase**, and then either add the **SysTestMethodAttribute** attribute or prefix the case with "test" in the method name.

```
class FMUnitTestSample extends SysTestCase
{
    [SysTestMethod]
    public void testTotalsEngineConfig()
    {
    }
}
```

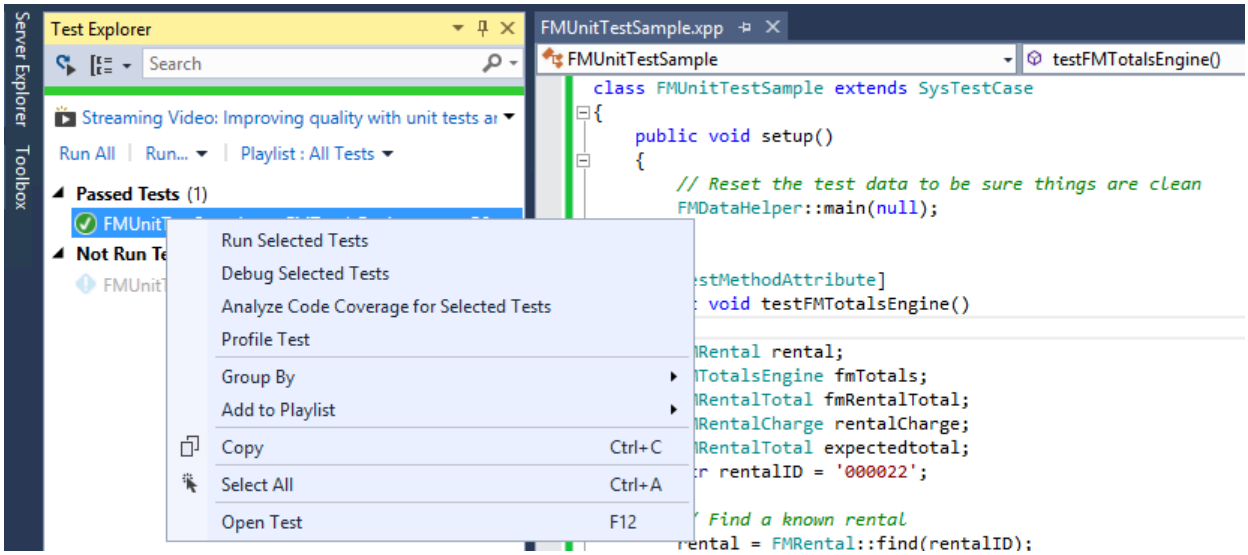
After you save the class, each test appears in Test Explorer, just as a C# test would appear.



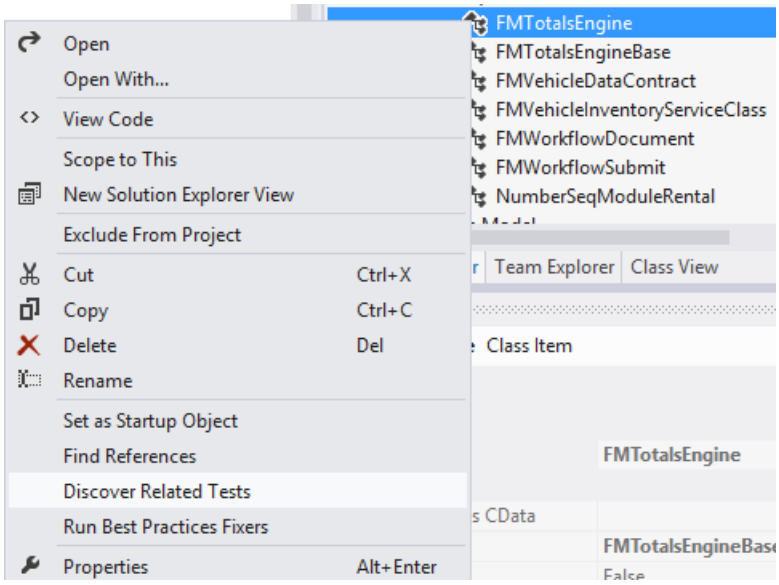
In Test Explorer, you can run the tests, or you can debug the test case by right-clicking and running or debugging the selected tests.

NOTE

Before you can run tests, you must build the project so that it includes tests.



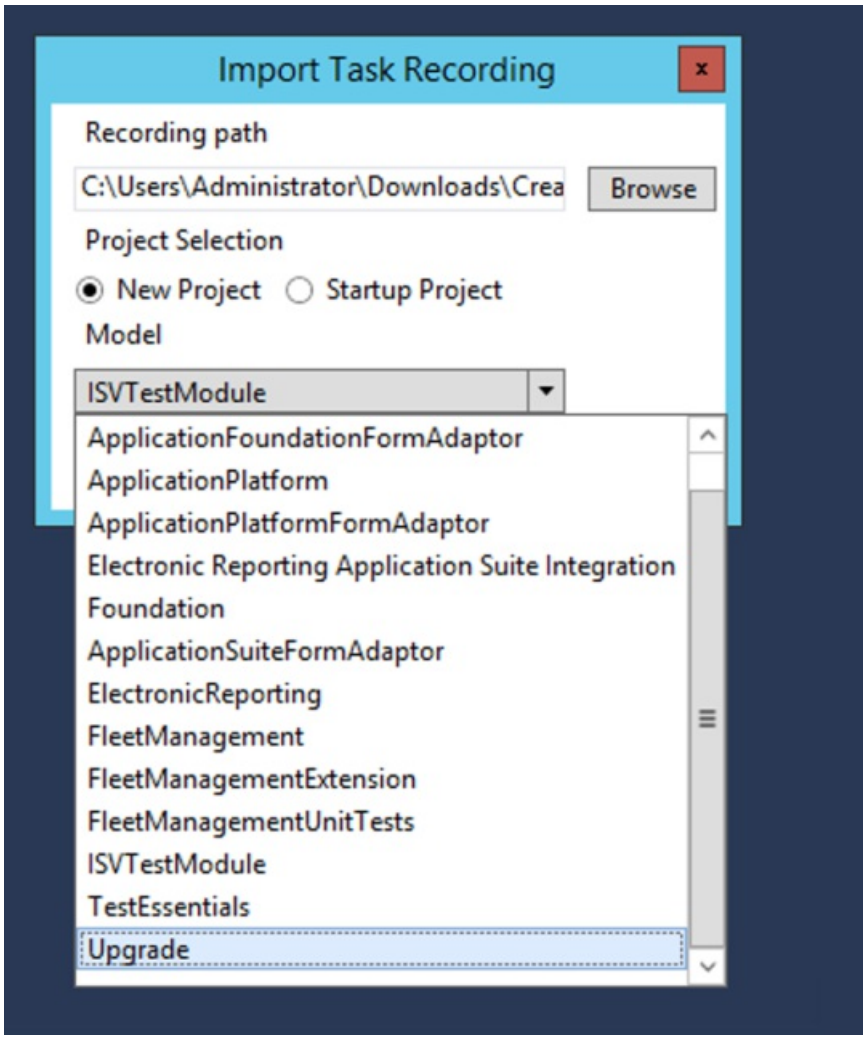
You can also discover existing tests for an object in your project. Discovery uses cross-reference data. Right-click an object in the project, and then select **Discover Related Tests**. This command queries the cross-reference data and returns any tests that reference the object. The list of test cases is displayed in Test Explorer.



By using this functionality, you can run all the relevant tests. Test Explorer contains all tests for the current project and all tests for the referenced objects.

Generate test code by importing Task Recorder recordings into Visual Studio

You can import the XML for Task Recorder recordings to generate test code that can be used to validate various business process scenarios.



Generated code is based on the SysTest Framework and FormAdaptors. FormAdaptors are wrapper classes over pages. They provide strongly typed application programming interfaces (APIs) that can be used to test page functionality. Pre-generated FormAdaptors are included for each package for built-in pages. In a test module, add a reference to a corresponding FormAdaptor for packages and "Test Essentials," which contain helper methods to run test code.

Advanced Options

For advanced options to categorize and filter tests for execution, see SysTest Filtering using class and method attributes.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Deploy and use a continuous build and test automation environment

2/18/2021 • 7 minutes to read • Edit Online

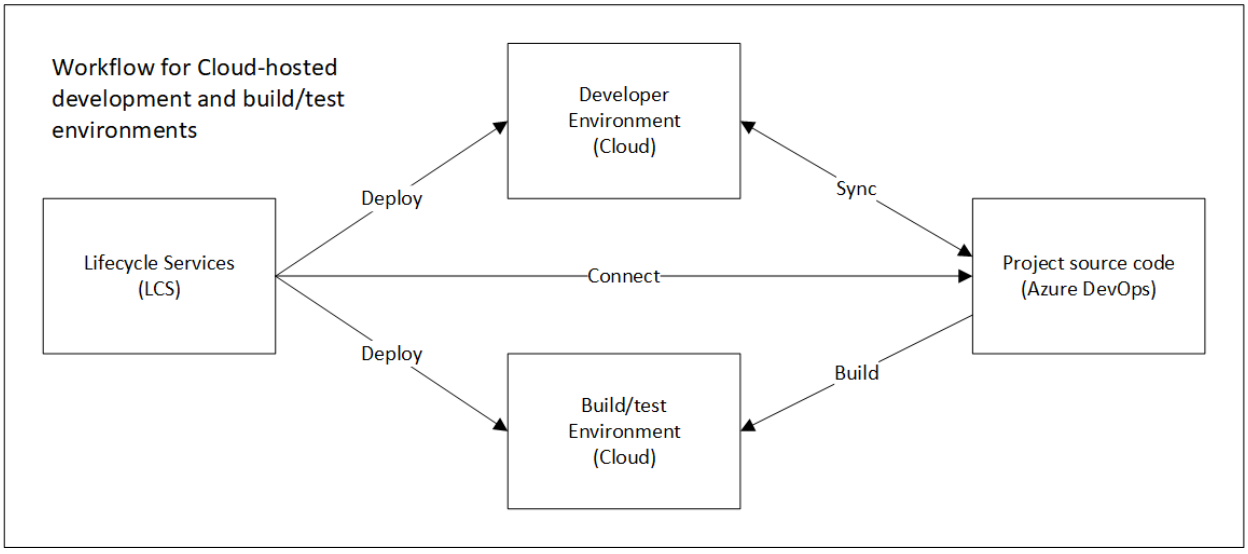
This topic describes how to deploy and use an environment that supports continuous build and test automation.

Prerequisites

Cloud deployment of virtual machines (VMs) requires a Microsoft Azure DevOps subscription.

Workflow

After you configure an Azure DevOps subscription in Microsoft Dynamics Lifecycle Services (LCS), you can use LCS to deploy developer VMs or build/test VMs. LCS configures a developer VM that can be mapped to an Azure DevOps project. LCS also configures a build VM that is automatically mapped to an Azure DevOps project and has a build agent/controller that builds modules from the Azure DevOps project and runs automated tests that have an external endpoint for validation. The following illustration shows a typical workflow.



This workflow includes an LCS deployment of a developer VM and a build/test VM in Azure.

- LCS creates developer and the build/test environments in Azure. To create a build/test environment, LCS must be able to determine where the source code for the Azure DevOps project is.
- The developer works on source code on the developer VM, and the work is synced to the Azure DevOps project.
- The build process synchronizes the code from Azure DevOps onto the build/test VM and produces deployable packages that you can apply to sandbox and production environments. The source code doesn't flow directly from the development VM to the build/test VM. They are synced through Azure DevOps.

For information about how to write custom test code or generate automated test code to integrate with the build infrastructure, see Testing and validations.

Set up Azure DevOps

Choose a plan

The first step is to choose an Azure DevOps plan for your organization.

> **NOTE**
>
> TFVC is the only source control repository that is supported. Git isn't supported.

Set up Azure DevOps

To set up Azure DevOps, follow these steps.

1. Create a personal access token. The token is used for all LCS background actions. These actions include upgrade and deployment. When users initiate actions from LCS, LCS expects that those users will be added to Azure DevOps. The users must authorize LCS access to Azure DevOps on their behalf.
2. Configure LCS.

Until you authorize LCS access to Azure DevOps, you will see a "setup is not complete" message in action center.

Suspend current builds

If you're deploying a build environment on an existing Azure DevOps project that already has a build definition, make sure that you don't have any active triggers to queue the build. Additionally, make sure that no builds are scheduled or queued against the build pool.

Deploy Developer and Build/Test environments from LCS

LCS provides an option to deploy Development and Build/Test environments. With this option, you can deploy developer and build VMs in the cloud that are connected to your Azure DevOps project.

Azure DevOps credential setup and linking to LCS project

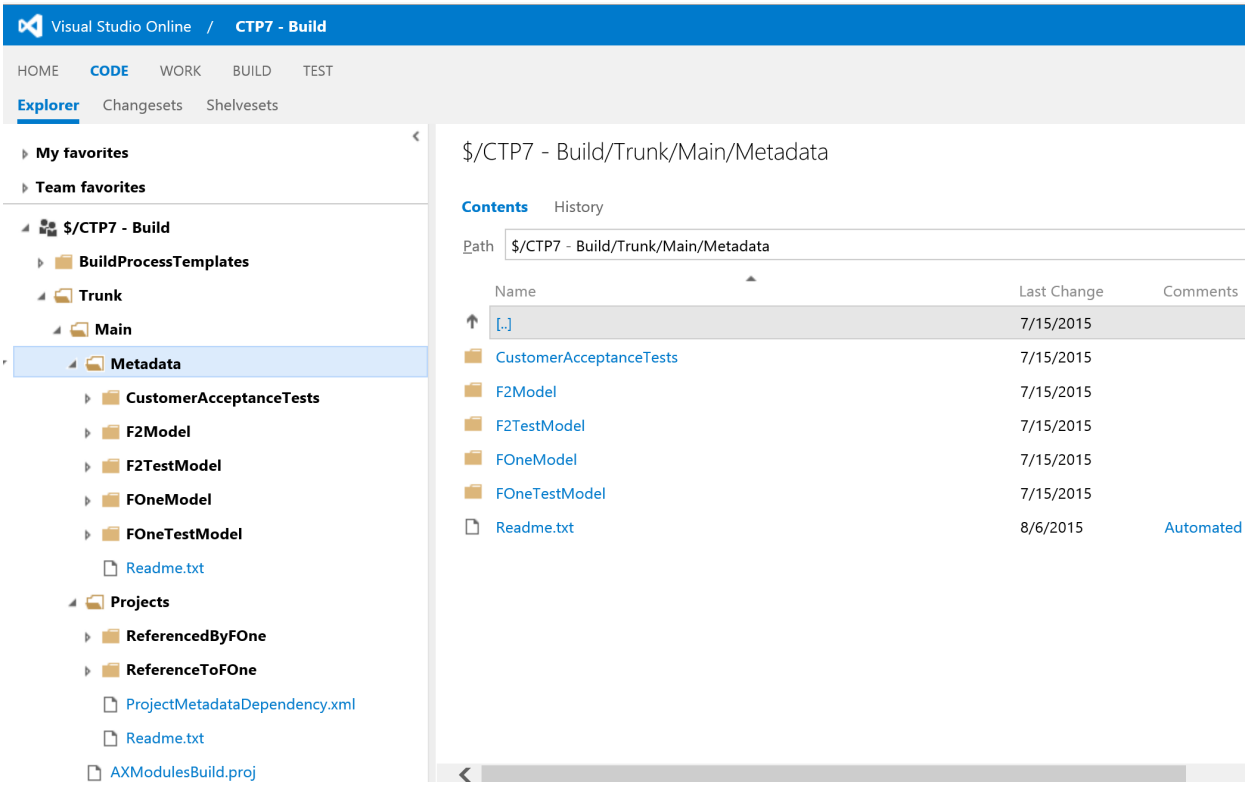
If you have not already done so, you need to first setup your LCS project to connect to your Azure DevOps project before you deploy a build environment.

1. Login to the LCS portal to connect to Azure DevOps and your LCS project at https://lcs.dynamics.com/.
2. Select a project that you are working on.
3. Click the **Project Settings** tile.
4. Select **Azure DevOps** and enter the Azure DevOps URL where the source code for your module project is located.
5. Specify the Azure DevOps link, authorize, and then click **Choose default project**.

   > **NOTE**
   >
   > We currently support VSTF as source control and do not support Git.

Check-in migrated or new module code into Azure DevOps

As part of code Migration process or development activities, we expect you to check-in your model source files and the associated test model source files into Azure DevOps. If you have migrated your code using the LCS migration service, this is automatically done for you. If you have not checked in any code into Azure DevOps and work on direct check-in, you must follow certain guidelines for the Azure DevOps folder structure. This will help with setting up correct build definition. All modules should be added to root folder **Metadata**. Under each module, there should be two folders. One folder contains all models. The other folder should contain descriptor XML for that module.



Deploy a Build environment

The topic Deploy and access development environments describes how to deploy developer environments. Use the same flow to deploy a build environment. As you are going through the deployment or configuration wizard, when prompted to **Select a Topology**, select **DevTest** then select a **Build and Test** topology.

As part of the deployment wizard, you can configure the build agent name and build agent pool.

Click **Advanced settings**, select **Azure DevOps**

1. Build Agent Name: Friendly name for build agent on Azure DevOps
2. Build Agent Pool: specify build agent pool name which should be used for build machine deployment. Make sure Azure DevOps contains at least one agent pool. By default, there will be the default pool. If you have deleted the default pool then build deployment will fail.
3. Branch Name: Specify your Azure DevOps source code branch which will be default source code sync location for the build VM. Default branch is "Main".

Test integration with the build

There are two ways to integrate test as part of build process for testing and validation:

- SysTest framework based unit and component level tests.
- Generate code from Task Recorder recording XML for automated test execution.

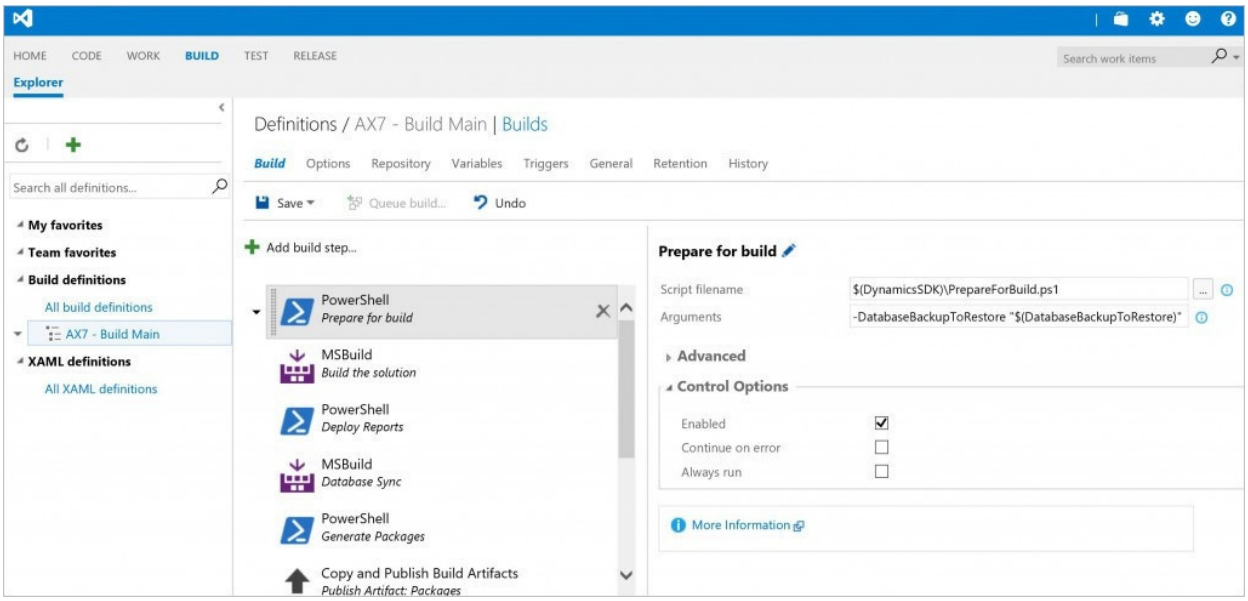
The details of these two approaches are mentioned in the Testing and validation article. Review this article for testing and validation strategy.

Use the Build VM environment

When a Build VM is deployed in Developer topology through LCS, it is pre-configured and ready to start a build. You can change the default configuration at any time from the Visual Studio IDE or the Azure DevOps interface. On a Build VM, the module source code is synchronized to the build machine for easy build setup. The build machine is also auto-configured with default settings for build agent, build controller, build process template, and build definition. Tests that are integrated with build definition are executed after the build is successful.

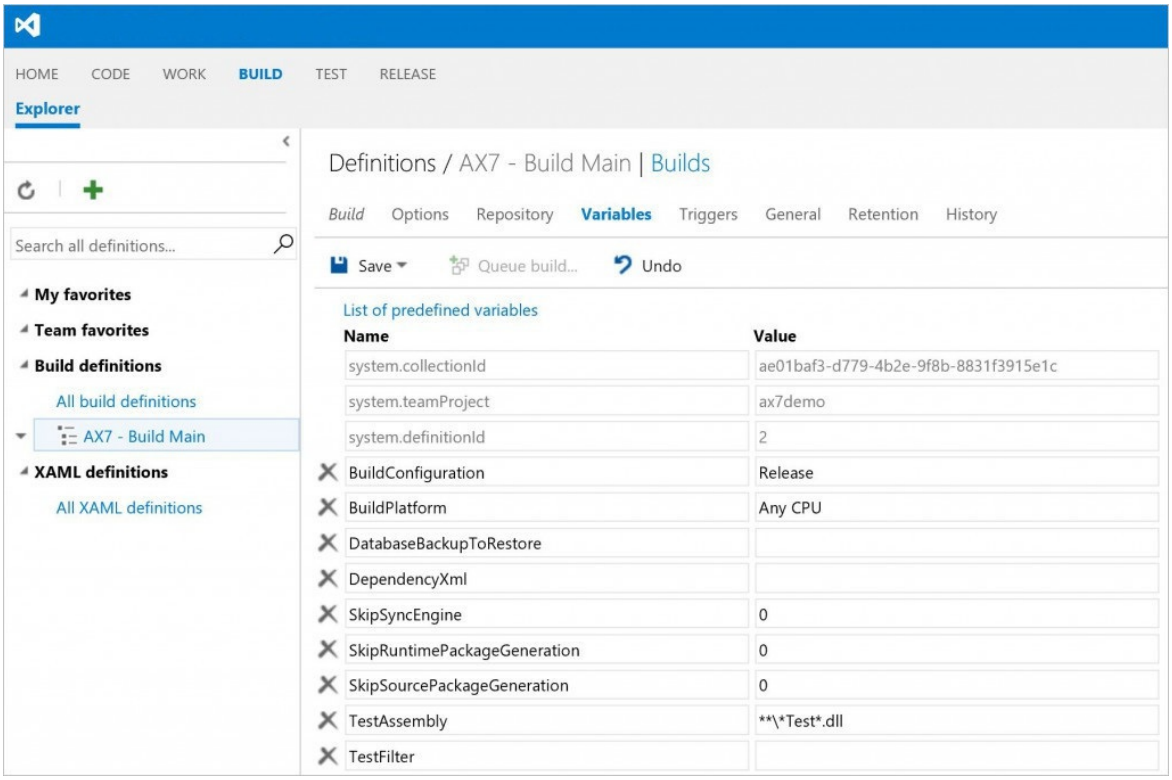
Review a pre-configured customizable build environment

The build VM contains the vNext build agent which was released as part of **Azure DevOps**. When you deploy the Build VM, the build agent is configured by default to connect and sync with the Azure DevOps project. As a part of the Build VM configuration, the default build definition is also created and configured, as shown below.

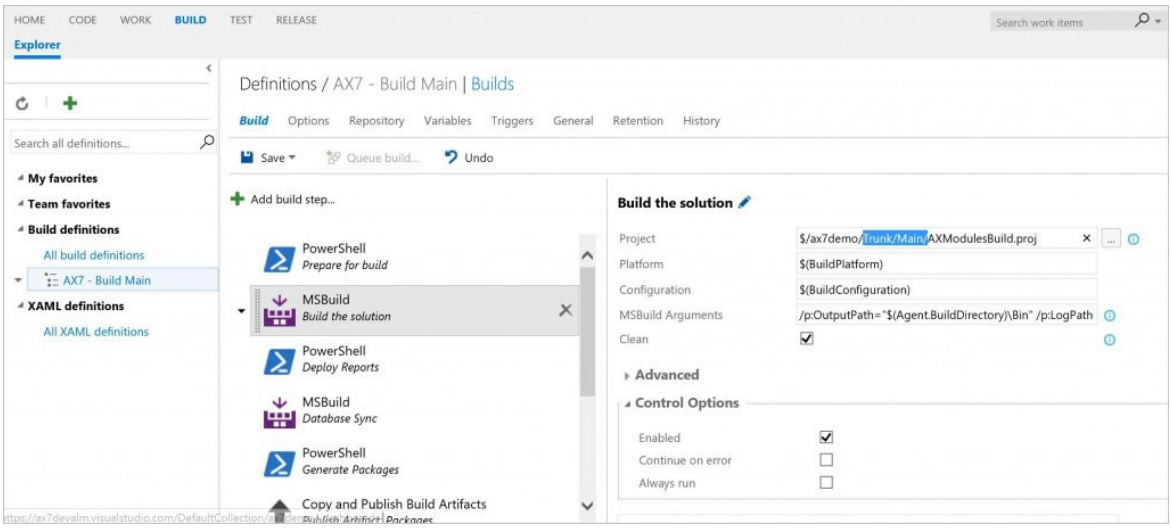


Default build definition contains multiple tasks to perform specific operation, as described below.

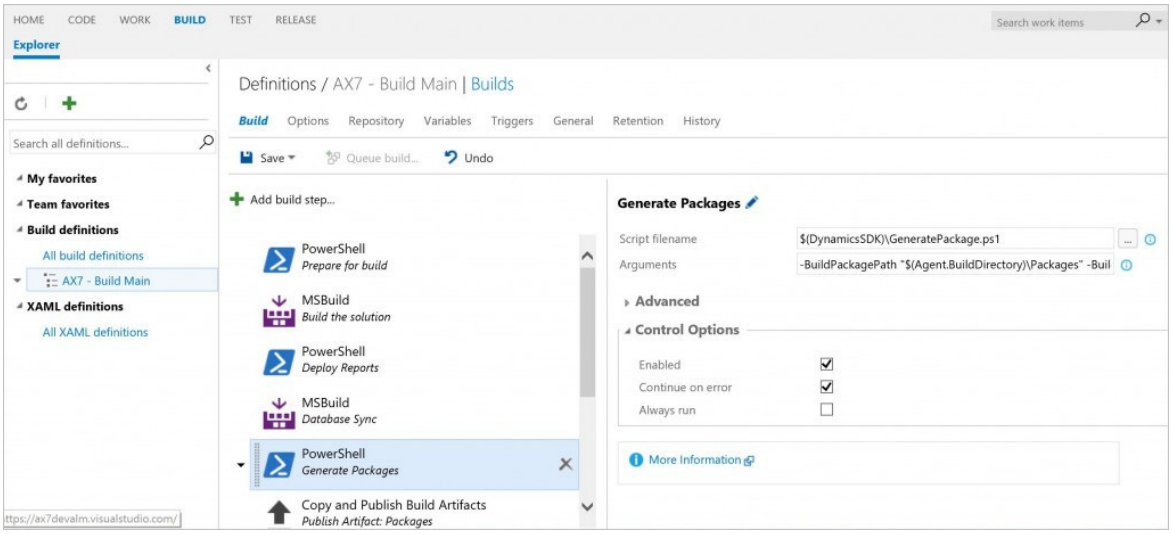
1. Configure the predefined variables parameters that will be passed to the build. To set up a clean database for every build execution, provide the name of the database backup file for the **DatabaseBackupToRestore** variable. The packages folder is restored at every build with a copy of a clean package folder.



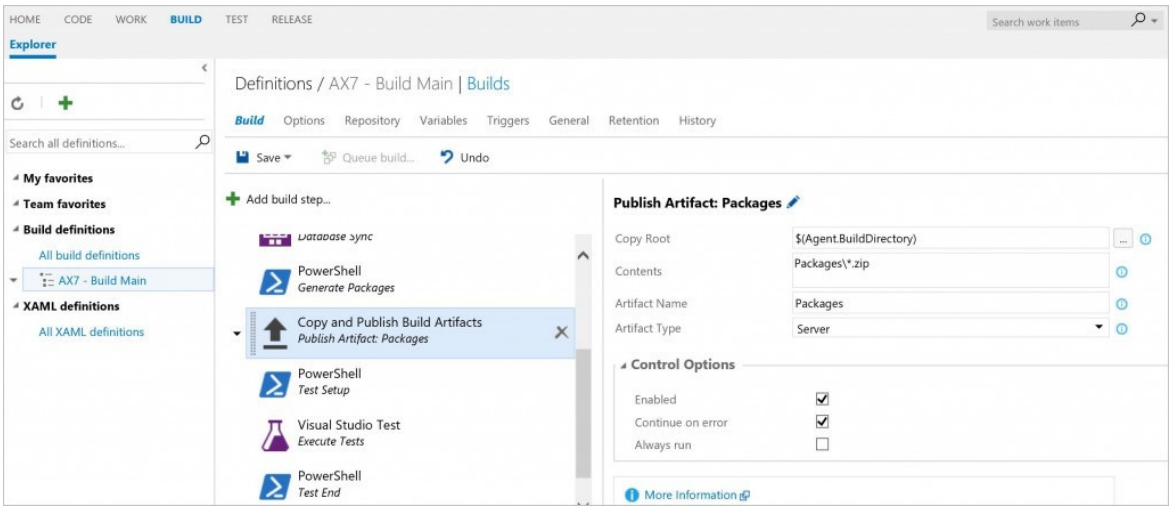
2. Build the solution to discover and build all modules under "Trunk/Main" branch as shown below.



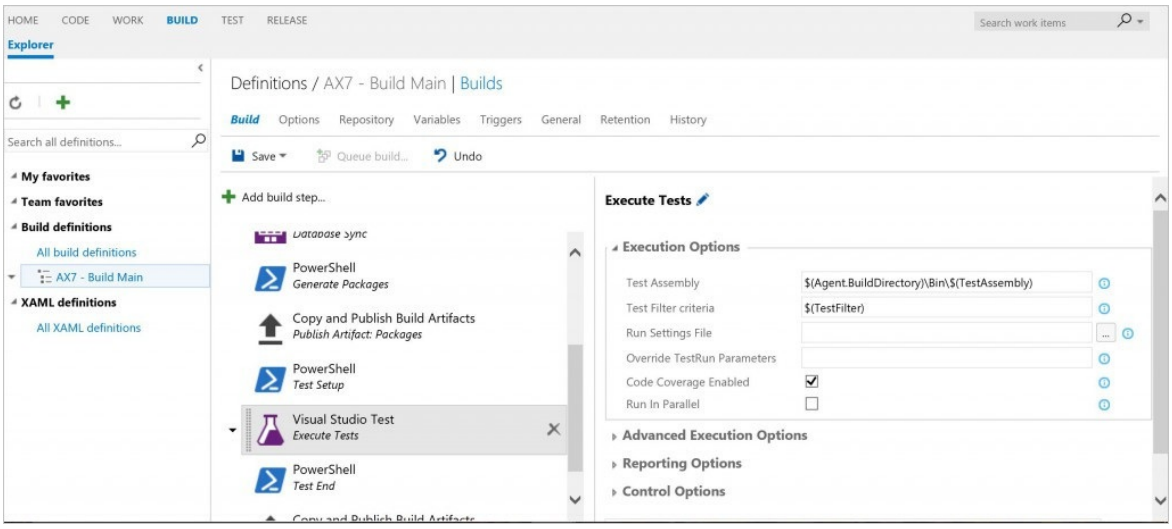
3. Use "Deploy Report" task to generate reports and deploy on build VM.
4. Use "Database Sync" task to synchronize the database to local SQL on build VM.
5. After the build is successful, create a deployable package that can be used to update sandbox/ staging environment.



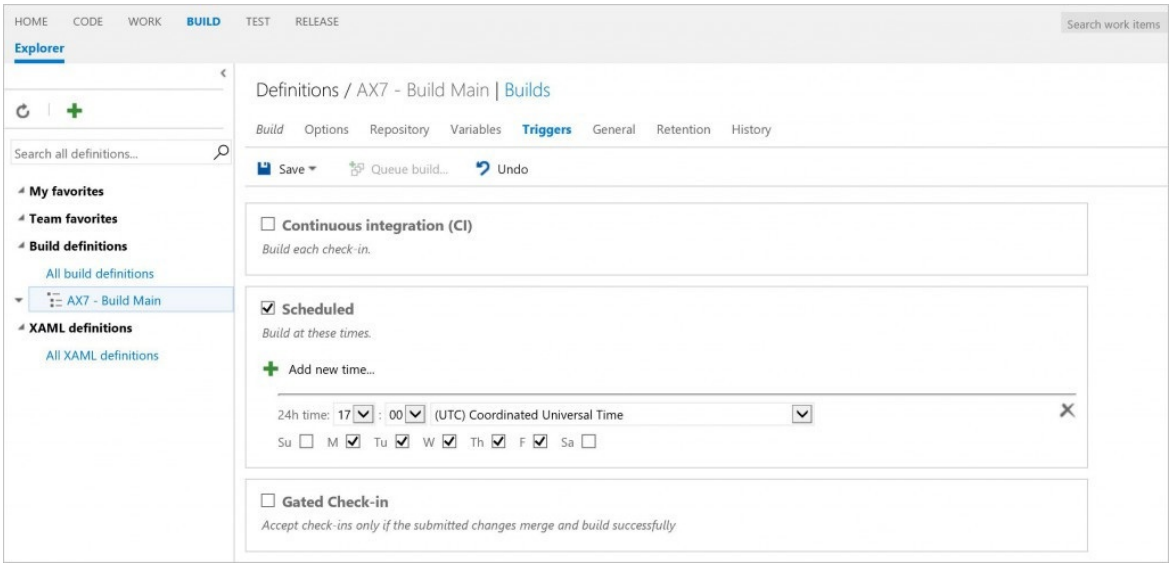
6. "Copy and publish build artifacts" uploads the deployable package to Azure DevOps artifacts location.



7. For test execution, there are three default tasks "Test Setup", "Execute Test" and "Test End".



8. The default build is scheduled to trigger start every day at 5 P.M. You can change trigger as per your team's need to "Continuous" for each check-in.



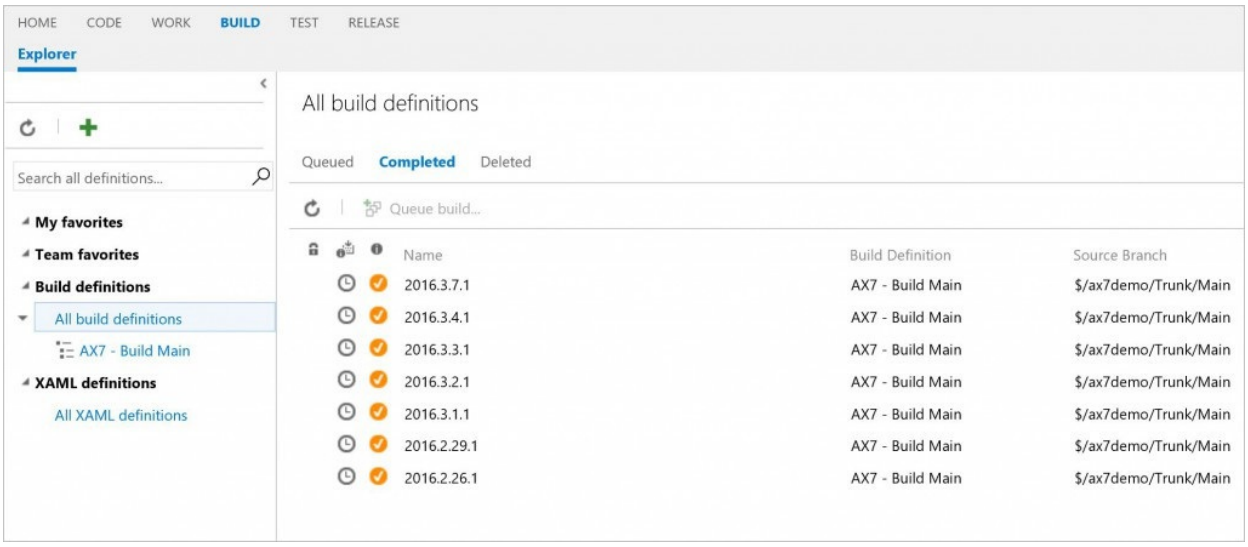
You can make changes to the default configuration, and the build VM will be ready to trigger a build.

Start a build and verify the build and test execution results

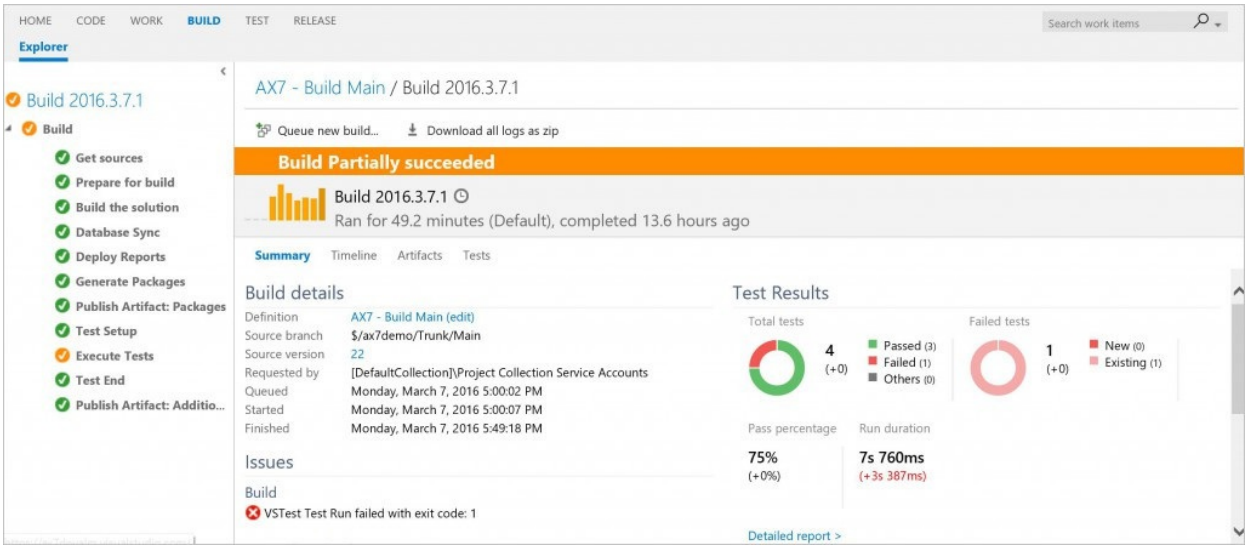
After you review the default build configuration, you can manually trigger a build from Visual Studio IDE or Azure DevOps web interface.

1. Open your browser and connect to the Azure DevOps URL.
2. Login using your credentials.
3. On the home page, under **Recent projects and solutions**, select a project.
4. From top links options, select **BUILD**.
5. On the left panel, select the default build definition instance.
6. Right-click and select **Queue Build** to trigger a build for your module and test module that is already checked into the Azure DevOps source control.

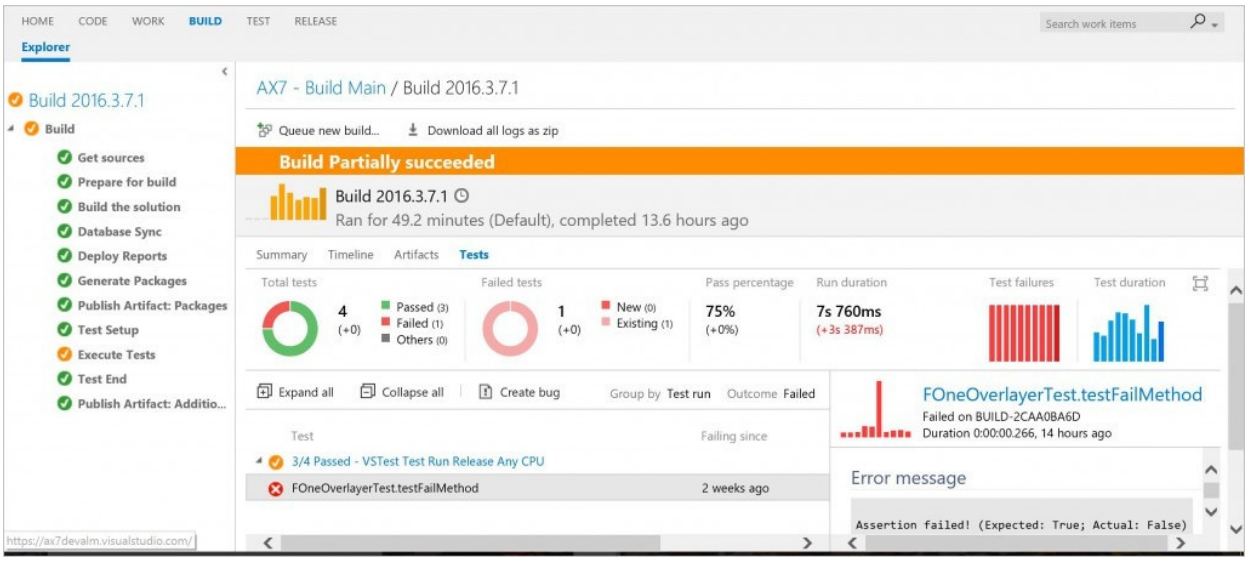
Success or failure for the build will display, as shown by the following examples. View all builds.



Select specific completed build and view success/ failure details.



Click on Test link to visualize test execution failure.



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

SysTest Filtering using class and method attributes

2/18/2021 • 2 minutes to read • Edit Online

Starting with Platform update 12, the SysTest framework contains improvements to the SysTest class and method attributes in X++. These improvements also change how these attributes work in the Visual Studio test window as well as the Visual Studio Test Console, which is the tool used in the automated build process. The SysTest framework now supports the major test attributes in the adaptor to be on par with the MSTest framework adaptor. This includes attributes like **Category**, **Owner**, **Priority**, and **Test Property**.

TestCategory

The **Category** attribute, **SysTestCategory**, was already available in previous platform updates using the **SysTestCategory** attribute. Starting with Platform update 12, you can specify multiple categories on both the class level and the individual method level. Additionally, **TestCategory** is enabled for filtering in the Visual Studio Test Console. This means that you can create build pipelines with test filters on specific categories. You can use the **TestFilter** variable in the build pipeline. For example, to run tests only with a category **Nightly**, set the variable to **TestCategory=Nightly**.

Priority

The **Priority** attribute **SysTestPriority**, which requires an integer value, is now available. A priority can only be specified once, but is supported on both the class and method level, with method level taking precedence over class level. The priority is also exposed as a test filter. This means that you can use the **TestFilter** variable in the build pipeline. For example, to only run tests with a priority of **1**, set the variable to **Priority=1**.

Owner

The **Owner** attribute, **SysTestOwner**, has also been added. This attribute was technically already supported for filtering in the **Test Toolbox** window, but the attribute itself was missing in X++. Similar to **Priority**, an owner can only be specified once and is supported on both the class and method level, with the method level taking precedence. **Owner** is not available as a test filter for the console, which aligns with the MSTest adaptor for Visual Studio Test Console. However, **Owner** will appear in the **Test Toolbox** window in Visual Studio.

Test Property

The **Test Property** attribute, **SysTestProperty**, has existed in previous releases of the platform, but wasn't fully functional. Unfortunately, the **SysTestProperty** attribute exists in the **Application Foundation** package as opposed to the other attributes which exist in the **Test Essentials** package. Because of our commitment to backward compatibility of the platform, we currently cannot move this attribute to its expected location, so your code will need a reference to the **Application Foundation** package to use it. **SysTestProperty** specifies a property and a value (two strings), and can now be used in the **Test Toolbox** window in Visual Studio. **Test Property** can be specified multiple times, and can exist on both the class and method level. **Test Property** is not available for test filtering in the Visual Studio Test Console, in line with the MSTest adaptor.

For advanced filtering syntax that can be used with the Visual Studio Test Console and to review the filtering example for the MSTest framework, see Running selective unit tests.

NOTE

For test filtering purposes, the SysTest framework allows you to filter on **FullyQualifiedName** and **Name**.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Acceptance test library resources

2/18/2021 • 2 minutes to read • Edit Online

The Acceptance test library (ATL) is an X++ test library that offers the following benefits:

- It lets you create consistent test data.
- It increases the readability of test code.
- It provides improved discoverability of the methods that are used to create test data.
- It hides the complexity of setting up prerequisites.
- It supports high performance of test cases.

Example of a test that is written in ATL

```
// Create the data root node
var data = AtlDataRootNode::construct();

// Get a reference to a well-known warehouse
var warehouse = data.invent().warehouses().default();

// Create a new item with the "default" setup using the item creator class. Adjust the default warehouse
before saving the item.
var item = items.defaultBuilder().setDefaultWarehouse(warehouse).create();

// Add on-hand (information about availability of the item in the warehouse) by using the on-hand adjustment
command.
onHand.adjust().forItem(item).forInventDims([warehouse]).setQty(100).execute();

// Create a sales order with one line using the sales order entity
var salesOrder = data.sales().salesOrders().createDefault();
var salesLine = salesOrder.addLine().setItem(item).setQuantity(10).save();

// Reserve 3 units of the item using the reserve() command that is exposed directly on the sales line entity
salesLine.reserve().setQty(3).execute();

// Verify inventory transactions that are associated with the sales line using the inventoryTransactions
query and specifications
salesLine.inventoryTransactions().assertExpectedLines(
    invent.trans().spec().withStatusIssue(StatusIssue::OnOrder).withInventDims([warehouse]).withQty(-7),
    invent.trans().spec().withStatusIssue(StatusIssue::ReservPhysical).withInventDims([warehouse]).withQty(-
3));
```

Concepts

The structure and naming of the classes and methods in ATL are quite rigid. This rigidity helps improve discoverability and also makes it easier to write tests, even in domains that you're unfamiliar with.

The classes are grouped into the following concepts:

- Navigation – Discover entities and test data methods in a familiar hierarchy.
- Test data methods – These methods are used to set up test data.
- Entities – Entities represent data and associated behavior that is perceived as a single unit.
- Creators – Creators let you create specific test data.
- Commands – Commands run business operations.
- Queries – Queries find entities.

- Specifications – Specifications describe expected entities at the end of the test.

Code generation

Queries and specifications help simplify the process of creating entities. For more information, see Acceptance test library Code generation wizard. The **Code generation** wizard can be used to create scenarios and update them.

Further reading

Microsoft has used ATL internally for several years, as the foundation for thousands of tests. For more information see, Best practices for the Acceptance test library and Acceptance test library FAQ.

Navigation concepts for test data

2/18/2021 • 2 minutes to read • Edit Online

To simplify the discoverability of generation methods for test data, a set of navigation objects is introduced. For more information about the generation methods, see Test data methods.

Navigation should start from the root object, the module must be specified, and then the entity must be specified together with the test data methods.

```
data.module().entity().testDataMethod();
```

Examples

```
modelGroup = data.invent().modelGroups().fifo();

itemBuilder = data.products().items().whsBuilder();

data.sales().salesOrders().ensureCanCreate();

data.invent().parameters().enableQualityManagement();
```

Advantages

- Discoverability of data creation application programming interfaces (APIs). IntelliSense helps you get to the helper methods that are relevant for the entity.
- Local references to frequently used nodes (for example, `items.whsBuilder()`). Therefore, long names aren't required.

Root navigation object

The root navigation object is the starting point for finding the test data methods that are required. The root navigation object exposes modules where test data methods are defined.

Variable naming

The suggested name for variables that reference the navigation root is `data`.

Class naming

The root class is named `AtlDataRootNode`.

Module navigation object

The module navigation objects let you group test data methods by relevant modules. Module navigation objects can expose entity navigation objects.

Navigation node naming

Module names should be based on the names of the modules on the main menu. However, a short version or an abbreviation should be used to support brevity of test code.

Examples

- **Sales and marketing** module: `data.sales()`
- **Inventory management** module: `data.invent()`

Class naming

`AtlData<ModuleName>`

Examples

- **Sales and marketing** module: `AtlDataSales`
- **Procurement and sourcing** module: `AtlDataPurch`
- **Inventory management** module: `AtlDataInvent`
- **Accounts receivable** module: `AtlDataCust`
- **Accounts payable** module: `AtlDataVend`

Entity navigation objects

Entity navigation objects let you group test data methods by relevant entities.

Navigation node naming

The plural of the entity name should be used as the name of the entity navigation node.

Examples

```
data.products().items();

data.whs().warehouses();
```

Class naming

`AtlData<ModuleName><EntityNamePlural>`

Examples

```
AtlDataProductsItems

AtlDataInventChargeGroups
```

Notes

The same entity navigation object can be exposed from multiple modules when this approach makes sense. For example, `AtlDataProductsItems` is exposed from both `data.product()` and `data.invent()`.

Helper navigation objects

Sometimes, a test data method isn't specific to any entity. In this case, a helpers node can be exposed at the module level.

Navigation node naming

The helpers navigation node should be named `helpers`.

Examples

```
data.helpers();

data.whs().helpers();
```

Class naming

`AtlData<ModuleName>Helpers`

Examples

```
AtlDataHelpers

AtlDataWHSHelpers
```

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Test data methods

2/18/2021 • 4 minutes to read • Edit Online

Entity and helper navigation objects expose test methods that let you set up test data. This topic provides information about the most common types of test data methods.

Factory methods

Factory methods focus on creating data that doesn't yet exist in the database. There are two types of entity factory methods, `init` methods and `create` methods. An `init` method initializes the entity but doesn't save it to the database. A `create` method initializes the entity and saves it to the database.

Naming convention

`init<EntitySpecification>`

`create<EntitySpecification>`

In this naming convention, `<EntitySpecification>` is the description of the key characteristics of the object that must be created.

Examples

```
salesOrder = data.sales().salesOrders().initDefault();

purchaseOrder = data.purch().purchaseOrders().createDefault();
```

Best practices

The `create` method should always call the `init` method that has the same entity specification.

Example

```
public AtlEntitySalesOrder createDefault()
{
    AtlEntitySalesOrder salesOrder = this.initDefault();
    salesOrder.save();
    return salesOrder;
}
```

Prerequisite data

The `init` method should take care of setting up prerequisites.

Before some entities can be created, specific prerequisites must be set up. In these cases, the `ensure` method must be called before the entity is initialized. You must also subscribe to all the entity events that require automatic setup of prerequisites.

Example

```
public AtlEntitySalesOrder initDefault()
{
    AtlEntitySalesOrder salesOrder;

    this.ensureCanCreate();

    salesOrder = new AtlEntitySalesOrder();
    salesOrder.parmCustomer(data.cust().customers().default());

    _salesOrder.postingInvoice += eventhandler(this.ensureCanPostInvoice);
    _salesOrder.postingPackingSlip += eventhandler(this.ensureCanPostPackingSlip);
    _salesOrder.releasingToWarehouse += eventhandler(this.ensureCanReleaseToWarehouse);
}
```

Builder methods

Builder methods are responsible for initializing creator objects that will be used to create data that doesn't yet exist in the database.

Naming convention

`<EntitySpecification>Builder`

In this naming convention, `<EntitySpecification>` is the description of the key characteristics of the object that must be created.

Example

```
catchWeightItem = data.invent().items().cwBuilder();
```

Well-known data methods

Well-known data methods provide a way to reference an entity that is set up in a specific way. If the entity doesn't exist in the database, it's created.

Naming convention

`<EntitySpecification>`

In this naming convention, `<EntitySpecification>` is the description of the key characteristics of the object that must be retrieved.

Example

```
fifo = data.invent().modelGroup().fifo();
```

In this example, the contract of the method specifies that the model group should use first in, first out (FIFO) as the inventory model. The rest of the settings can be left at their default values.

Sometimes, a real-world name communicates the contract better.

```
pieces = data.common().units().pieces();
```

In this example, it's clear that **pieces** is a unit of measure of the "quantity" class, and that is has a decimal precision of 0 (zero).

Contract of well-known data methods

Here are a few things to remember about the common contract of the well-known data methods.

- Two calls to the same well-known data method should provide the caller with the reference to the same entity.

```
fifo1 = data.invent().modelGroups().fifo();
fifo2 = data.invent().modelGroups().fifo();
fifo1.InventModelGroupId == fifo2.InventModelGroupId;
```

- Creation of a test entity isn't always worth the effort. If a test entity isn't created, the corresponding record buffer should be returned from the well-known data method. For example, if you don't invest the time and effort to create the Site entity, the `site` well-known data method will return InventSite records.

```
InventSite site = data.invent().sites().default();
```

- Well-known data methods can take IDs as optional parameters when this approach makes sense.

```
item1 = data.products().items().default('Item1');
item2 = data.products().items().default('item2');
```

Implementation

If there is already a builder or a factory method that is named `<EntitySpecification>`, it should be used as the internal implementation to create the well-known entity.

Example

```
public InventTable whsBatchAbove(ItemId _itemId = this.whsBatchAboveItemId())
{
    InventTable whsItem = InventTable::find(_itemId, true);
    if (!whsItem)
    {
        whsItem = this.whsBatchAboveBuilder().setItemId(_itemId).create();
    }
    return whsItem;
}
```

Ensure methods

Ensure methods are responsible for setting up prerequisites that are required in order to create an entity or run a business operation.

Naming convention

`ensureCan<ExecuteBusinessOperation>`

In this naming convention, `<ExecuteBusinessOperation>` is a verb that describes the business operation.

Examples

```
data.sales().salesOrders().ensureCanCreate();

data.purch().purchaseOrders().ensureCanPostProductReceipt();
```

Implementation

Figuring out prerequisites for a complex business operation, such as invoice posting, can be complicated and requires lots of knowledge about the feature area.

Example

```
public void ensureCanCreate()
{
    data.helpers().setNumberSequenceReference(extendedTypeNum(InventDimId));
}
```

Automatic prerequisite setup

To enable automatic prerequisite setup, you must call the `ensure` methods in the appropriate `init` method. For more information, see the Factory methods section earlier in this topic.

Query methods

Query methods are responsible for initializing new queries for the entity type of the navigation node that they are defined on.

Naming convention

`query`

Examples

```
loadLinesQuery = data.whs().loadLines().query();

purchaseLinesQuery = data.invent().transferOrderLines().query();
```

Specification methods

Specification methods are responsible for initializing new specification objects for the entity type of the navigation node that they are defined on.

Naming convention

`spec`

Example

```
loadLinesSpec = data.whs().loadLines().spec();
```

Find methods

Find methods let you find an entity based on the primary key.

Naming convention

`find`

Example

```
salesOrder = data.sales().salesOrders().find(salesOrderId);
```

Automatic prerequisite setup

If the entity has query support, the implementation should use the query that has already set up prerequisite support. Otherwise, after you find the record buffer and initialize the new instance of the entity, you should subscribe `ensure` methods to the business operation events of the entity.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Entities in the Acceptance test library

2/18/2021 • 6 minutes to read • Edit Online

A test entity class represents data and behavior that are perceived as a single concept. Test entity classes are based on pages such as **Sales order**, **Transfer order**, and **Released product**. The test entity classes expose the properties that are most often used in test scenarios, and the behavior that is most important from the perspective of test data setup and scenario tests.

An entity in the Acceptance test library (ATL) **must** have the following methods:

- Property methods that are used to get and set entity properties.
- Fluent setter methods that enable entity properties to be set in a fluent manner.
- A method that saves the entity to the database.

An entity in ATL *can* have the following methods:

- Action methods that are used to expose business operations that are relevant to the entity.
- Query methods that are used to enable navigation to components and related entities.

Naming convention

`[AtlEntity]<ModuleName><EntityName>`

In this naming convention:

- `<ModuleName>` is based on the names of the modules in main menu. You should use a short version or an abbreviation to support brevity of test code.
- `<EntityName>` is based on the user interface (UI) names instead of the table names. For example, use `SalesOrder`, not `SalesTable`.

If an entity has two UI names, it's OK to use the shorter name. For example, you can use `Item` instead of `ReleasedProduct`, because these names are used interchangeably.

Examples

```
AtlEntitySalesOrder

AtlEntityTransferOrderLine
```

Property methods

One of the main purposes of a test entity is to expose data. The properties of the entity can be set or retrieved by using `parm` (property) methods.

Primitive type properties

Create a `parm` method to expose a primitive type property.

Example

```
public SalesQty parmQuantity(SalesQty _qty = 0)
{
    if (!prmisDefault(_qty))
    { // setter
        salesLine.SalesQty = _qty;
        this.onSalesQtyOrInventDimChange();
    }
    return salesLine.SalesQty;
}
```

Entity references

If there is a customer entity that is named `AtlEntityCustomer`, for example, a reference to `customer` should be exposed as a property method on the `AtlEntitySalesOrder` entity.

```
public AtlEntityCustomer parmCustomer(AtlEntityCustomer _custTable = null)
```

The property method can be used as either a setter or a getter.

```
salesOrder.parmCustomer(customer); // setter

customer = salesOrder.parmCustomer(); // getter
```

Entity reference methods naming conventions

The `parm` prefix should be used to identify property methods. When you expose an entity reference property, use the UI name of the field instead of the Application Object Tree (AOT) name. If the UI name includes the `Id`, `Code`, or `Number` suffix, omit the suffix. For example, use `parmItem` instead of `parmItemNumber`.

Record references

If the customer entity hasn't yet been created and won't be created in the near future, the reference property should expose the corresponding record buffer (`CustTable`).

```
public CustTable parmCustomer(CustTable _custTable = null)
```

Record reference naming conventions

Use the same naming conventions that are used for entity references.

Id references

In addition to having an entity or record reference, you can introduce the `Id` reference property.

```
public CustAccount customerId(CustAccount _custTable = null)
```

Don't introduce `Id` references unless you also introduce corresponding entity or buffer references. `Id` references are shortcuts to the entity or buffer reference methods. The implementation of `Id` references should delegate the call to the entity or buffer reference.

Id reference naming conventions

Use the UI name if it includes terms such as `Id`, `Number`, `Account`, `Code`, or `Name`. Otherwise, add an appropriate suffix to the name of the entity or record reference.

Id reference methods contract

The `Id` reference method always uses the provided `Id` to find the referenced entity, and it delegates the call to the entity or record reference method. If no entity or record is found based on the specified `Id` value, an error message is thrown.

Fluent setter methods

Create fluent setter methods to support the fluent initialization and modification of entities.

Declaration example

```
public AtlEntitySalesLine setQty(SalesQty _qty)
```

Code example

```
salesLine.setItem(batchItem).setInventDims([warehouse]).setQty(10).save();
```

Naming convention

`set<PropertyName>`

In this naming convention, `<PropertyName>` should match what is used in the name of the corresponding property method.

Action methods

Entities represent not only data but also relevant actions. Actions can be implemented either as a simple action method or as a command object initializer.

Simple action methods

Simple action methods represent a complete action. They should not be fluently chained. The exception is the `save` method, which should be fluent.

Naming convention

`<ExecuteBusinessOperation>`

In this naming convention, `<ExecuteBusinessOperation>` is a verb that represents the business operation. It should be the same term that is used on the menu item in the UI.

Examples

```
salesOrder.save();

salesOrder.postInvoice();
```

Command object initializers

Command object initializers return a command object that lets you specify parameters of the command and run it.

```
transferLine.pick().setQty(10).setWMSLocation(bulkLocation).execute();
```

Naming convention

`<ExecuteBusinessOperation>`

In this naming convention, `<ExecuteBusinessOperation>` is a verb that represents the business operation. It should be the same term that is used on the menu item in the UI.

Examples

```
salesOrder.pick().execute();

purchaseOrder.register().execute();
```

Action entities

Some actions that are available for an entity can be considered entities themselves. Vendor invoices are one example. Before you post an invoice, you might want to set up parameters of the invoice, edit lines, and save the invoice for later. For these commands, you can introduce a separate entity class.

Naming convention

`new<ActionName>`

In this naming convention, `<ActionName>` is a noun that represents the business operation. The name should be the UI name of the business operation.

Example

```
receipt = transfer.newReceipt().setEditLines(true).setExplodeLines(true);
receipt.lines().withBatch(batch1).single().setReceiptQty(6).setScrapQty(1).save();
receipt.lines().withBatch(batch2).single().setReceiptQty(4).setScrapQty(1).save();
receipt.post();
```

Class naming convention

`AtlEntity<ModuleName><EntityName><ActionName>`

Example

```
AtlEntityInventTransferOrderReceipt
```

Adding components

Composition is a relationship where the composite entity has sole responsibility for the disposition of the component parts. The relationship between the composite and the component is a strong "has a" relationship, because the composite object takes ownership of the component. Therefore, the composite is responsible for creating and destroying the component parts.

An object instance can be part of only one composite. If the composite object is destroyed, all the component parts must be destroyed. The component parts have no independent existence outside the composite object, and they can't be transferred to another object. Composition enforces encapsulation, because the component parts are usually members of the composite object.

An example of a composite object is a source document that is made up of source document lines.

Example

In the source document example, the document entity serves as the composition root and is responsible for creating any new instances of document lines. In this case, the source document entity will have an `addLine()` method that initializes and returns a new line for the document.

```
public AtlEntitySalesLine addLine()
```

The `addLine()` method adds the line object (`salesLine` in this example) to a collection of lines and returns the parent entity (`SalesOrder` in this example) to preserve the fluency of application programming interfaces (APIs). To create a new line, create a `newLine()` method.

Naming convention for adding components

Methods for adding components should use the UI names of the buttons.

Example

```
salesLine = salesOrder.addLine();
```

Component collections

You can search for components by using query methods.

Naming convention for component collections

Methods for accessing component collections should use the UI names of the grid on the hosting page.

Query methods

Query methods on an entity let you search for components and related entities.

Example

```
transferOrderLine = transferOrder.lines().withItem(item).single();
```

In this example, `lines()` is a query method that returns the `AtlQueryTransferOrderLines` query. This query is already filtered so that it returns only transfer order lines for the transfer order that the `lines()` method was called on.

Naming convention

Use the UI names whenever you can. Abbreviations are acceptable if the UI name is too long to be used in test automation.

Example

```
public AtlQueryWHSLoadLines lines()
{
    return new AtlQueryWHSLoadLines().forLoadId(this.parmLoadId());
}
```

Additional resources

Queries in the Acceptance test library

Acceptance test library commands

2/18/2021 • 2 minutes to read • Edit Online

Command classes are responsible for running business operations. They let you use fluent application programming interfaces (APIs) to set the parameters of these operations.

Naming convention

`AtlCommand<ModuleName><EntityName><ExecuteBusinessOperation>`

In this naming convention:

- `<ModuleName>` is based on the names of the modules on the main menu. You should use a short version or an abbreviation to support brevity of test code.
- `<EntityName>` is optional and is used when the command applies to different types of entities.
- `<ExecuteBusinessOperation>` is a verb that represents the name of the business operation.

Examples

```
AtlCommandInventMark

AtlCommandSalesReturnOrderLineRegister
```

Implementation

Command objects that are returned should implement the `AtlICommand` interface and should inherit from the `AtlCommand` class.

Command objects should provide fluent setter methods that are used to set the parameters of the command.

Example

```
salesLine.pick().setInventDims([locationOut]).setQty(pickedQty).execute();
```

Fluent setter methods

Commands allow for two types of fluent setter methods, `for` methods and `set` methods:

- **for** – These methods are used for command parameters that represent the entities that the command applies to.

  For example, the invoice command can apply to sales orders. Therefore, a `for` method is used to set the sales order that the invoice command applies to.

- **set** – These methods are used for all other parameters of the command.

  For example, when you reserve a sales line, you usually specify a quantity. The quantity isn't something that the command applies to but is instead a simple parameter of the command. Therefore, a `set` method is used to specify the quantity parameter of the reservation command.

Naming convention

`for<CommandParameterName>`

`set<CommandParameterName>`

In this naming convention, `<CommandParameterName>` is the name of the parameter that is being set for the command by using the fluent method.

Examples

```
onHandAdjustment.forItem(item).setQuantity(10).execute();

picking.forSalesLine(salesLine).setInventDims([warehouse, batch1]).setQuantity(10).execute();
```

Creators in the Acceptance test library

2/18/2021 • 2 minutes to read • Edit Online

Creator classes provide fluent application programming interfaces (APIs) that are used to create test data.

Naming convention

`AtlCreator<ModuleName><EntityName>`

In this naming convention:

- `<ModuleName>` is optional and is based on the names of the modules on the main menu. You should use a short version or an abbreviation to support brevity of test code.
- `<EntityName>` is optional and is used when the command applies to different types of entities.

Examples

```
AtlCreatorCostGroup

AtlCreatorCustomer
```

Fluent setter methods

Creator classes should provide fluent setter methods that are used to set the properties of the entity that is being constructed.

Naming convention

`set<EntityPropertyName>`

In this naming convention, `<EntityPropertyName>` is the name of the property that is being set for the entity by using the fluent method.

Example

```
item = new AtlCreatorProductsReleasedVariant()
    .setItemId('DemoItem')
    .setColor(ecoResColor)
    .setStyle(ecoResStyle)
    .setConfig(ecoResConfig)
    .setSize(ecoResSize)
    .create();
```

When should creators be used instead of entities?

For information that will help you choose between entities and creators, see Should I implement an entity or a creator class.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Queries in the Acceptance test library

2/18/2021 • 2 minutes to read • Edit Online

A query class provides fluent application programming interfaces (APIs) that are used to find an instance of the corresponding entity, based on various criteria. Query classes are often used in validation scenarios. They are usually used together with specifications.

Naming convention

`AtlQuery<ModuleName><EntityNamePlural>`

In this naming convention:

- `<ModuleName>` is optional and is based on the names of the modules on the main menu. However, a short version or an abbreviation should be used to support brevity of test code.
- `<EntityNamePlural>` is the plural version of the entity name.

Examples

```
AtlQueryWHSLoadLines

AtlQueryInventTransferOrderLines
```

Implementation

Query classes inherit from the `AtlQuery` class that is common to all queries.

Fluent setters

Query classes should provide fluent setter methods to specify ranges for the query.

Example

```
loadLine = data.whs().loadLines().query().forSalesOrder(salesOrder).single();
```

Queries allow for two types of fluent setter methods, `for` methods and `with` methods:

- **for** – These methods are used for filters of the query that act as parents of composition or aggregation relationships. For example, the sales lines query exposes a `for` method to filter sales lines for a specific sales order.
- **with** – These methods are used for all other ranges of the query.

Naming convention

`for<QueryRangeName>`

`with<QueryRangeName>`

In this naming convention, `<QueryRangeName>` is the name of the field that the range is applied on.

Examples

```
loadLine = data.whs().loadLines().query().forLoad(load).withInventQty(10).single();

transferLine =
data.invent().transferOrderLines().query().forTransferOrder(transferOrder).withInventDims([batch1]).single()
;
```

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Specification classes

2/18/2021 • 2 minutes to read • Edit Online

A specification class provides fluent application programming interfaces (APIs) that are used to define the set of criteria that an entity should meet. Specifications are often used in validation scenarios. They are usually used together with query classes.

An advantage of specification classes is that the validation code becomes very concise and expressive. Basically, you can do multiple validations in a single line of code.

Naming convention

`AtlSpec<ModuleName><#EntityName>`

In this naming convention:

- `<ModuleName>` is optional and is based on the names of the modules on the main menu. However, a short version or an abbreviation should be used to support brevity of test code.
- `<#EntityName>` represents the name of the entity that is used throughout the Acceptance test library (ATL).

Examples

```
AtlSpecWHSLoadLine

AtlSpecWHSWorkLine
```

Implementation

Specification classes should provide fluent setter methods to specify various criteria of the specification.

Example

The following code verifies that the work contains six lines that meet the specified criteria. For example, the first line should have **1** as the line number of **1**, **Pick** as the work type, **1** as the quantity, **Closed** as the status, and **bulk** as the location.

```
work.lines().assertExpectedLines(
    workLines.spec().withLineNum(1).withWorkType(WHSWorkType::Pick).setQuantity(1)
        .setStatus(WHSWorkStatus::Closed).setLocation(locations.bulk()),
    workLines.spec().withLineNum(2).withWorkType(WHSWorkType::Pick).setQuantity(1)
        .setStatus(WHSWorkStatus::Closed).setLocation(locations.floor()),
    workLines.spec().withLineNum(3).withWorkType(WHSWorkType::Put)
        .setQuantity(2).setStatus(WHSWorkStatus::Closed).setLocation(locations.stage()),
    workLines.spec().withLineNum(4).withWorkType(WHSWorkType::Pick).setQuantity(2)
        .setStatus(WHSWorkStatus::Cancelled).setLocation(locations.stage()),

workLines.spec().withLineNum(5).withWorkType(WHSWorkType::Put).setQuantity(2).setStatus(WHSWorkStatus::Cance
lled)
);
```

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Acceptance test library Code generation wizard

2/18/2021 • 2 minutes to read • Edit Online

The Acceptance test library (ATL) code generator quickly generates and updates new ATL entities, queries, and specifications, based on tables and data entities.

Create the AtlEntity class by using the wizard

Follow these steps to create the `AtlEntity` class by using the **Code generation** wizard.

1. In Microsoft Visual Studio, open the table in the designer window.
2. Right-click the name of the table, and then, on the **Add-ins** menu, select **Generate ATL Entity**.
3. Select the fields that should be included in the `AtlEntity` class, and then select **Add**.
4. Rename the entity and the fields as you require.
5. Select **Generate** to create the class.

Additional optional steps

When you create the `AtlEntity` class, you can also complete these tasks:

- Add required actions for the scenario.
- Add a `default` method to `AtlData` classes.
- Override the `setMainRecordField` method to call the `modifiedField(_fieldId)` method on the table.

```
protected void setMainRecordField(FieldId _fieldId, anytype _value)
{
    super(_fieldId, _value);
    common.modifiedField(_fieldId);
}
```

Create the AtlQuery class by using the wizard

Follow these steps to create the `AtlQuery` class by using the **Code generation** wizard.

1. In Visual Studio, open the table in the designer window.
2. Right-click the name of the table, and then, on the **Add-ins** menu, select **Generate ATL Query**.
3. Select the fields and relations that should be included in the `AtlQuery` class, and then select **Add**.
4. Rename the query, the fields, and the relations as you require.
5. Select **Generate** to create the class.

Additional optional steps

When you create the `AtlQuery` class, you can also add a `query` method to the `AtlData` class that returns an instance of the `AtlQuery` class that you created earlier in this topic.

Create the AtlSpec class by using the wizard

Follow these steps to create the `AtlSpec` class by using the **Code generation** wizard.

1. In Visual Studio, open the table in the designer window.
2. Right-click the name of the table, and then, on the **Add-ins** menu, select **Generate ATL Specification**.

3. Select the fields that should be included in the `AtlSpec` class, and then select **Add**.
4. Rename the specification and the fields as you require.
5. Select **Generate** to create the class.

Additional optional steps

Add a `spec` method to the data class that returns an instance of the `AtlSpec` class that you created earlier in this topic.

Best practices for the Acceptance test library

2/18/2021 • 2 minutes to read • Edit Online

Use var and declare variables inline

- Use the `var` keyword (type inference).
- Declare variables inline instead of in a separate statement.

Do this

```
var item = items.default();
var salesOrder = data.sales().salesOrders().createDefault();
var salesLine = salesOrder.addLine().setItem(item).setInventDims([warehouse]).setQuantity(10).save();
```

Don't do this

```
InventTable item;
AtlEntitySalesOrder salesOrder;
AtlEntitySaleOrderLine salesLine;
…
item = items.default();
salesOrder = data.sales().salesOrders().createDefault();
salesLine = salesOrder.addLine().setItem(item).setInventDims([warehouse]).setQuantity(10).save();
```

Justification

The advantages of using `var` are that you write less code, you don't have to remember exact type names, and the test logic isn't cluttered with unimportant information. Overall, the test code easier to read.

In the previous example, it doesn't matter whether `item` is of the `ItemId`, `InventlTable`, or `AtlEntityInventItem` type. The important detail is that you're creating a sales line that has a well-known default item. The exact types of the `salesOrder` and `salesLine` variables aren't important. The contracts of these types are clear from the naming and usage.

Considerations

- Don't use type inference if you want compilation to fail if the return type of a method changes.
- Don't use type inference if you can't invent meaningful variable or method names.

Use entities instead of IDs as method parameters

Well-known data methods, creator methods, and `init` methods usually return records or entities instead of IDs. We recommend that you use records or entities as method parameters.

Do this

```
var salesLine = salesOrder.addLine().setItem(item).save();
```

Don't do this

```
var salesLine = salesOrder.addLine().setItemId(item.ItemId).save();
```

Justification

The code is easier to read, because it isn't cluttered with unimportant technicalities.

Considerations

If you know only the ID, use the method that takes the ID as an argument.

Use navigation node shortcuts

When you automate a new domain area, introduce a base class that holds shortcuts to the most frequently used navigation objects in that area.

For example, for the warehouse management area, there is a base class that is named `AtlWHSTestCase`. It contains shortcuts to `data.whs()`, `data.invent()`, `data.invent().items()`, `data.invent().units()`, and other navigation objects. The shortcuts simplify your test code.

```
class AtlWHSTestCase extends SysTestCase
{
    AtlDataRootNote          data;
    AtlDataInvent            invent;
    AtlDataInventOnHand      onHand;
    AtlDataProductItems      items;
    AtlDataWHS               whs;

    protected void initDataSetupReferences()
    {
        data = new AtlDataRootNode();
        invent = data.invent();
        onHand = data.invent().onHand();
        items = data.invent().items();
        whs = data.whs();
```

It also makes sense to introduce shortcuts that are shared among many test methods in the same class.

Do this

```
class WHSMinMaxReplenishmentScenarioTest extends AtlWHSTestCase
…
    var item = items.default();
    var warehouse = invent.warehouses().default();
```

Don't do this

```
class WHSMinMaxReplenishmentScenarioTest extends SysTestCase
…
    var item = data.invent().items().default();
    var warehouse = data.invent().warehouses().default();
```

Considerations

You don't have to create a shortcut for every navigation node that you need. However, consider creating them for the navigation nodes that are frequently used.

Acceptance test library FAQ

2/18/2021 • 2 minutes to read • Edit Online

Which fluent prefix should I use: set, for, or with?

Depending on the class that you want to add a fluent method to, different rules might apply:

- Entities
- Creators
- Commands
- Queries
- Specifications

Should I implement an entity or a creator class?

For most entities, the effort of creating an entity class is the same as the effort of creating a creator class. Therefore, the entity class should be created. However, in some cases, the process of creating an entity might not be straightforward. A good example is the Item entity. Because more than ten different tables make up the Item entity, it's hard to create the entity class. Because you will almost never have to update existing items in your test cases, it's OK if you just have a creator class, which is much easier to implement.

Does the order of the chained fluent setters matter?

In most of the cases, the order of the chained fluent setters doesn't matter. However, be aware that defaulting occurs at the time of the call to the setter method. Therefore, a change in the order of the methods can produce different results. Here is an example for the sales line.

Option 1

```
salesLine.setQuantity(10).setUnitPrice(100).setAmount(2000).save()
```

This option produces a sales line where the amount == 2,000, because the amount is set last.

Option 2

```
salesLine.setAmount(2000).setQuantity(10).setUnitPrice(100).save()
```

This option produces a sales line where the amount == 1,000, because after you set the unit price, the amount is set to quantity × price by default.

Can I use ATL for tests that run on the empty data set?

The Acceptance test library (ATL) can be used on the empty data set without issues. The automatic setup of prerequisites is done on demand. For example, prerequisites for invoice posting will be set up only during the first call to `salesOrder.postInvoice()`. For more information, see Ensure.

`Ensure` methods can also be called in the `setUpTestCase` method to improve performance if more than one test in your test class must post invoices.

Can I use ATL for unit tests?

ATL should be used mostly for data setup and validation in integration and component tests. However, in some cases, it's also used for unit tests.

Why should I be cautious about using ATL in unit and component tests?

In some of the more complex entities, such as Sales order, all the business logic that is associated with the `modifiedField`, `insert`, and `update` events is called. Creation of invoice transactions, for example, is also done by running real invoice posting logic. Therefore, the performance of some operations will be slow. However, these issues don't occur for most of the entities that represent master data. Therefore, you should be able to use those entities in any type of test.

There should not be significant overhead if specifications and queries are used to do validation. These artifacts can also be used in unit tests.

Date effectivity

2/18/2021 • 8 minutes to read • Edit Online

This topic provides information about date-effective data entities and data sources, and shows how to create a date-effective entity. It also explains how date effectivity applies to read and write activities.

There are different design patterns for date-effective features that involve data entities. The patterns are classified into two main categories:

- **Date-effective entities** – The entity has at least one date-effective data source, and the entity itself is also date effective.
- **Non-date-effective entities** – The entity itself is not date effective, but it does contain date-effective data sources.

The next sections describe the small list of properties and methods that control the date-effective behavior of entities and their date-effective data sources.

Date-effective entities

The following table describes the properties that control the date-effective behavior of a data entity.

PROPERTY NAME OF THE ENTITY	NODE OF THE PROPERTY	VALUE	DESCRIPTION
ValidTimeStateEnabled	Data entity node in the designer	Yes (or No)	The value **Yes** makes the entity date effective. The entity must have **ValidFrom** and **ValidTo** fields. These fields are mapped to the **ValidFrom** and **ValidTo** fields of a date-effective data source. The value **No** does *not* disable the enforcement of date effectivity on any date-effective tables that are data sources of the entity.
ValidTimeStateKey	Under the data entity node, **Keys** > **EntityKey**	Yes (or No)	The value **Yes** identifies the key that is required to enforce the date-effective values on this particular entity.

Read activities

When date effectivity is set at the data entity level, reads from the entity behave the same way as reads from a table. The entity has **ValidFrom** and **ValidTo** fields that the system applies date filters to during reads.

Query modes and the validtimestate keyword of X++ SQL select

A date-effective entity supports the following three *query modes*, which vary in their use of the X++ **validtimestate** keyword:

- **Default mode** – Current records are returned using `select * from FMVehicleRateEntity; // X++ SQL.`

- **AsOfDate mode** – Records valid for the specified date are returned using `select validtimestate(d1) * from FMVehicleRateEntity;`
- **AsOfDateRange mode** – Records valid for the specified date range are returned using `select validtimestate(d1,d2) * from FMVehicleRateEntity;`

Important: For data entities that aren't themselves date effective, but that have a data-effective data source, only the default query mode is available. This concept is discussed later in this article.

Applying a date filter at the data source level

There are scenarios where date-effective filtering is required outside the data entity, at the data source level. For example, the customer entity (CustTableTestEntity) contains CustTable and LogisticsPostalAddress as data sources, where LogisticsPostalAddress is a date-effective table and CustTable is a regular table. The purpose of a customer entity is to have a list of customers and their active primary addresses, if they have primary addresses. Therefore, the customer entity itself isn't date effective, but it requires date filters on one of the data sources. In this case, the entity isn't marked **ValidTimeStateEnabled**. Instead, an **Apply Date Filter** property is added on the data source. If the value of **Apply Date Filter** is set to **Yes**, date filters are automatically applied to that data source. The following table describes the properties that control the date-effective behavior of a date-effective data source of a data entity.

PROPERTY NAME OF THE DATA SOURCE	NODE OF THE PROPERTY	VALUE	DESCRIPTION
Apply Date Filter	Node of any particular data source of the entity	Yes (or No)	For *reads*, this property controls whether date filters are applied on the entity data source. In this case, the data source should be marked **ValidTimeStateEnabled**. This property value has effect regardless of whether the entity itself is date effective. For *writes*, this property has no effect.

This article describes the use of these date-effective properties and the interactions between them.

State matrixes for reads

This section concerns only reads from the data entity. The following pair of reference matrixes describe the combinations of date-effective states that can exist between a data entity and its data source. Each table contains four cases, and each case discusses two distinct targets. Here are the primary points that you should understand:

- On any given read from the entity, the query mode is the same for both the entity and date-effective data sources.
- If the entity is not date effective, the query mode is limited to the default mode. Therefore, the date-effective data source is accessed only for the current date.
- On the date-effective data source, the **Apply Date Filter** property can be set to **No** to make the data source return all data – past, current, and future.
- For OData, date-effective filters are not applied to the data entity. However, filters on the data source are applied at all code paths.

A. Entity *is* date effective, because ValidTimeStateEnabled = Yes

Data source *is* date effective

Data source is *not* date effective

Apply Date Filter = Yes

- **Entity:** Date filters are applied. Any query mode is supported.
- **Data source:** Filters are applied. Any query mode is supported, but the mode is the same as is coded for the entity.

Non-date-effective data sources aren't affected.

Apply Date Filter = No

- **Entity:** Date filters are applied. Any query mode is supported.
- **Data source:** No date filters are applied.

Non-date-effective data sources aren't affected.

B. Entity is *not* date effective, because ValidTimeStateEnabled = No

Data source *is* date effective

Data source is *not* date effective

Apply Date Filter = Yes

- **Entity:** No date filters are applied.
- **Data source:** Date filters are applied. Only the default query mode is supported, where the X++ **validtimestate** keyword is omitted.

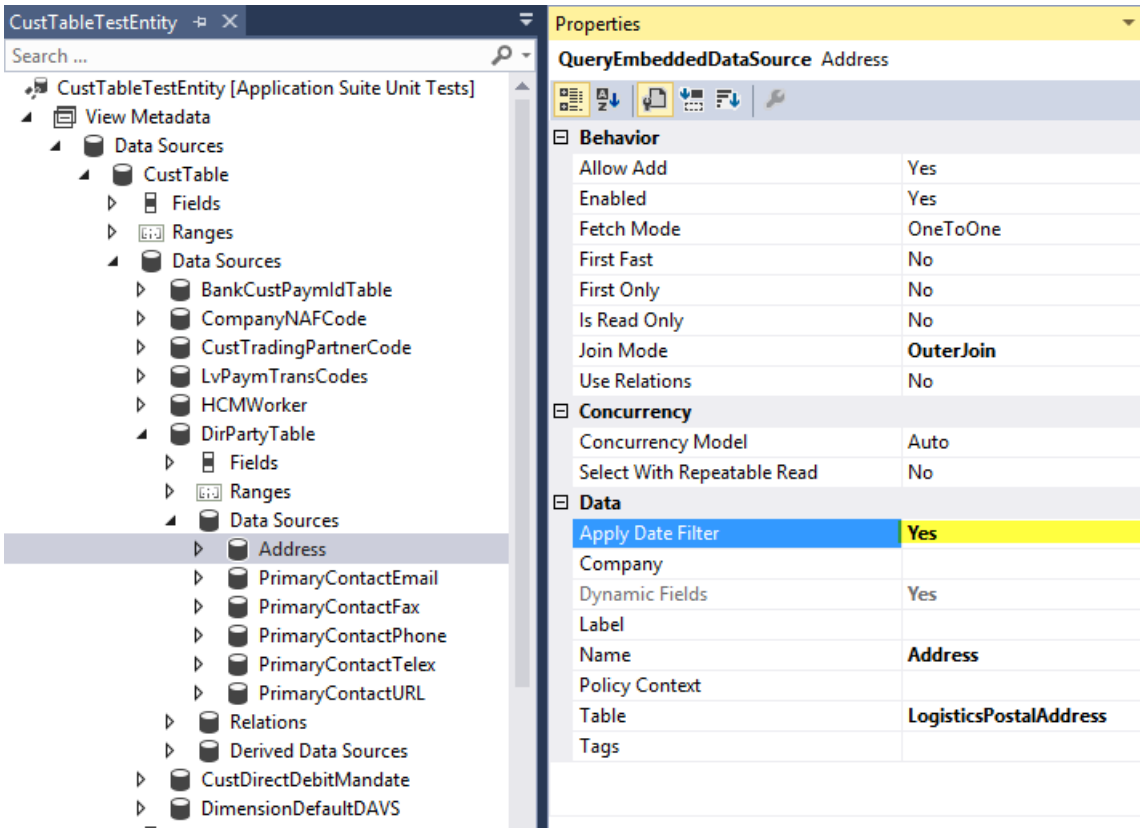
Non-date-effective data sources aren't affected.

Apply Date Filter = No

- **Entity:** No date filters are applied.
- **Data source:** No date filters are applied.

Non-date-effective data sources aren't affected.

The following screen shot shows the **Apply Date Filter** property set to **Yes**. Therefore, date filters will be applied to reads of the **Address** data source.



Write activities

This section describes your options for configuring the behavior of date-effective entities and their date-effective data sources. We will start by reviewing the concept of date-effective tables and contrasting them with date-effective entities. **Date-effective table:** When data is inserted or updated in a date-effective table, the process has the option of calling the **xRecord.validTimeStateUpdateMode** method on the table buffer. The method accepts an element of the **ValidTimeStateUpdate** enumeration. Here are the available element values:

- CreateNewTimePeriod
- Correction
- EffectiveBased

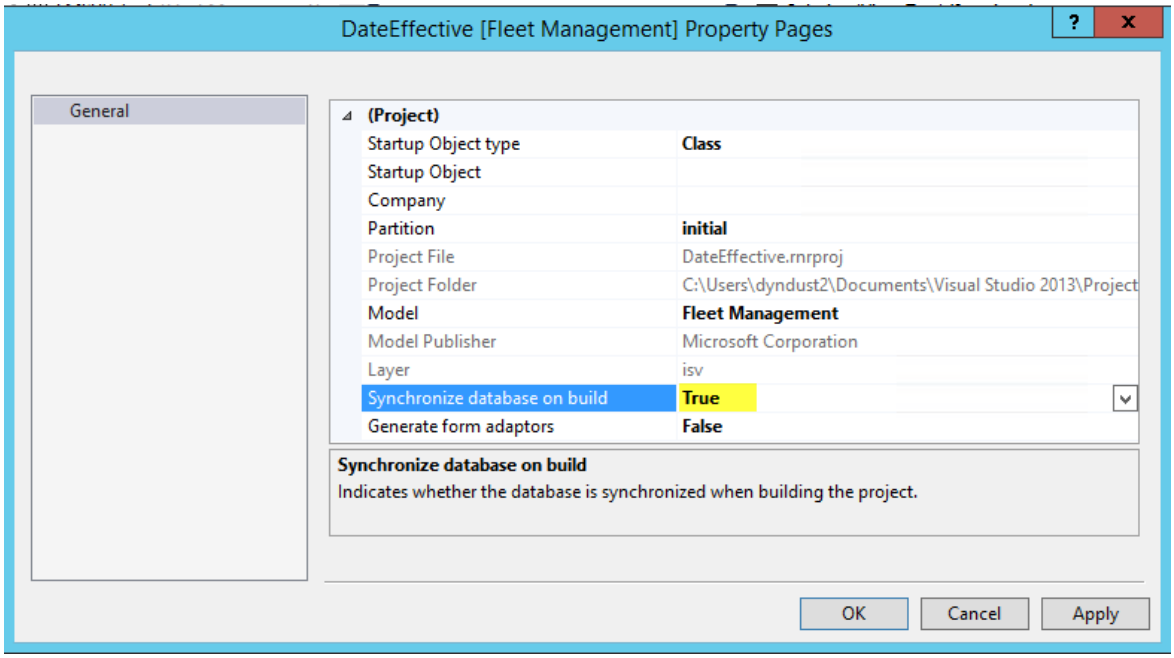
Date-effective entity: By contrast, when data is inserted or updated in a date-effective data entity, the **validTimeStateUpdateMode** method isn't used at the entity level. For writes, the data entity leaves the date-effective processing to the table level. You can use the **Valid Time State Update** property on the entity data source to specify the **validTimeStateUpdateMode** method to use for each data source of the data entity.

Creating a date-effective entity

This section shows how to create a date-effective entity.

Create a new project

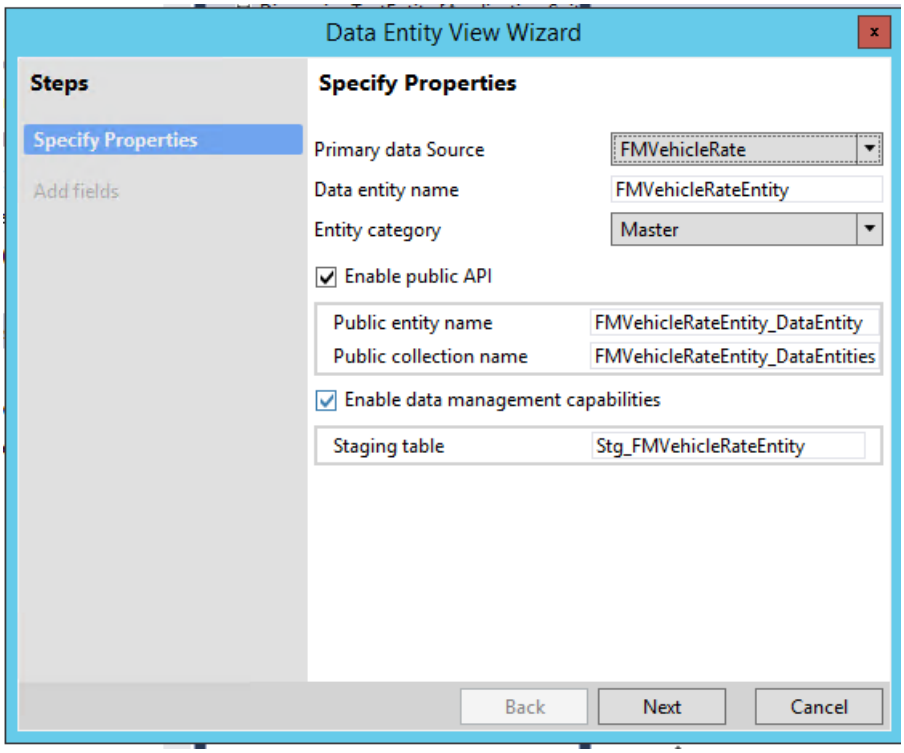
1. Click **File** > **New** > **Project** to create a new project.
2. In Solution Explorer, right-click your project, and then click **Properties**. The **Property Pages** dialog box for your project opens.
3. Change the value of the **Synchronize database on build** property to **True**, and then click **OK**. You must set this property only one time per project.



Add a new data entity to your project

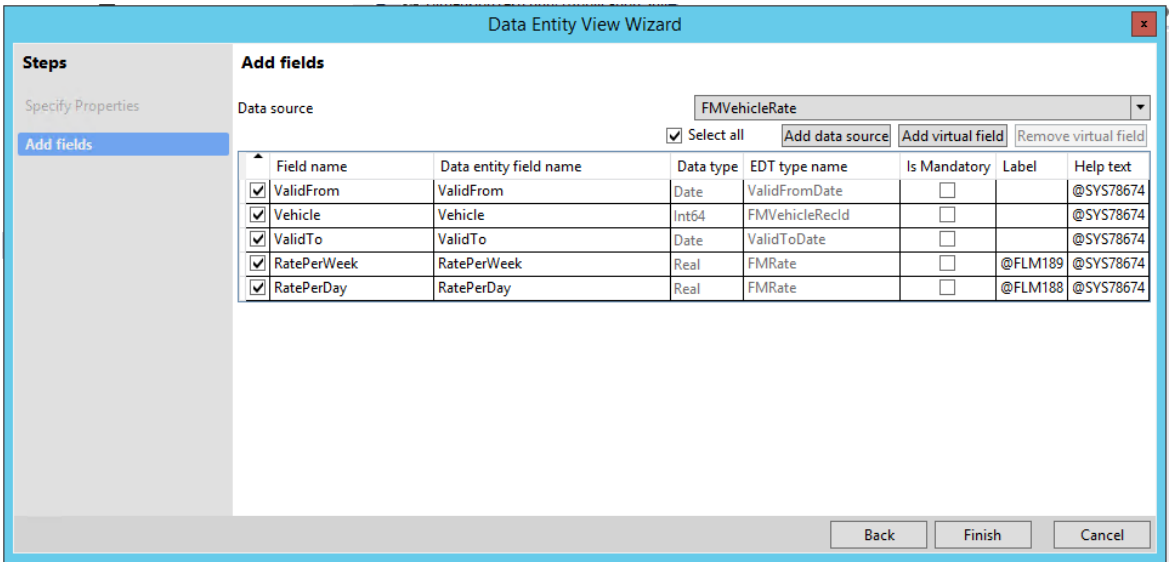
Create a new entity that is named **FMVehicleRateEntity**, and add it to the project.

1. In the left pane, select **Microsoft Dynamics 365 Artifacts**, and then click **Data Entity** in the left column of the main pane.
2. Click **Add**. The **Data Entity View** wizard starts.
3. Specify the property values for the data entity that you are creating, as shown in the following screen shot. The most important field is **Primary data source**, where you select **FMVehicleRate**.

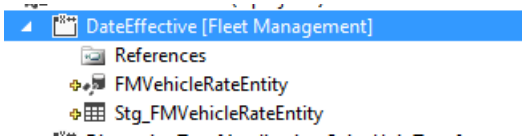


Click **Next**.

4. Add fields to the entity from the primary data source, **FMVehicleRate**.
5. Select all fields, and then click **Finish**.



The items are added to the project in Solution Explorer.

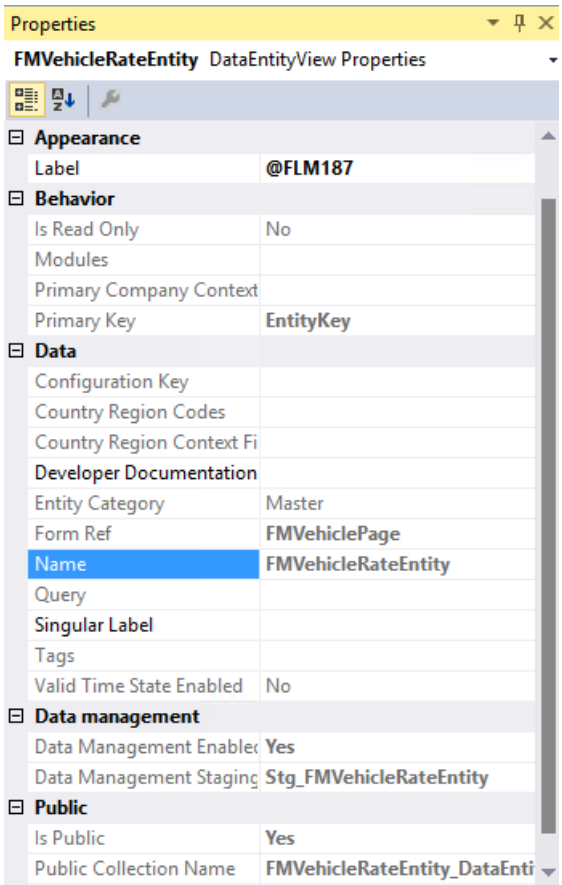


Build your project

1. Click **Build** > **Build Solution** to build your project.
2. Verify that the build has no errors. Warnings should be tolerated at this stage in the process.

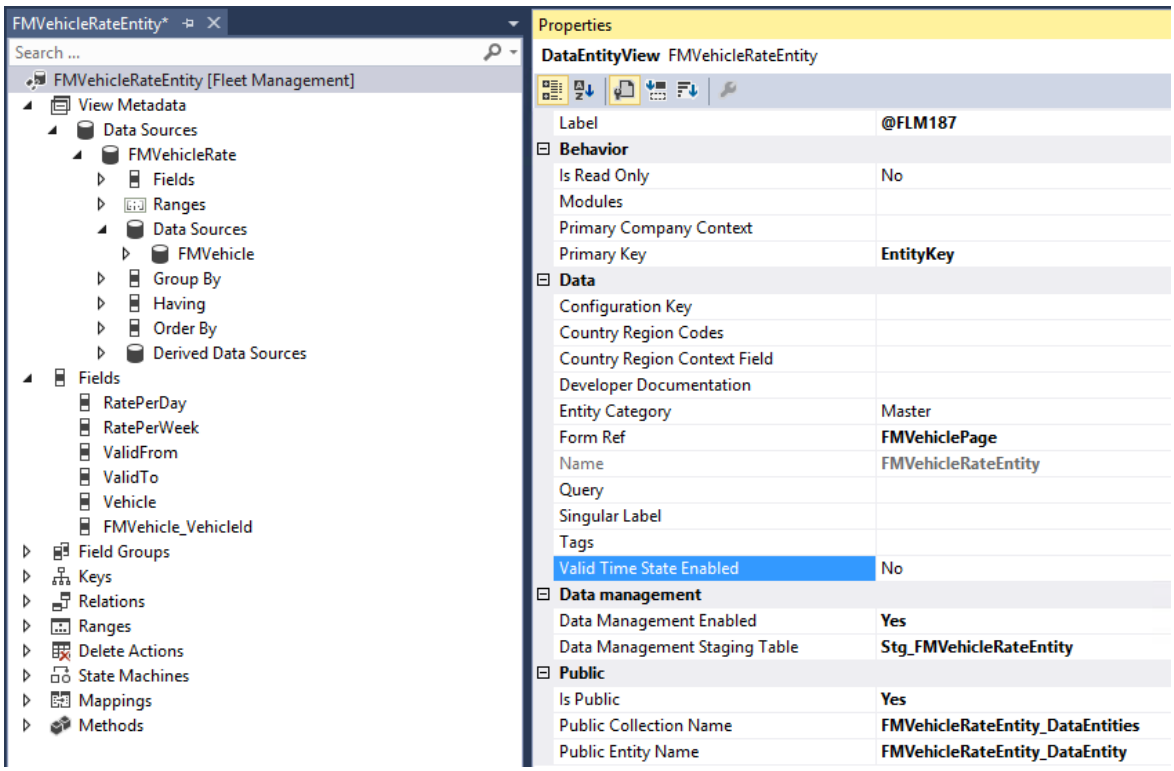
Validate the property values

- In Solution Explorer, select the **FMVehicleRateEntity** node, and validate the properties of the **FMVehicleRateEntity** entity by comparing them to the values in the **Properties** pane.

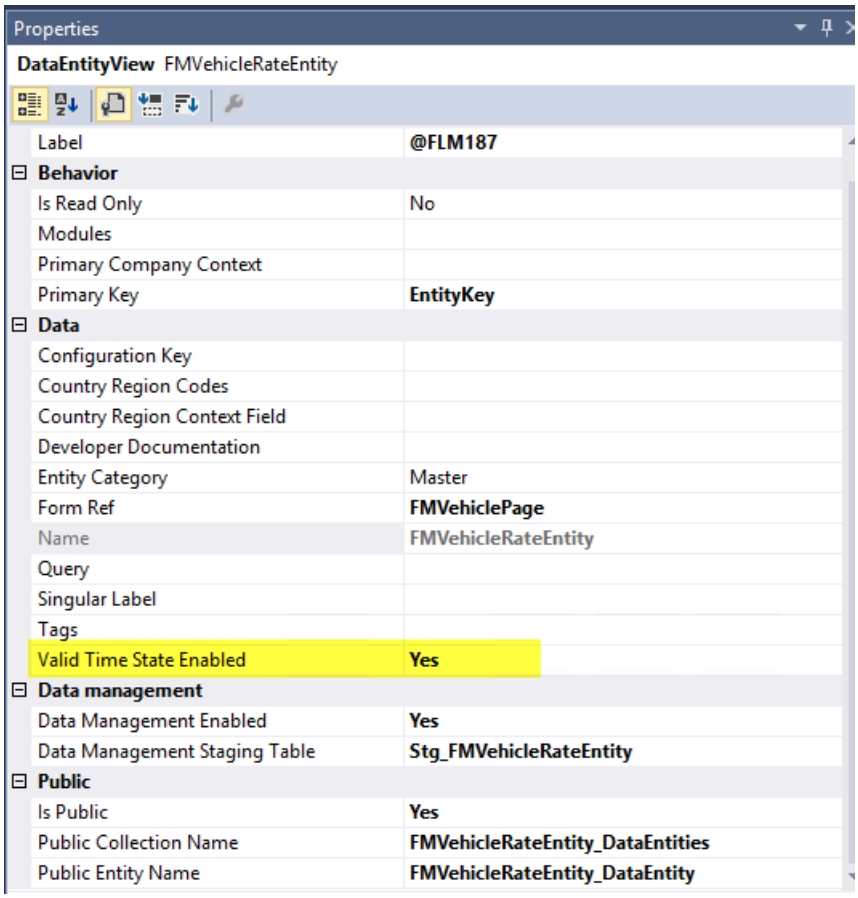


Make your entity date effective

1. In Solution Explorer, right-click the **FMVehicleRateEntity** node, and then click **Open**. The designer for the entity opens in the middle pane.

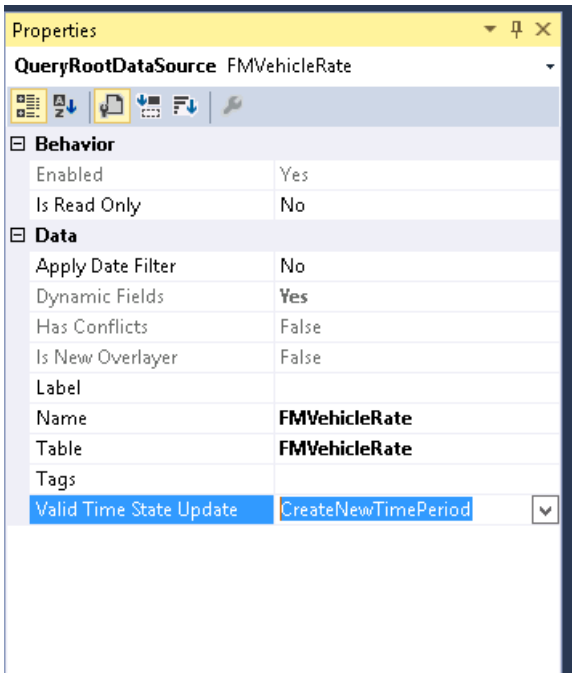


2. Change the value of the **Validate Time State Enabled** property to **Yes**.



Configure the Valid Time State Update property for the date-effective data source

- Select the **FMVehicleRate** data source, and then set the **Valid Time State Update** property to **CreateNewTimePeriod**.



Test your project

- Build your project again, and run the following X++ code to test your project.

```
/// <summary>
    /// Runs the class with the specified arguments.
    /// </summary>
    /// <param name = "_args">The specified arguments.</param>
    public static void main(Args _args)
    {
        FMVehicleRateEntity FMVehicleRateEntity;
        FMCarClass          vehicle;
        FMVehicleModel      model;
        FMVehicleRate       vehicleRateTable;
        TransDate           d1=1\1\1999,d2=31\12\2014;

        ttsbegin;

        select count(RecId) from FMVehicleRateEntity;
        info(strfmt("Entity - Valid today before insert %1",FMVehicleRateEntity.RecId));

        select count(RecId) from vehicleRateTable;
        info(strfmt("Table - Valid today before insert %1",vehicleRateTable.RecId));

        select firstonly model;

        vehicle.VehicleModel = model.RecId;
        vehicle.VehicleId = "TestV1001";
        vehicle.insert();

            if (vehicle)
            {
                FMVehicleRateEntity.clear();
                FMVehicleRateEntity.FMVehicle_VehicleId = vehicle.VehicleId;
                FMVehicleRateEntity.ValidFrom = d1;
                FMVehicleRateEntity.ValidTo = d2;
                FMVehicleRateEntity.RatePerDay = 100;
                FMVehicleRateEntity.RatePerWeek = 600;
                FMVehicleRateEntity.insert();

                // Should increase by one as compared to before insert numbers
                select count(RecId) from FMVehicleRateEntity;
                info(strfmt("Entity - Valid today after insert %1",FMVehicleRateEntity.RecId));

                // Should increase by one as compared to before insert numbers
```

```
            select count(RecId) from vehicleRateTable;
            info(strfmt("Table - Valid today after insert %1",vehicleRateTable.RecId));

            // New record should show in count
            select validtimestate(d1) count(RecId) from FMVehicleRateEntity;
            info(strfmt("Entity - Valid 1999 %1",FMVehicleRateEntity.RecId));

            // New record should show in count
            select validtimestate(d1) count(RecId) from vehicleRateTable;
            info(strfmt("Table - Valid 1999 %1",vehicleRateTable.RecId));

            // update newly created record
            // This should split record into two - 2009 to Today, today to 2014
            // Split happens because of mode in saveEntityDatasource
            select forupdate validtimestate(d1,d2) FMVehicleRateEntity
                 where FMVehicleRateEntity.FMVehicle_VehicleId == vehicle.VehicleId &&
                       FMVehicleRateEntity.ValidFrom == d1 &&
                       FMVehicleRateEntity.ValidTo == d2;
            FMVehicleRateEntity.RatePerDay = 200;
            FMVehicleRateEntity.update();

            // validate the split
            while select validtimestate(d1,d2) FMVehicleRateEntity
                 where FMVehicleRateEntity.FMVehicle_VehicleId == vehicle.VehicleId
            {
                info(strfmt("Entity - %1 to %2 , RatePerDay-%3, RatePerWeek-%4",
                       FMVehicleRateEntity.ValidFrom,
                       FMVehicleRateEntity.ValidTo,
                       FMVehicleRateEntity.RatePerDay,
                       FMVehicleRateEntity.RatePerWeek));
            }
            ttsabort;
        }
    }
```

Independent software vendor (ISV) licensing

2/18/2021 • 11 minutes to read • Edit Online

This topic describes the independent software vendor (ISV) licensing feature. It includes information about benefits and capabilities of the ISV licensing feature, and explains how to enable licensing for an ISV solution, create a package and generate a customer-specific license, and create self-signed certificates for test purposes.

The Microsoft Dynamics ecosystem provides tools and frameworks that let independent software vendors (ISVs) build, deploy, sell, and therefore monetize vertical industry solutions that can be repackaged. The ISV licensing feature provides the following benefits:

- It provides a safer licensing mechanism for ISV solutions for customers and partners. ISV solutions are enabled only if the customer has purchased a valid license key from the ISV.
- It aligns how customers handle licenses for ISV solutions from different ISVs, and therefore lowers the total cost of ownership (TCO).
- ISVs can independently generate, manage, and distribute ISV licenses by using industry standard frameworks.

This feature doesn't enable ISV competitor copycat protection (that is, source-based protection).

Capabilities

This section describes various capabilities of the ISV licensing feature.

ISVs can generate their own licenses

ISVs can independently generate their own licenses, apply them to solutions, and deliver those solutions to partners and customers. Each ISV license enables run-time features that help protect the ISV solution. Additionally, each ISV license is tied to an ISV Authenticode certificate, which ensures that the software was distributed by the ISV.

A run-time check makes sure that an ISV-generated license key exists in the customer's environment

Each ISV solution that is tied to a license runs only when a valid license key exists in the customer's environment. Therefore, if an ISV ties its solution to a license, but the customer doesn't have a valid license key, the solution doesn't run.

There are two types of license: Boolean and Number

ISVs can create two types of license: **Boolean** and **Number**. ISVs can associate an expiration date with either type of license. This expiration date is applied only to the ISV licenses and is independent of the system expiration date. A Boolean license is a simple activation license. The type of license (**Boolean** or **Number**) is set through a property in the license code node. ISVs can write their own custom logic to check the count that is provided in the ISV license, to make sure that their solutions are being used within the license terms. For more information, see Licensing Framework for ISVs.

License validation errors

When an ISV license becomes invalid after import, the ISV solution continues to run until the server is restarted. (After the server is restarted, the solution is disabled.) An error is thrown when the instance of the Application Object Server (AOS) starts. The error is written to the event log.

Implementing ISV licensing in a solution

ISVs must have a valid Authenticode certificate (X.509) from a certificate authority (CA). Microsoft doesn't

recommend any particular CA. However, many companies offer these certificates. Authenticode certificates come in various key sizes. The ISV licensing feature supports certificates of both 1024-bit and 2048-bit key sizes. By default, many providers use the 2048-bit key size, and we recommend that ISVs use this bit key size, because it provides stronger encryption. However, if an ISV already has an existing 1024-bit key size, that key size works with the ISV licensing feature.

> **NOTE**
>
> The ISV licensing feature doesn't support 4096-bit key sizes. Authenticode certificates can have various cryptographic service providers. The ISV licensing feature uses Enhanced Cryptographic Provider (which also covers Base Cryptographic Provider). There are many independent providers that you can purchase an Authenticode certificate from. Microsoft doesn't recommend any particular provider. Some providers that are often used are Symantec VeriSign, Thawte, and Go Daddy.

Certificate import and export

The certificate is used to sign your customer license files and validate the license files at the time of import. Authenticode certificates support four file formats. For the ISV licensing feature, you must have the certificate files in two formats:

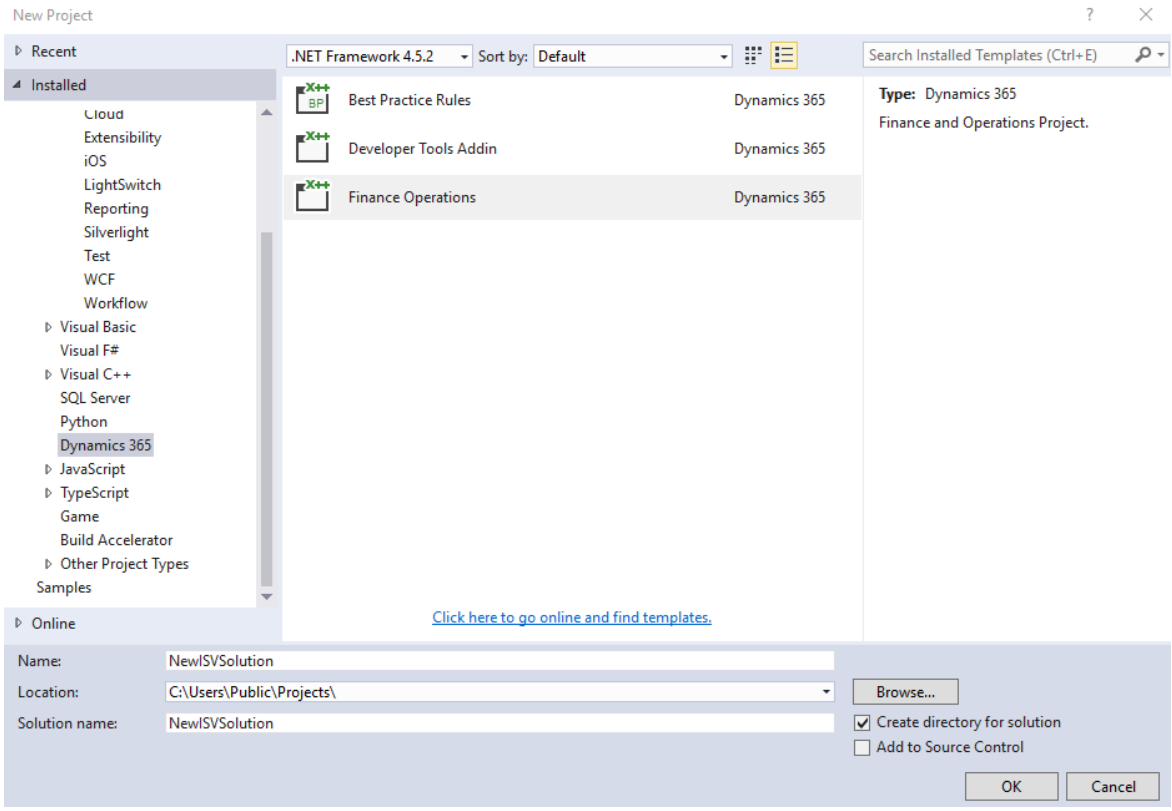
- **Personal Information Exchange (PFX, also known as PKCS #12)** – The PKCS #12 format, which uses the .pfx file name extension, supports secure storage of certificates, private keys, and all certificates in a certification path. The PKCS #12 format is the only file format that can be used to export a certificate and its private key.
- **Base64-encoded X.509** – The Base64 format supports storage of a single certificate. This format doesn't support storage of the private key or certification path.

There is a restriction on the format. The PFX (PKCS #12) format should be used only to export the certificate together with its private key for signing/generating purposes. It should never be shared outside the ISV organization. The DER-encoded binary X.509 format, which uses the .cer file name extension, should be used to export the public key of the certificate that must be embedded in the Application Object Tree (AOT) License. This public key is distributed to customers via the model. It's used when a license is imported, to make sure that the license is signed by the ISV license that owns the private key.

Enable licensing for your ISV solution

Follow these steps to enable licensing for your solution.

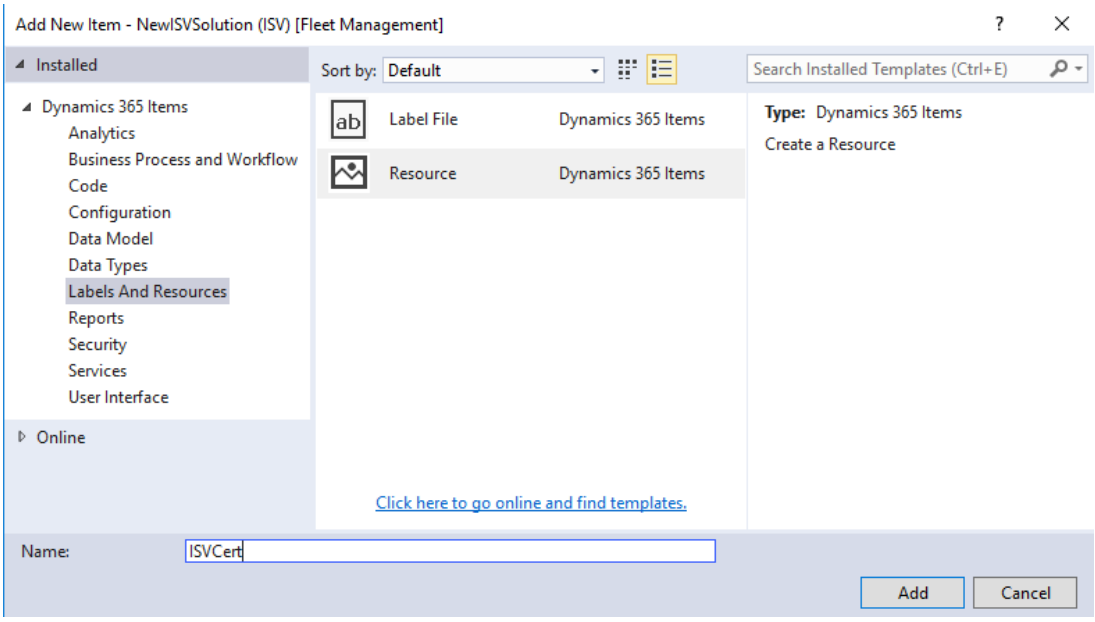
1. Create an ISV solution. In Visual Studio, click **File > New project**. In the **New Project** dialog, click **Installed > Templates > Dynamics 365**. Create a **Finance Operations** project. In this example, we named the project **NewISVSolution**.



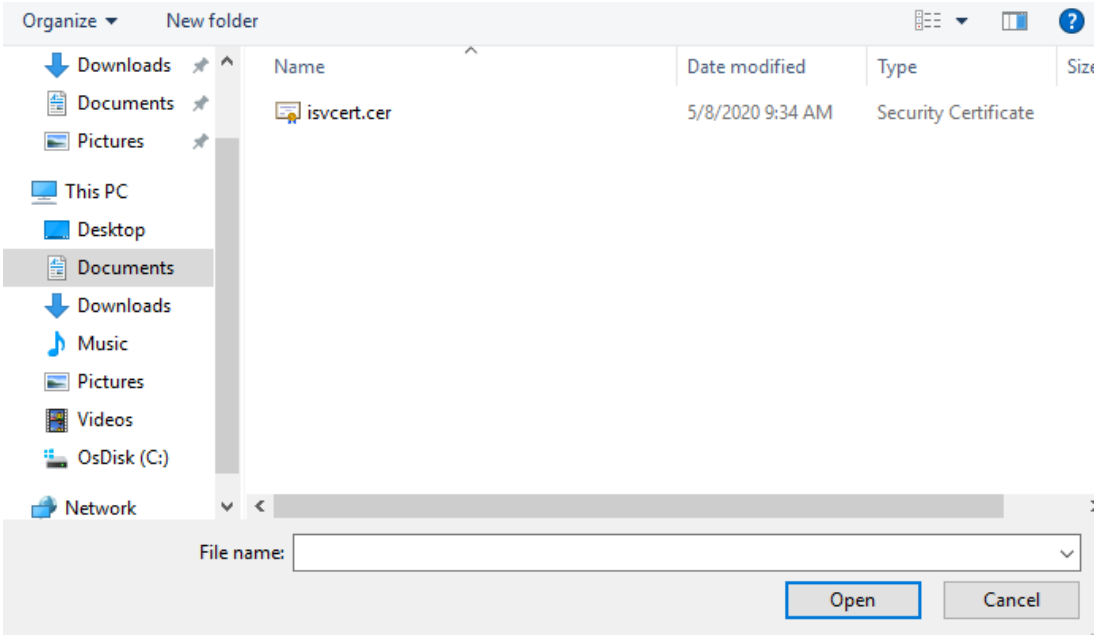
2. Add the certificate's public key (.cer file) to your project as a resource. To create a certificate for testing, see Appendix: Create self-signed certificates for test purposes.

   a. Right-click the project in Solution Explorer, then click **Add > New item**.

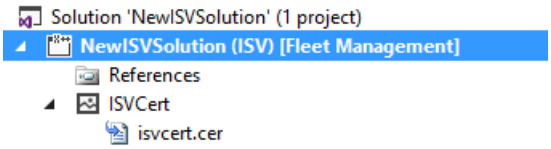
   b. Under **Installed > Dynamics 365 Items**, click **Labels And Resources**, and then select **Resource**. Name the resource. In this example, we named the resource **ISVCert**.



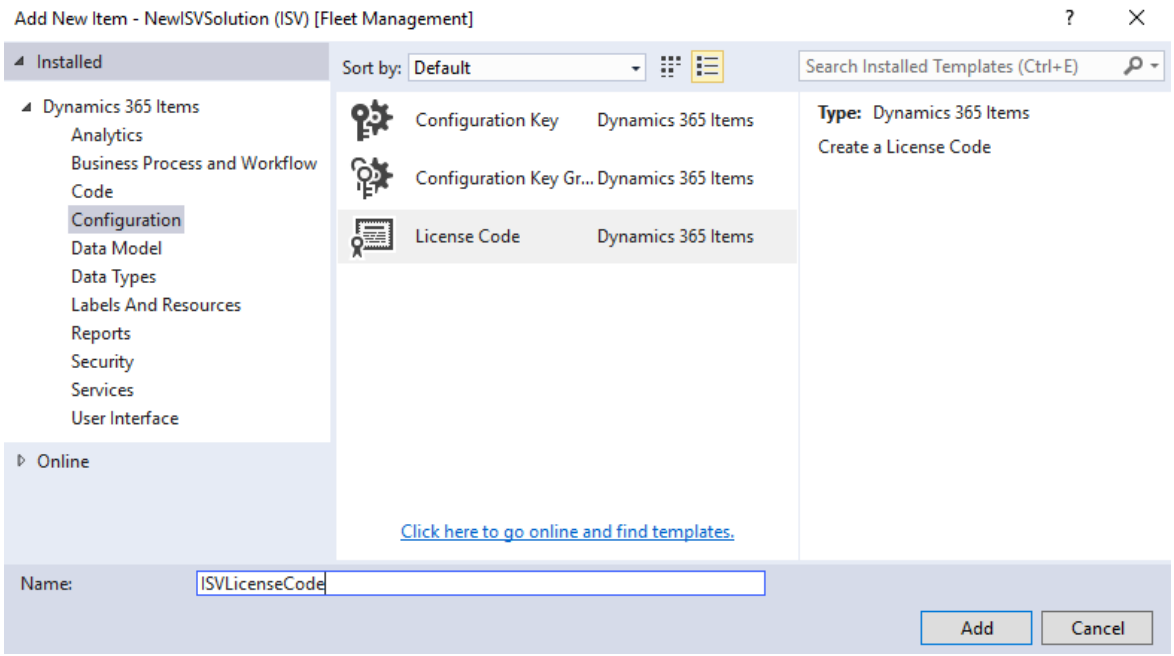
   c. Click **Add** and select the certificate's public key file (.cer file).



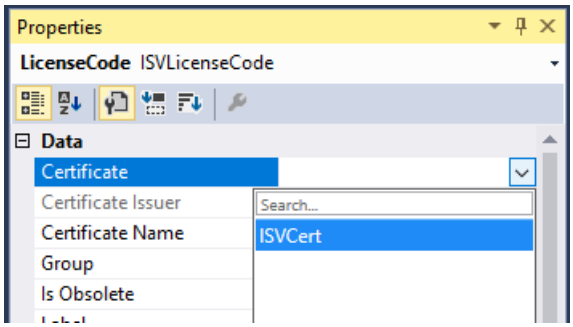
d. Click **Open** to add the certificate.



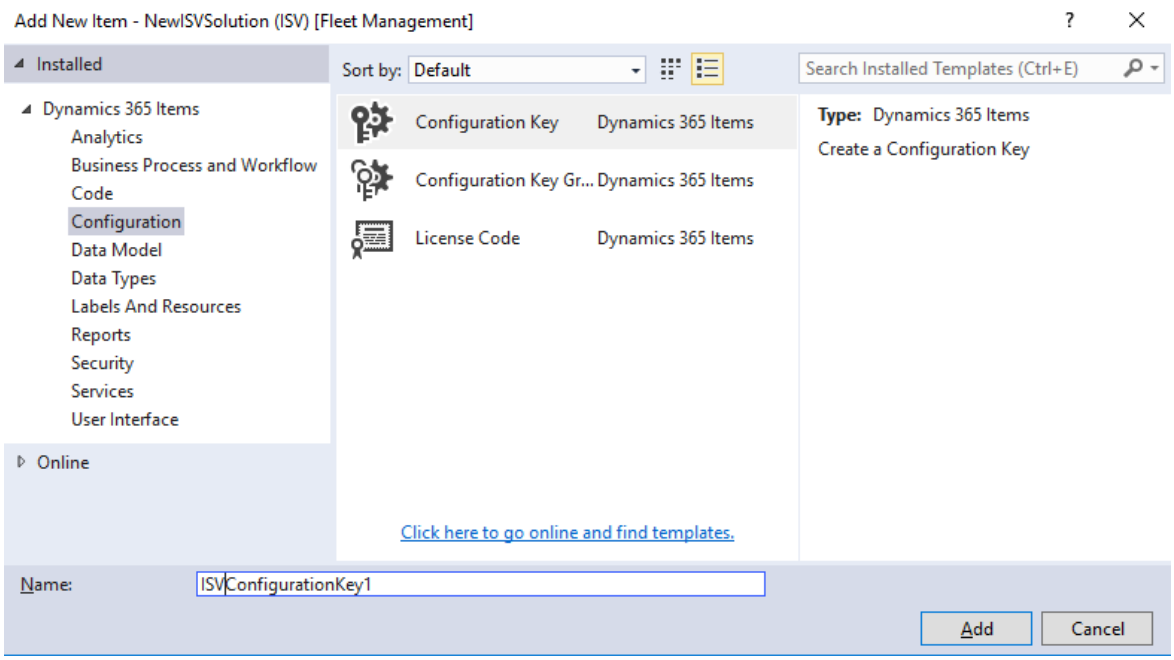
3. Create a license code. Right-click the project in Solution Explorer, then click **Add > New item**. Under **Installed > Dynamics 365 Items**, choose **Configuration**. In the list, choose **License Code** and name the license code. In this example, we named the license code **ISVLicenseCode**. Click **Add**.



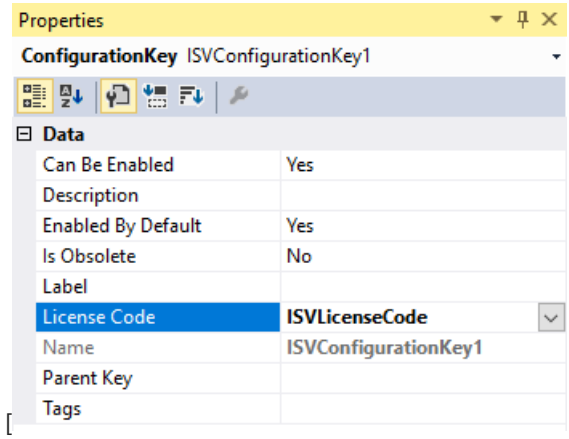
4. Map the certificate to the license code. In the Properties window for the license code, set the **Certificate** property to your certificate resource. In this example, we set **Certificate** to **ISVCert**.



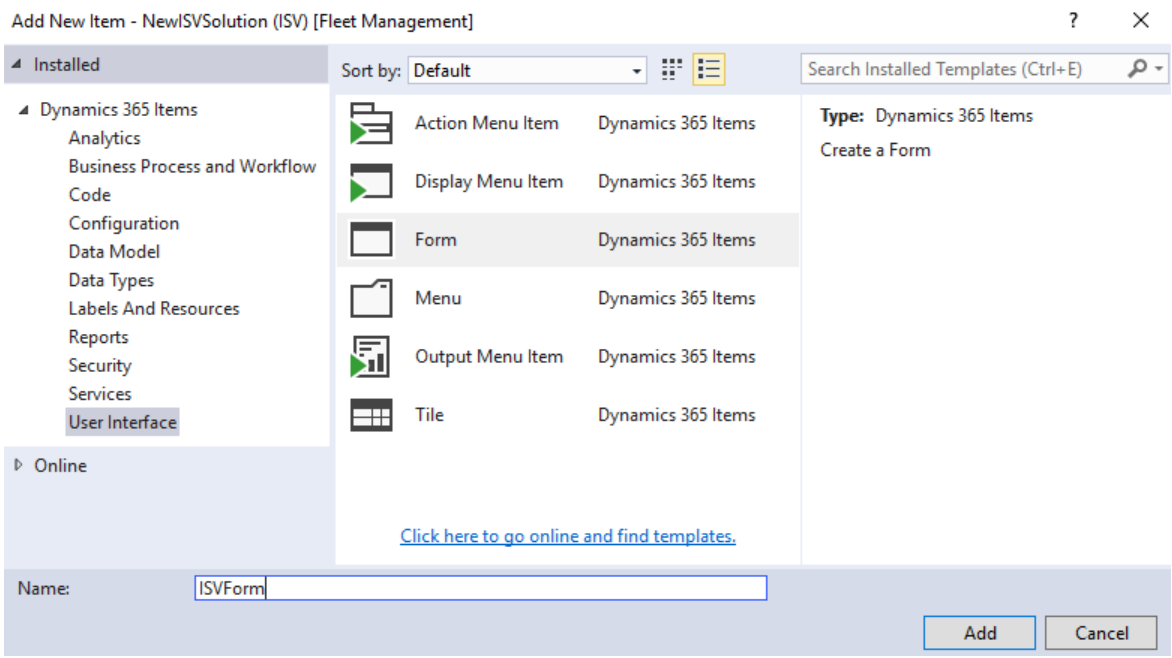
5. Create one or more configuration keys. Right-click the project in Solution Explorer, then click **Add > New item**. Under **Installed > Dynamics 365 Items**, choose **Configuration**. In the list, choose **Configuration Key**. Name the key and click **Add**. In this example, we named the configuration key **ISVConfigurationKey1**.



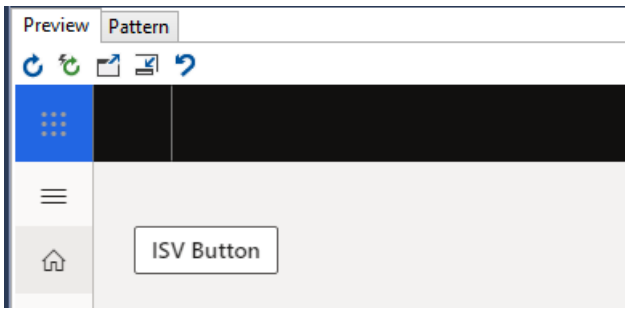
6. Associate the license code with the configuration key. In Solution Explorer, double-click the configuration key to open the Properties window. In the Properties window, set the **LicenseCode** property to your license code. In this example, we set the **LicenseCode** to **ISVLicenseCode**.



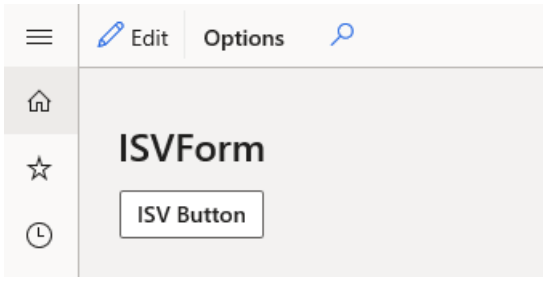
7. Associate a configuration key to an element in your solution. For example, create a new form. Right-click the project in Solution Explorer, then click **Add > New item**. Under **Installed > Dynamics 365 Items**, choose **User Interface**. In the list, choose **Form** and give it a name. In this example, we named the form **ISVForm**.



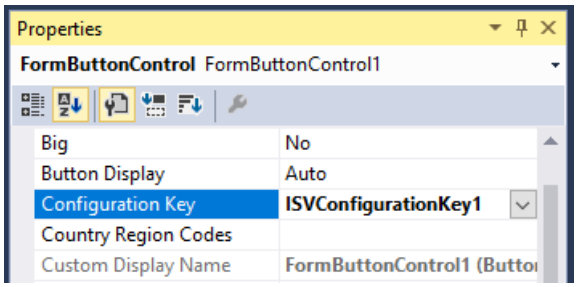
8. Add a button to the form. Double-click the form in the Solution Explorer. In the Design window, right-click and select **New**, and then **Button**. Set the **Text** property to **ISVButton**.



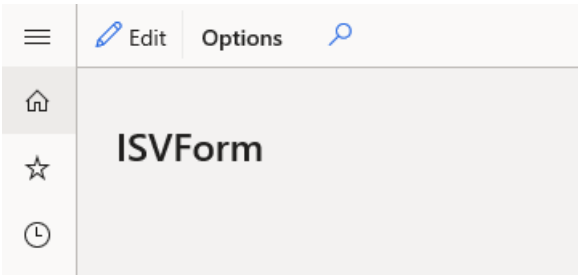
At runtime, the button is visible because it isn't controlled by a configuration key at first.



9. Associate a configuration key with the button. In the Properties window for the button, set the **Configuration Key** property to your configuration. In this example, we set the **Configuration Key** to **ISVConfigurationKey1**.

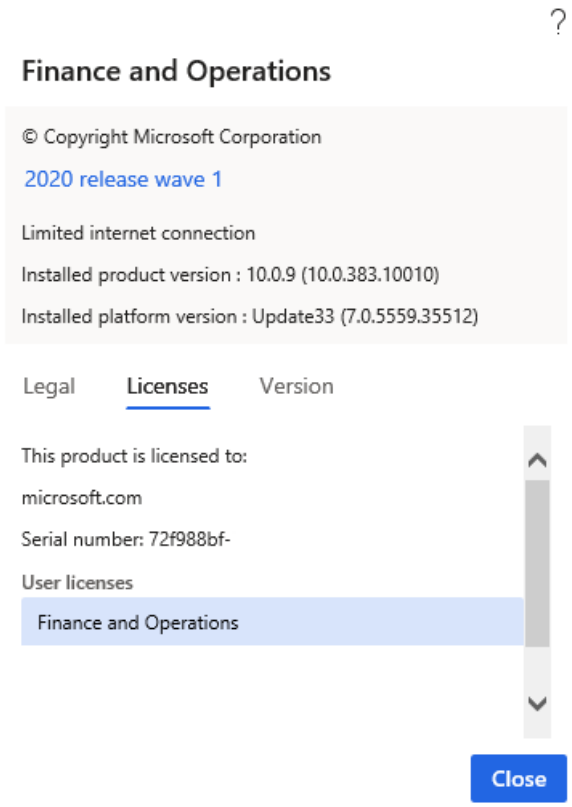


At runtime, the button is not visible because the configuration key must be available and enabled.



Create a package and generate a customer-specific license

1. Collect the tenant name and ID for the customer to issue the license to. You can find this information at **Settings > Help & Support > About** on the **Licenses** tab.



2. Generate a license for the customer (tenant ID and name), and sign the license by using the certificate's private key. You must pass the following parameters to the **axutil genlicense** command to create the license file.

PARAMETER NAME	DESCRIPTION
file	The name of your license file.
licensecode	The name of your license code (from Microsoft Visual Studio).
certificatepath	The path of your certificate's private key.
password	The password for your certificate's private key.
customer	The customer's tenant name (from the screenshot under step 1).

PARAMETER NAME	DESCRIPTION
serialnumber	The customer's tenant ID (labeled "Serial number" in the screenshot).
expirationdate	Optional: The expiration date for the license.
usercount	Optional: The number that custom validation logic can use as required. This could be users, but is not limited to users.

Here is an example.

```
C:\AOSService\PackagesLocalDirectory\Bin\axutil genlicense /file:c:\templicense.txt
/certificatepath:c:\tempisvcert.pfx /licensecode:ISVLicenseCode /customer:TAEOfficial.ccsctp.net
/serialnumber:4dbfcf74-c5a6-4727-b638-d56e51d1f381 /password:********
```

3. Import the license into the target environment.

> **NOTE**
> In production systems, you complete this step from Microsoft Dynamics Lifecycle Services (LCS), by using a deployable package. For more information, see the "Production environments" section later in this topic.

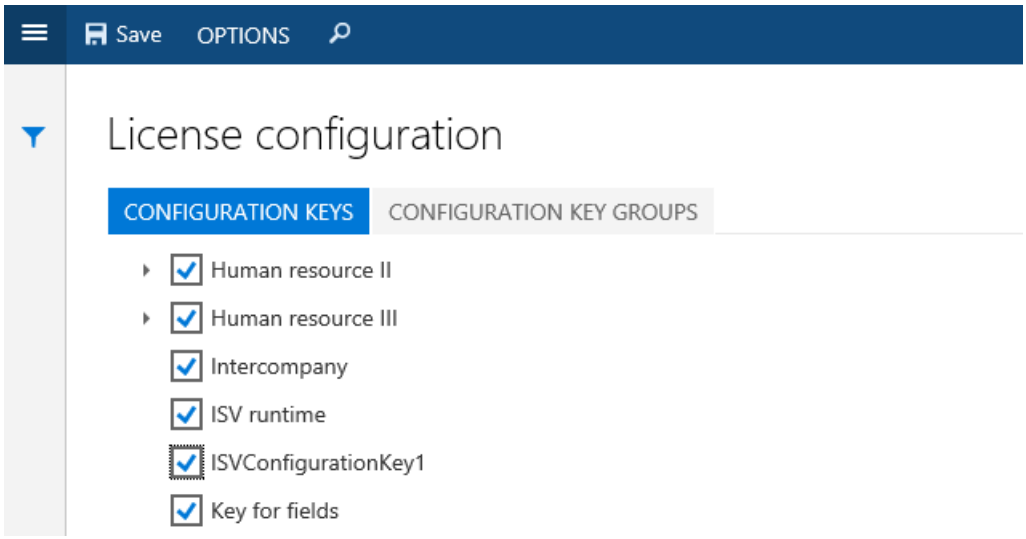
PARAMETER NAME	DESCRIPTION
--setupmode importlicensefile	Use this parameter to inform the setup tool that a license will be loaded.
--metadatadir	Use this parameter to specify the metadata directory. You should use the default packages directory.
--bindir	Use this parameter to specify the binaries directory. You should use the default packages directory.
--sqlserver	Use this parameter to specify the Microsoft SQL Server. For one-box environment, use a period (.).
--sqluser	Use this parameter to specify the SQL Server user. You should pass in **AOSUser**.
--sqlpwd	Use this parameter to specify the SQL Server password.
--licensefilename	Use this parameter to specify the license file that will be loaded.

Here is an example.

```
C:\AOSService\PackagesLocalDirectory\Bin\Microsoft.Dynamics.AX.Deployment.Setup.exe --setupmode
importlicensefile --metadatadir c:\packages --bindir c:\packages --sqlserver . --sqldatabase axdbrain
--sqluser AOSUser --sqlpwd ******** --licensefilename c:\templicense.txt
```

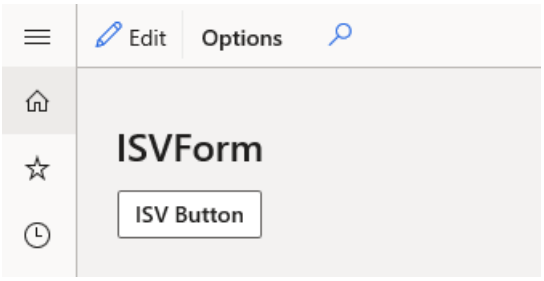
4. The corresponding configuration key will be available and enabled on the **License configuration** page. By default, the configuration is enabled. For example, see the **ISVConfigurationKey1** configuration key

in the following screenshot.



5. In non-production installations, you must start the database synchronization process from Visual Studio.

After the configuration key is enabled, the button becomes visible, as shown in the following screenshot.



Protection best practices

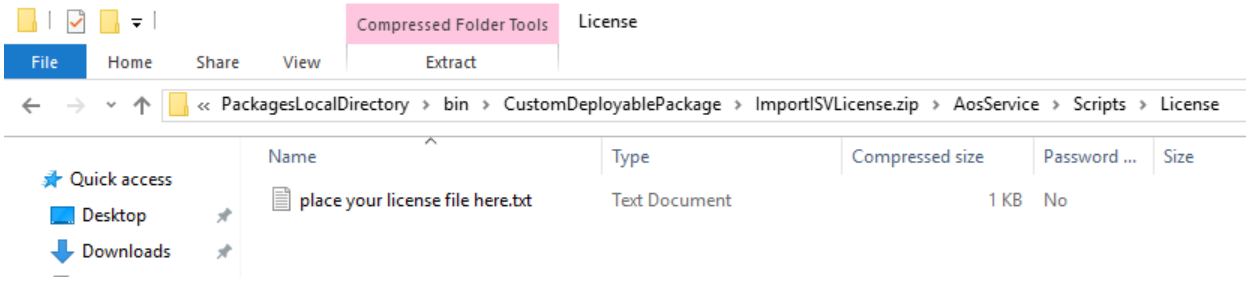
Solutions can be delivered in two forms:

- Model files (source code)
- Deployable packages (binary)

To protect your configuration keys and license codes, we recommend that you release them in binary form, by using a deployable package. Customers will then be able to install and interact with those elements in Visual Studio. Although customers will be able to refer to items in the deployable package, they won't be able to access source code or make modifications to the items. (However, they can create extensions.) More details about the capability to release solutions in binary form will be available soon. The deployable package (binary) can also include classes and other logic that your customer doesn't require access to and should not be able to customize.

Production environments

To install ISV licenses in production systems, you must use a deployable package through LCS. You can find a template package for configuration mode at the following location in all installations: <PackagesFolder>\bin\CustomDeployablePackage\ImportISVLicense.zip (Packages folder is typically under j:\AOSService\PackagesLocalDirectory or c:\AOSService\PackagesLocalDirectory\)



1. Make a copy of the package template.
2. Put the license file in the following folder within the package template: ImportISVLicense.zip\AosService\Scripts\License

More than one license can be installed at a time. If one of the licenses depends on another, make sure that it's named accordingly. (Licenses are installed in alphabetical order.)

Appendix: Create self-signed certificates for test purposes

> **NOTE**
> Self-signed certificates can be used only during development. They aren't supported in production environments.

For Platform update 34 and earlier: (Deprecated - uses SHA1 hash algorithm for license creation)

1. For test purposes, create a self-signed CA certificate. Use the Visual Studio tools prompt to run the following command.

   ```
   makecert -r -pe -n "CN=IsvCertTestAuthority O=IsvCertTestAuthority" -ss CA -sr LocalMachine -a sha256
   -len 2048 -cy authority -sky signature -b 01/01/2016 -sv c:\temp\CA.pvk c:\temp\CA.cer
   ```

   For more information, see the MakeCert documentation.

2. Create a certificate by using the CA.

   ```
   makecert -pe -n "CN=IsvCertTest O=IsvCertTest" -ss ISVStore -sr LocalMachine -a sha256 -len 2048 -cy
   end -sky signature -eku 1.3.6.1.5.5.7.3.3 -ic c:\temp\ca.cer -iv c:\temp\ca.pvk -b **/**/**** -sv
   c:\temp\isvcert.pvk c:\temp\isvcert.cer
   ```

3. Convert the ISV certificate to PFX format.

   ```
   pvk2pfx -pvk c:\temp\isvcert.pvk -spc c:\temp\isvcert.cer -pfx c:\temp\isvcert.pfx -po ********
   ```

4. For a test scenario, import the self-signed CA certificate manually on all the AOS instances.

   ```
   certutil -addstore root c:\temp\ca.cer
   ```

   However, if a self-signed ISV certificate was used, that certificate must be imported instead of the CA certificate.

   ```
   certutil -addstore root c:\temp\isvcert.cer
   ```

For Platform update 35 and later: (Uses SHA256 hash algorithm for license creation)

1. For test purposes, create a self-signed certificate using the PowerShell command `New-SelfSignedCertificate`:

   a. Create the certificate. (Note: adjust start and end dates accordingly.)

   ```
   $cert = New-SelfSignedCertificate -CertStoreLocation Cert:\LocalMachine\My -DnsName "IsvCert"
   -Type CodeSigningCert -KeyExportPolicy Exportable -HashAlgorithm sha256 -KeyLength 2048 -
   KeySpec Signature -Provider "Microsoft Enhanced RSA and AES Cryptographic Provider" -NotBefore
   (Get-Date -Year 2020 -Month 1 -Day 1) -NotAfter (Get-Date -Year 2022 -Month 12 -Day 31)
   ```

   b. Get a reference to the new certificate.

```
[String]$certPath = Join-Path -Path "cert:\LocalMachine\My\" -ChildPath "$($cert.Thumbprint)"
```

c. Create the secure string password that the certificate uses. (Replace "###############" with the certificate password)

```
[System.Security.SecureString]$certPassword = ConvertTo-SecureString -String "###############"
-Force -AsPlainText
```

d. Export the certificate private key as **.pfx** file using the password.

```
Export-PfxCertificate -Cert $certPath -FilePath "C:\Temp\IsvCert.pfx" -Password $certPassword
```

e. Export the certificate public key as a **.cer** file.

```
Export-Certificate -Cert $certPath -FilePath "C:\Temp\IsvCert.cer"
```

2. Add the certificate to the root store.

```
certutil -addstore root C:\Temp\IsvCert.cer
```

Independent software vendor (ISV) licensing (on-premises)

2/18/2021 • 4 minutes to read • Edit Online

This topic explains how to import independent software vendor (ISV) licenses into an on-premises deployment.

> **IMPORTANT**
> The process that is described in this topic is available only for customers who have on-premises environments that are deployed with Platform update 12 or later.

For general information about the benefits of ISV licensing, information about how to enable licensing for your solution, and other information that is related to self-signed certificates, see Independent software vendor (ISV) licensing.

Prerequisites

Before you import the ISV license file into your on-premises environment, verify that the following prerequisites are met:

- The most recent version of the local agent was used when the environment was deployed.
- The environment is deployed with Platform update 12, and all hotfixes for Platform update 12 are applied. This step is mandatory because Microsoft has released a fix for an ISV licensing scenario. To get the latest set of hotfixes, use the tiles on the **Environment details** page in Microsoft Dynamics Lifecycle Services (LCS).
- Before you import the ISV license, the environment must be deployed, and the Application Object Server (AOS) service must be running.
- Before you import the ISV license, the ISV solution must be applied to the on-premises environment. The ISV solution can be applied to an on-premises environment during the deployment flow. Alternatively, you can use the **Apply updates** flow in LCS to apply the ISV solution as a post-deployment step. If the ISV solution isn't applied before you import the license, the customizations won't be enabled.

Import licenses

The following procedure can be used for a sandbox environment or a production environment that is deployed in an on-premises project.

> **NOTE**
> Because import of an ISV license requires downtime, no business transactions can be performed in the environment during import. When you complete the import, make sure that no one is using the system, and that an official downtime notice has been communicated to all the users.

1. Collect the tenant name and ID for the customer to issue the license to:
    a. Connect to the instance of Service Fabric Explorer where the environment is hosted.
    b. Go to **Clusters** > **Applications** > **AXSFType** > **fabric:\AXSF**, and then, on the right page, select the **Details** tab.
    c. In the **Parameters** table, find the values for the **License_TenantDomainGuid** and

Licence_TenantId keys.

2. Generate a license for the customer (tenant ID and name), and sign the license by using the certificate's private key. The following parameters must be passed to theAXUtil **genlicense**command to create the license file. The command will generate an XML file.

PARAMETER NAME	DESCRIPTION
file	The name of the license file.
licensecode	The name of the license code from Microsoft Visual Studio.
certificatepath	The path of the certificate's private key.
password	The password of the certificate's private key.
customer	The customer's tenant name.
serialnumber	The customer's tenant ID.
expirationdate	Optional: The expiration date of the license.
usercount	Optional: The number that can be required for the custom validation logic. This number can be the number of users, but it isn't limited to users.

Here is an example of the command.

```
C:\AOSService\PackagesLocalDirectory\Bin\axutil genlicense /file:c:\templicense.txt
/certificatepath:c:\tempisvcert.pfx /licensecode:ISVLicenseCode /customer:TAEOfficial.ccsctp.net
/serialnumber:4dbfcf74-c5a6-4727-b638-d56e51d1f381 /password:********
```

3. Copy the licenses that are generated to a folder on one of the machines that is running fabric:/AXSF, and verify that fabric:/AXSF is healthy.

4. Run the **Import-LicensePackage.ps1** script from one of the AOS machines. You can find this script in the latest **Deployment scripts** folder on the **Model** tab in the Shared asset library in LCS. Here is a list of the parameters that you must pass to the script:

   - **LicenseFilesPath** – The path of a folder that contains the license files that must be imported.
   - **SqlUser** – The same user who is specified in the credentials.json file to run the AOS.
   - **SqlPassword** – The password that can be used to connect to SQL.
   - **EnvironmentConfigPath** – The configuration file for the environment. This file is named config.json and is located under the agent share in a folder that has the format wp\<environment-name>\StandaloneSetup.

     After the command is run, log files are generated for each license file that is processed. The names of the log files are in the format {license_file_name}.output.log and {license_file_name}.error.log. The logs that are generated during database synchronization are located in files that are structured like dbsync.output.log and dbsync.error.log.

5. When the script has been run successfully, validate that the configuration key has been imported and enabled. In the product, the corresponding configuration key will be available and enabled on the**License**

configuration page. By default, the configuration is enabled. For example, if you added a configuration key that is named ISVConfigurationKey1, it will appear in the list of configuration keys.

When the configuration key is enabled, the changes in the ISV solution will be visible in the product.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

General Data Protection Regulation overview

2/18/2021 • 19 minutes to read • Edit Online

Overview of the GDPR

The European Union's General Data Protection Regulation (GDPR) sets a new global standard for privacy rights, security, and compliance for the citizens and residents of the European Union (EU). The GDPR governs the handling and use of personal data of EU citizens and residents. Enforcement of the GDPR begins May 25, 2018, and there are significant consequences for non-compliance. For more information about the regulation, see the European Union site.

> **NOTE**
>
> For information about the scope and coverage of this documentation, see Clarification of the scope of this content section at the end of this topic.
>
> Before utilizing any product features in support of your GDPR compliance efforts, please ensure that you have applied all of the related hotfixes.

The GDPR gives EU citizens specific data subject rights (DSRs) that let them perform the following actions:

- View their personal data.
- Correct errors in their personal data.
- Erase their personal data.
- Object to processing of their personal data.
- Export their personal data.

The GDPR defines personal data in the following way in article 4 of the regulation (organizations do not have personal data):

> (1) 'personal data' means any information relating to an identified or identifiable natural person ('data subject'); an identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person;

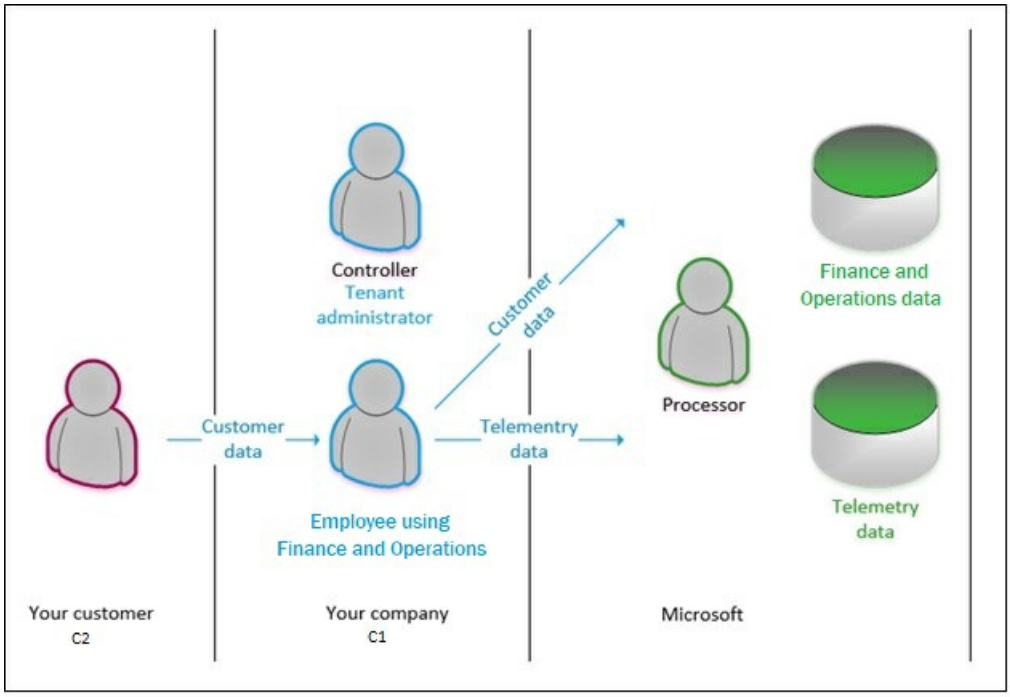
To determine responsibilities for compliance, the GDPR identifies the following roles:

- **Data controller** – The controller controls personal data and determines how it's used. The responsibilities of the controller include but are not limited to collecting, maintaining, directing actions, protecting, modifying and deleting personal data. The controller either adds users to the system, grants access to the system, and collects data from data subjects, or has employees who complete these tasks on the company's behalf. The burden of understanding the process for GDPR requests and carrying out a GDPR request rests with the controller.
- **Data processor** – The processor provides services to, and processes data on behalf of, the data controller. The processor performs actions on behalf of the controller. The processor makes it possible for the controller to be GDPR compliant, but has no ownership of the data and does not respond directly to DSR requests.
- **Data subject** – A data subject is a natural person whose personal information is being used.
- **C1** – C1 is a Microsoft direct customer (IT Admin in the Enterprise Cloud).

- **C2** – C2 is C1's customer.

For Finance and Operations apps, Microsoft acts as a processor. As a data processor, Finance and Operations provides processes and features that help you comply with your GDPR obligations as a data controller.

The following illustration shows the flow of data from your customer to the application database, and the roles that you and Microsoft play in that process. For each application, the controller is the tenant administrator, and Microsoft is the processor. In this scenario, the data is sent to the processor (Microsoft), who then processes the data by storing it, retrieving it, sorting it, and so on.



When a data subject chooses to submit a DSR, the data subject makes the request to the controller. Data subjects won't approach Microsoft to exercise their rights for data that your business has collected. As the processor, Microsoft assists the controller by providing features, or just by making sure that the actions are possible. In other words, the controller accepts and responds to a DSR request, and the processor assists with or enables the compliance request. The following table outlines some of the roles and responsibilities that are relevant.

Role	Scenarios	Implementation	Level of data access
Your customer (2)	• View personal data • Correct personal data • Erase personal data • Object to processing • Export personal data	You must provide a mechanism for your customer to exercise a DSR (process or service).	Your customer sees only their personal data.

Your employee – information worker	• View personal data • Correct personal data • Erase personal data • Object to processing • Export personal data	You must provide a mechanism for your worker to exercise a DSR (process or service). Some activity information may be obtained from Microsoft	Your information worker sees only their personal data.
Your employee – GDPR administrator	• Validates the user identity request • Locates the personal data across systems • Curates the data based on your policy • Creates a data package or executes an action	• Uses Finance and Operations to locate the data and fulfill the request. • Writes a customization. • Reaches out to third parties for shared-controller DSRs. • Reaches out to Microsoft for activity data.	Your GDRP administrator sees the data that has been obtained to fulfill the DSR request.

Responding to requests to view, correct, erase, object, or export personal data

Suppose that a customer decides that they want to understand what personal data of theirs is maintained by an organization. That customer approaches that organization and asks to exercise their DSR. When data subjects exercise their DSRs, controllers must address each of the following items specifically:

- Properly identify the person and role (is the person an employee, a customer, a vendor?) by using information that the data subject gave you as part of their request. This information might be a name, an employee ID or customer number, or another identifier.
- Record the date and time of the request. (You have 30 days to complete the request.)
- Affirm that the DSR request is proper and valid. You will need to work with your legal counsel to determine what is valid. For example, you must make sure that compliance with a DSR request doesn't conflict with any other legal obligations that you have.
- Verify that you have the information that is related to the request.

Reasons why certain personal data may not be modified or deleted

The following table lists several reasons why personal data modification or deletion is restricted in certain

scenarios.

REASON	COMMENT
Financial, tax, generally accepted accounting principles (GAAP)	A party can't be deleted, but the party's name can be updated.
Financial, tax, GAAP	A current worker's data can't be deleted, but the worker's name can be updated.
GAAP	Posted or completed transactions can't be modified.

Right to view

An organization might decide to take any of the following actions in response to a DSR request to view data:

- Use the Person search report to find and collect personal data. To access this report, from the navigation pane, select **Modules > System administration > Inquiries > Person search report**.
- Extend the Person search report by authoring a new entity or extending an existing entity.
- Use search and filter features to find specific personal data and export that data by using the Microsoft Office Export functionality or print that information to a .pdf using browser extensions.
- Use provided documentation to identify data tables that contain data that the controller has identified as personal data.
- Author a custom form that locates and exports personal data.
- Author an external portal or website that allows an authenticated customer to see their personal data.

The Person search report might help you discover personal data that is subject to a DSR request. If the report doesn't include the information that you're looking for, check the Microsoft Dynamics Lifecycle Services (LCS) site for possible hotfixes that include the information. You can also extend the report yourself by creating additional entities, or extending the provided entities.

If the Person search report doesn't contain all the information that the data subject is requesting, you can extend it by using tools that Microsoft has provided. For information about how to extend the Person search report, see Extend the Person search report.

**Right to correct* **

An organization might decide to take any of the following actions in response to a DSR request to correct data:

- Use the Person search report to find and collect personal data.
- Extend the Person search report by authoring a new entity or extending an existing entity.
- Use search and filter features to find specific personal data.
- Author a custom form that locates personal data.
- Author an external portal or website that allows an authenticated customer to correct their personal data.

When data is located, use in-product features to correct the data where the product offers the ability to do so.

*You might find that some data that qualifies as personal data can't be modified directly. Typically, this data is part of a financial transaction or other business data that is kept "as is" for compliance with financial laws (for example, tax laws), prevention of fraud (such as security audit trail), or compliance with industry certifications.

** GDPR is not a law exclusive of all other laws. As an enterprise resource planning system, Finance and Operations does not allow for modification of certain business or transactional data, and will not endorse nor provide functionality for the modification of business data that is necessary for compliance with other laws or certifications. Finance and Operations will not provide support for modifications/customizations or other actions that result in the corruption of referential or business data integrity.

Right to be forgotten*

An organization might decide to take any of the following actions in response to a DSR request to erase data:

- Delete or otherwise erase personal data where the product enables that action directly.
- Anonymize the personal data where the product enables that action directly.
- Author a customization to erase/modify the personal data.

* GDPR is not a law exclusive of all other laws. As an enterprise resource planning system, Finance and Operations does not allow for deletion of certain business or transactional data, and will not endorse nor provide functionality for the deletion of business data that is necessary for compliance with other laws or certifications. Finance and Operations will not provide support for modifications/customizations or other actions that result in the corruption of referential or business data integrity.

Right to port

An organization might decide to take any of the following actions in response to a DSR request to port data:

- Use the Microsoft Office Add-in to export personal data.
- Author a custom report that enables the export of personal data.
- Author a customization that exports personal data.
- Use or extend the Person search report to gather information in support of a request for a copy of the data subject's personal information.

The Person search report might help you discover personal data that is subject to a DSR request. If the report doesn't include the information that you're looking for, check the LCS site for possible hotfixes that include the information. You can also extend the report yourself by creating additional entities.

If the Person search report doesn't contain all the information that the data subject is requesting, you can extend it by using tools that Microsoft has provided. For information about how to extend the Person search report, see Extend the Person search report.

The controller may, at their sole discretion choose to redact certain types of information that may fall outside of the scope of data that must be returned to the data subject as defined within the GDPR.

Right to restrict

An organization might decide to take the following action in response to a DSR request to restrict optional data processing:

- Remove the customer from, for example, a marketing campaign.

* GDPR is not a law exclusive of all other laws. As an enterprise resource planning system, Finance and Operations does not allow for restricted processing of certain business or transactional data, and will not endorse nor provide functionality for the restriction of processing of business data that is necessary for compliance with other laws or certifications. Finance and Operations will not provide support for modifications/customizations or other actions that result in the corruption of referential or business data integrity.

Controller considerations

Controllers can use the following information to complete DSR requests.

System inventory

- **Data inventory and tagging** – Microsoft has enabled a tagging infrastructure that developers and customers can use. Each data field that is defined in metadata contains a classification property that has a suggested value that the controller can confirm or change to any value or term they choose in order to identify data that they deem fits within the definition of personal data.

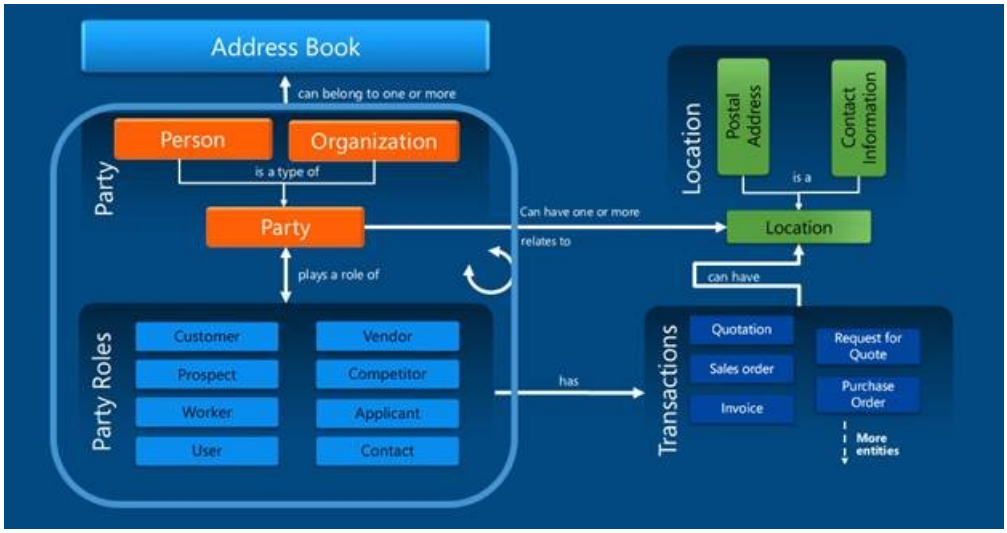
- **Data flow diagram** – Microsoft will publish a data flow diagram that identifies flows of data between systems in the customers production environments.
- **Person search report** – Finance and Operations includes the Person search report, which can be used to gather information in support of a request for a copy of the requestor's personal information.

Activity and diagnostic information

The controller can make DSR requests regarding telemetry data by using the Microsoft Enterprise Privacy Portal. Some telemetry data that we collect is in system generated logs. Without additional information or your assistance, the user's identity is anonymous.

Representation of a person in Finance and Operations

Finance and Operations has a common Global address book. Typically, every time that you add a contact, customer, user, worker, or other person in your system, you first create an address book entry for that person. Each person in the address book is referred to as a party and is assigned a PartyID. The person also takes on a role in the system, such as Customer, User, or Worker, and has a role ID: CustID, UserID, WorkerID, and so on.



Each person is a type of party

Roles that are associated with party records are referred to as party roles. There are several party roles, and they can be assigned to both party types (person and organization):

- **Customer** – An individual, company, or other entity that purchases goods and services that are produced by other individuals, companies, or entities.
- **Prospect** – A party that might be interested in goods or services an organization provides.
- **Worker** – A person who assumes the role of an employee or a contractor, or who is paid in exchange for services.
- **User** – A person who is a user of the system. The user isn't identified in the Global address book.
- **Vendor** – A party that supplies products to one or more legal entities in exchange for payment.
- **Competitor** – A person or organization that provides goods or services that are like the goods or services that your business provides. Out of the box, there is no particular identification for competitors.
- **Applicant** – A person who makes a formal written or electronic request to work for an organization or fill an open position in it.
- **Contact** – A person, either inside or outside your organization, that you've created an entry for. In this entry, you can save information such as the person's street and email addresses, telephone and fax numbers, and webpage URLs.

The right to view and port: It's all about the party

When a data subject approaches the controller to request a copy of their personal data, the controller might choose to use the Global address book information to locate the data that describes the person. As noted in the illustration earlier in this topic, a **person** is a type of **party** that plays a **role**.

Some organizations conduct their activities only through business-to-business relationships and will have modest DSR obligations. By contrast, other organizations conduct their activities through business-to-customer relationships. These organization might choose to use the Global address book and its associative data relationship to write custom reports, custom forms, custom queries, and custom data export features by using the extensibility and customization capabilities and Open in Excel experiences to serve the specific needs of the kinds of data that their business collects from their customers.

The Person search report

To support the controller, this report offers a refinement of the existing entity model reporting functionality that is available in the **Data management** workspace. The **Data management** workspace offers a collection of pre-packaged representations of most role types. These representations are known as entities.

> **NOTE**
> The Person search report is available for Finance, Supply Chain Managament, Commerce, and Human Resources. Currently the report does not support Microsoft Dynamics AX 2012.

An entity represents an instance of a specific role. The data management functionality lets the controller export entity data to several formats, such as colon-separated values, comma-separated values (CSV), semicolon-separated values, tab-separated values, Microsoft Excel, and XML.

The Person search report provides additional capabilities in the **Data management** workspace that export entity data by providing a party ID that is used to identify **all** roles (and corresponding entities) that are associated with the party. This capability lets you export all entity and transaction data in a single action, for either a single party or a collection of parties.

When a data subject approaches the controller to request a copy of their personal data, the controller might choose to use the Global address book information to locate the data that describes the person. As noted in the illustration earlier in this topic, a **person** is a type of **party** that plays a **role**.

Some organizations conduct their activities only through business-to-business relationships and will have modest DSR obligations. By contrast, other organizations conduct their activities through business-to-customer relationships. These organization might choose to use the Global address book and its associative data relationship to write custom reports, custom forms, custom queries, and custom data export features by using the extensibility and customization capabilities and Open in Excel experiences to serve the specific needs of the kinds of data that their business collects from their customers.

Additional notes that apply to requests for data

- Data in Management Reporter and in Microsoft Power BI presentations is generated from the information that is entered in various financial documents and then transferred to those applications for reporting purposes. Any request for data should be fulfilled from the financial documents by using tools such as reports, Export to Excel, and the Person search report. You should not need to do additional reporting from Management Reporter or Power BI to fulfill a GDPR request unless you have made customizations that have altered the base functionality.
- Personal data that is included in documents or attachments might also need to be returned to the data subject, independent of any reporting.
- If a master record has transactional data associated with it, it can't be deleted.
- Similarly, transactions that have been posted or completed can't be deleted.

Reasons why Finance and Operations might not support modifying or deleting data out of the box

The following table lists several reasons why data modifications might be restricted.

REASON	COMMENT
Audit	Data must be preserved for compliance and auditing.
Calculated	Data that has been calculated can be changed only by changing the data that is included in the calculation.
Financial, tax, generally accepted accounting principles (GAAP)	Posted transactions can't be modified or deleted.
Import log	Data must be preserved for compliance and auditing.

What types of personal data might exist in the product

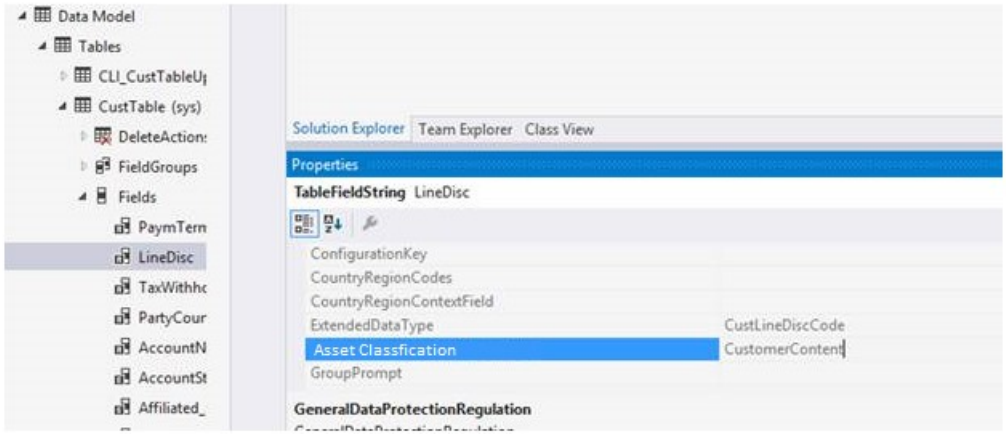
You should expect data requests to come to your company. You can categorize the people who request data into one or, in some cases, more than one relationship with your company:

- Customers
- Vendors
- Workers
- Users
- Warehouse workers
- Truck drivers
- Prospects
- Contacts
- Applicants
- Competitors

Personal data might also be contained in other roles that aren't listed here. Pages used to enter, view or edit personal data have been provided in worksheets for most roles in the preceding list. You can view or download the spreadsheets from the Reference documents for finding and managing personal data page on CustomerSource.

Detailed inventory

As you use Finance and Operations apps, you might find that you generate or collect large amounts of data that resides in multiple data stores. To help you make sense of where your data resides, we've introduced a data marker for each piece of data in our data stores. This marker is called "Asset Classification," and it can be used to identify or track personal data. Any data that you collect has been described as "customer content." Some customer content might contain personal data, and some customer content might contain business data. You can choose to treat all customer content as personal data, or you can change the classification yourself, so that you can identify and track any data that you feel is considered "Personal Data." Although Microsoft has a supplied a set of default classifications, you're free to use any classification or identifiers that you choose.



Age Gating: Preventing minors from using the service

Overview

Microsoft mandates that all users of Microsoft software where personal data is collected must use a Microsoft account (MSA) or Microsoft Azure Active Directory (Azure AD) account for authentication. Additionally, those accounts must be configured to enable minors who use the software or service to affirm parental consent for the service to use their personal data.

What is this feature?

As the tenant admin of the service, you will be required to set up Azure AD Age Gating and/or MSA age gating.

Any user who isn't configured by using Azure Age Gating will be restricted from using the service, even if the user isn't a minor. Age Gating must be configured.

We will restrict access to our software and systems by using a sign-in age gate.

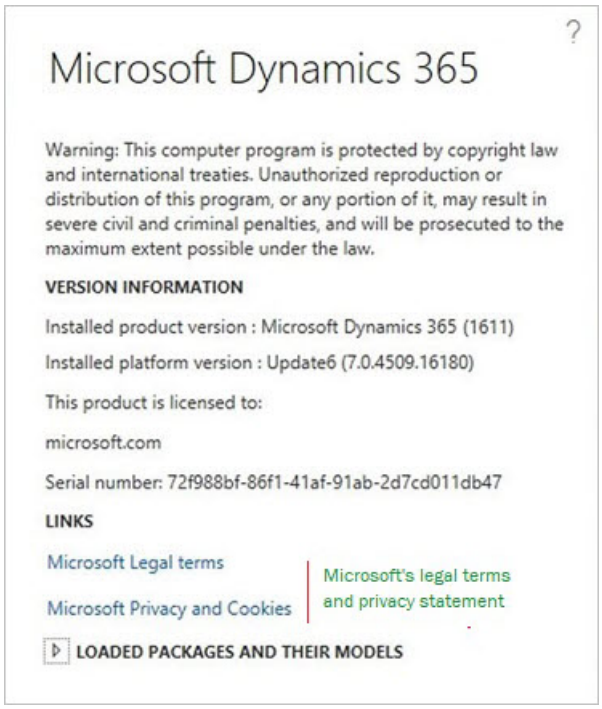
How will age gating work?

The GDPR specifies that systems must stop processing a minor's personal data if that minor doesn't have parental consent. Note that consent can be given and then withdrawn. Therefore, a user might have access to the system one day but not the next.

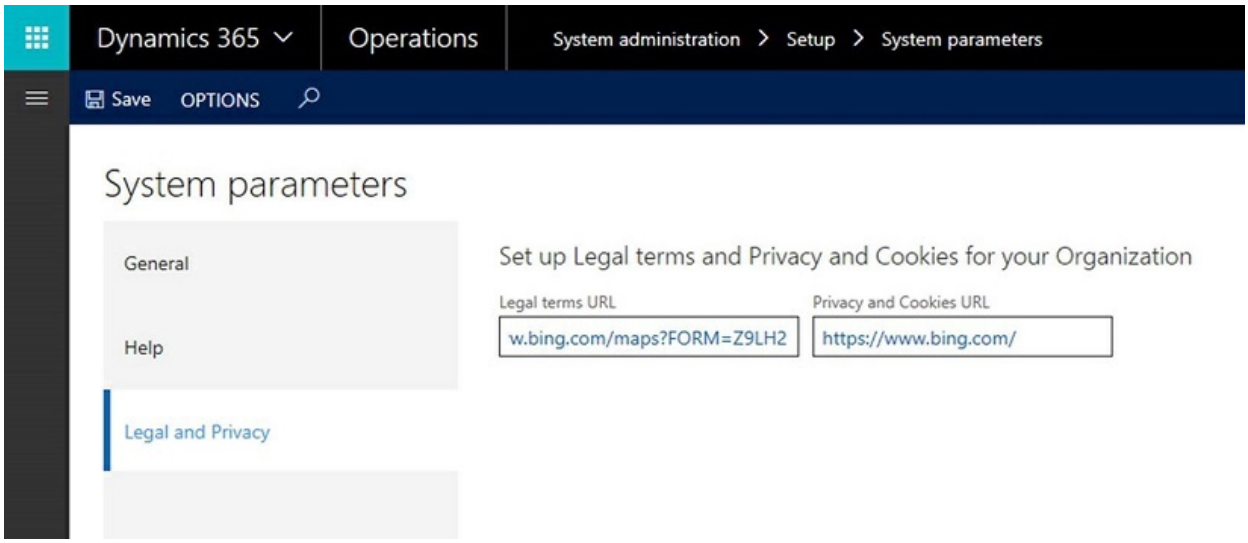
Privacy notices and user subject rights

Displaying your organizations user rights and privacy notice

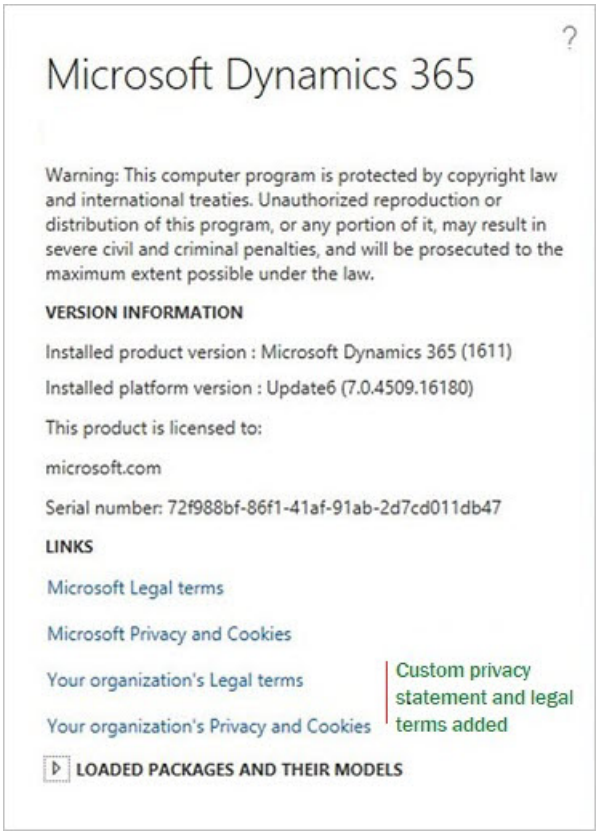
In the **About** box, you will find links to the Microsoft user rights documentation, and to the Microsoft privacy and cookies documentation. You can also add a link to your organization's privacy statement.



On the **System parameters** page, system administrator can add links to the organization's user rights and privacy notices. You can add a valid URL for one or both notice types.



When you've completed your entries in the system parameters, the link to your organization's privacy notice will appear in the **About** box, as show in the following illustration.



Clarification of the scope of this content

- This documentation is a commentary on the GDPR, as Microsoft interprets it, as of the publication date. We have spent a lot of time with the GDPR and believe that we have been thoughtful about its intent and meaning. But the application of the GDPR is highly fact-specific, and not all aspects and interpretations of the GDPR are well-settled.
- This documentation is provided for informational purposes only, and should not be relied upon as legal advice or to determine how the GDPR might apply to you and your organization. We encourage you to work with a legally qualified professional to discuss the GDPR, how it applies specifically to your organization, and how to best ensure compliance.
- MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED, OR STATUTORY AS TO THE INFORMATION PROVIDED IN THIS PRESENTATION. This documentation is provided "as is." Information and views expressed in this documentation, including URL and other Internet website references, may change without notice.
- This documentation does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this presentation for your internal, reference purposes only.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Respond to GDPR data requests resources

2/18/2021 • 2 minutes to read • Edit Online

This topic provides links to information that can help you respond to a request for information under the General Data Protection Regulation (GDPR) as a customer using Dynamics 365 Finance, Supply Chain Management, Commerce, Human Resources, and Microsoft Dynamics AX 2012.

Your first step in responding to a request for data will usually be to use the Person search report to locate the data that's requested. In some cases, you might need to use other reports, access specific pages in the product that you're using, or extend the Person search report. (The report is currently not available for Microsoft Dynamics AX 2012.) This topic points to content that will help you complete those tasks.

Overview

- General Data Protection Regulation overview
- Person search report
- Extend the Person search report
- Manage access to sensitive data

Product-specific considerations

- Respond to requests for personal data in AX 2012
- Respond to requests for personal data in Human Resources
- General Data Protection Regulation overview
- GDPR data requests for Lifecycle Services (LCS)

Compliance Manager

Compliance Manager is a cross–Microsoft Cloud services solution designed to help organizations meet complex compliance obligations like the GDPR. It performs a real-time risk assessment that reflects your compliance posture against data protection regulations when using Microsoft Cloud services, along with recommended actions and step-by-step guidance.

Compliance Manager

You can try Compliance Manager yourself by visiting https://aka.ms/compliancemanager.

Resources

There are a number of resources to help you learn more about Compliance Manager and what it can do you help you meet complex compliance obligations.

- Microsoft 365 - Compliance Manager preview (Office, November 2017)
- Compliance Manager preview is now available (Tech Community, November 2017)
- Get to know the new Service Trust Portal (TechNet, November 2017)
- Announcing Compliance Manager (Tech Community, September 2017)
- Advancing intelligence, management, and security to empower the modern workplace (Office, September 2017)
- New Microsoft 365 features to accelerate GDPR compliance (Microsoft Secure, September 2017)

Additional resources

For more information about the GDPR and the actions that your organization might need to take in response to a request for data, visit the Microsoft Service Trust Portal.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Asset classifications

2/18/2021 • 2 minutes to read • Edit Online

Finance and Operations apps provide a default set of classifications for the kinds of data that are stored in each table. These classifications are subject to change depending on the need to identify different kinds of data. The actual classification for each field in each table can change at any time, depending on differing needs for identifying data.

Through customization, you can change the classification of the following data to meet your own classification and tracking needs.

- **Customer content** – Data collected and managed by the controller (some of which can be personal data).
- **End User Identifiable Information** (EUII) – A natural value used to identify a user of the service.
- **End User Pseudonymous Information** (EUPI) – A generated value used to identify the user of the service.
- **Organizational Identifiable Information** (OII) – A value used to identify the organization using the service.
- **System metadata** – A value that describes the software or used by the software, typically generated by the software.
- **Object metadata** – A value that describes the software or used by the software, but can be provided by the tenant or user of the software.
- **Account data** – A value provided by or used by the tenant to identify the billing information or identify the software used by the tenant.
- **Support data** – Information used to provide customer support.

Person search report

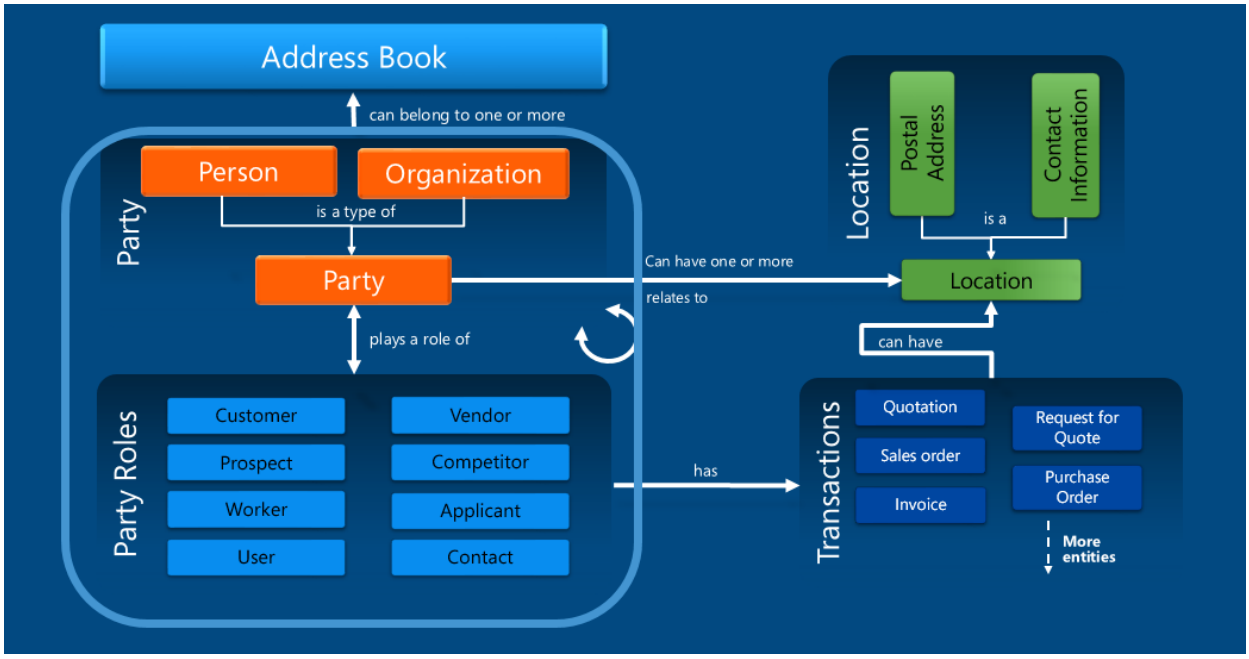
2/18/2021 • 4 minutes to read • Edit Online

The Person search report is a refinement of the existing Data management framework of Finance and Operations apps. The Data management framework offers a pre-packaged set of entities that Microsoft authored to identify personal data that is used to define a person and the roles that a person might be assigned to in Finance and Operations applications.

> **NOTE**
>
> You can use the report with Dynamics 365 Finance, Supply Chain Management, Commerce, and Human Resources. The report is not currently available for Microsoft Dynamics AX 2012. The Person search report is available in version 8.0. The report is also available in version 7.3 (delivered via monthly update 7.3.2), in version 7.2 (via KB 4132615), and in version 7.1 (via KB 4132441). The Person search report may be updated periodically. Before using this report, you need to ensure that you have obtained and applied all relevant hotfixes.

You can use the Global address book to create an instance of a person that is described in the data model as a party.

When you add a contact, customer, user, worker, or other person in Finance and Operations data, you typically start by creating an address book entry for that person. Each person in the address book is referred to as a party and is assigned a PartyID. The person also takes on a role in the system, such as customer, user, or worker, and has a role ID: CustID, UserID, WorkerID, and possibly others.



At times, you might want to verify that the information that is entered and used to describe or otherwise identify a person is correct. Situations might also arise where it's useful to share that information with the data subject who requested the data. The Person search report can help with both these tasks.

The Person search report is extensible. If you find that the existing entities do not contain all of the personal data you are looking for, they can be extended, or new entities can be written. In addition, you can change the data mappings for each entity and remove fields that you don't want to export.

The Person search report lets you specify different identifiers for a person, such as a CustomerID or VendorID. It

will then collect, filter, and populate the entity collection set with personal data that is related only to the person you specified.

On rare occasions, a single person might be entered in your system more than once. The Person search report lets you specify each person instance to be included on a single report. For example, someone named Fred Smith might be both "Fred Smith" and "F. D. Smith" in your address book.

An individual might exist as multiple parties in data. You can provide multiple identifiers for each party type, and each party type's personal data will be included on a single report.

Download the default template

The Person template contains a list of the entities that will be used to download information. The template must be loaded before the Person search report can be used. The template can be loaded from within the Templates form in Data management for versions 7.2 and later. To download templates fronm **Data management**, complete the following steps.

1. Open the **Data management** workspace.
2. If this is the first time that the workspace has been opened, it will load all of the data entities. You must load all the data entities before you load the template.
3. Click the **Templates** tile.
4. Select the **Load default templates** button.
5. Select **Person search**.
6. Click **Load selected**.

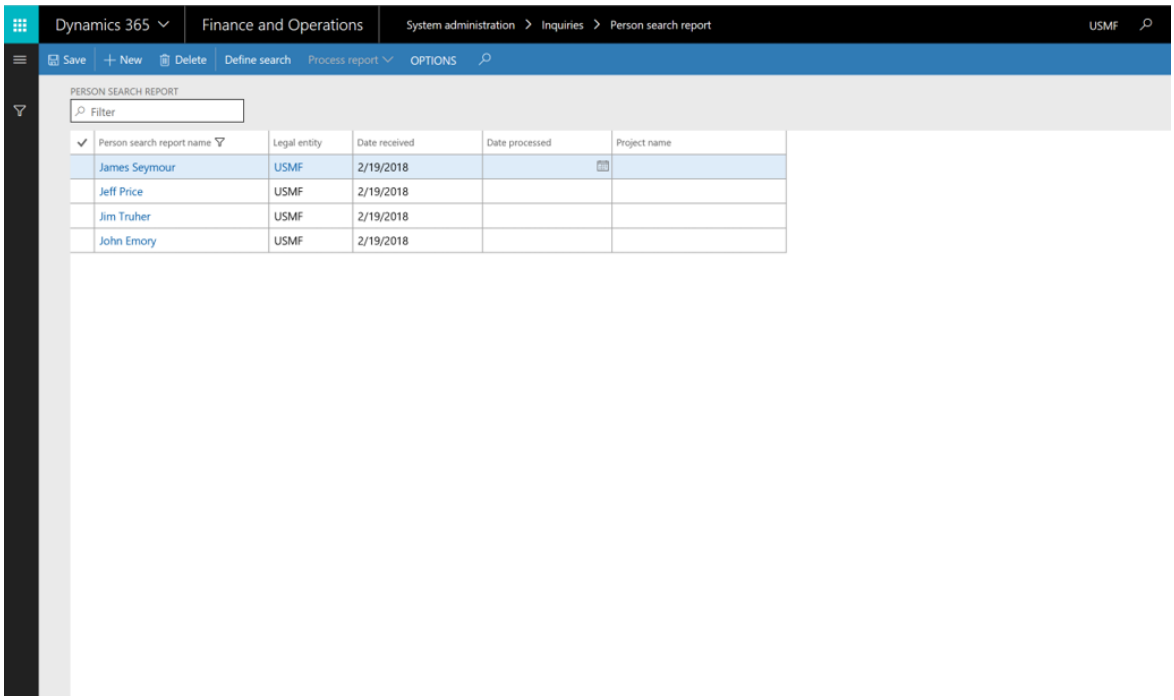
You can also download a template from LCS and import it for versions 7.1 or later. To do so, complete the following steps.

1. Log in to LCS.
2. Click the **Shared asset library** tile.
3. Select the **Data package asset** type.
4. Click the template named **Template-x.x-Person search**, where x.x is the application version that you're using, and download it.
5. Open the **Data Management** workspace.
6. If this is the first time that the workspace has been opened, the workspace will load all of the data entities. All entities loaded before you download the template.
7. Click on the **Templates** tile.
8. Create a new template called **Person search**.
9. Click **Import template**.
10. Browse to the template and click **Upload**.
11. Click **OK** to import the template.

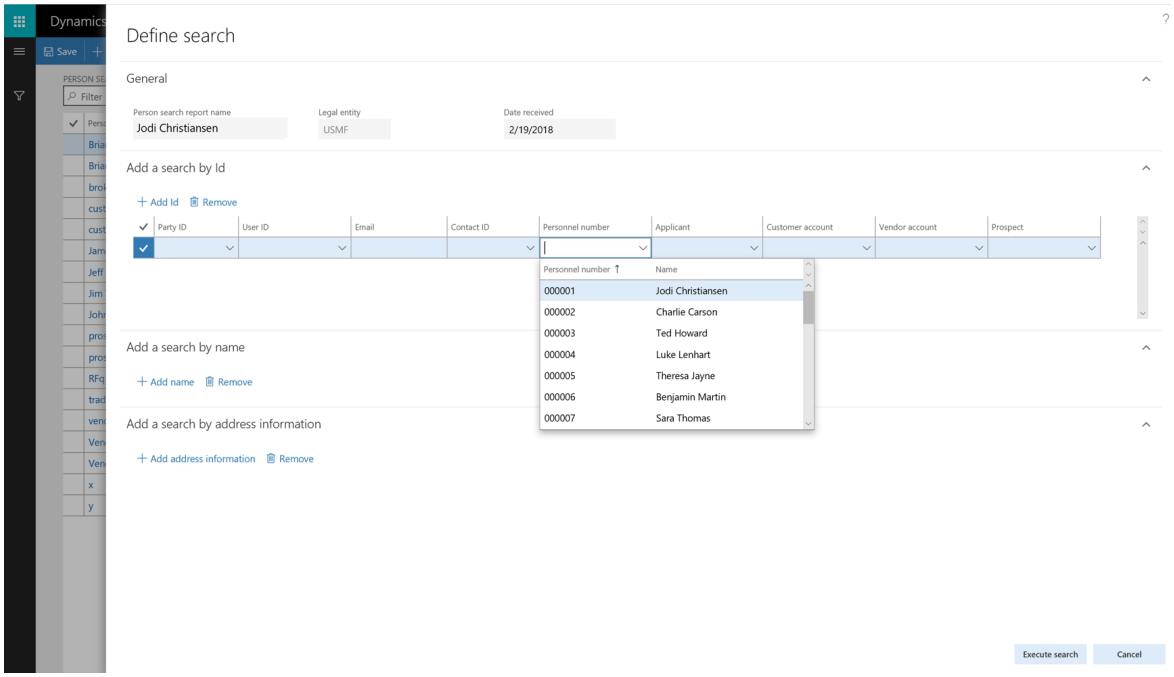
Generate a person search

To use the Person search report, you must complete these tasks.

1. From the System administration menu, open the Person search list page, and create a new search.

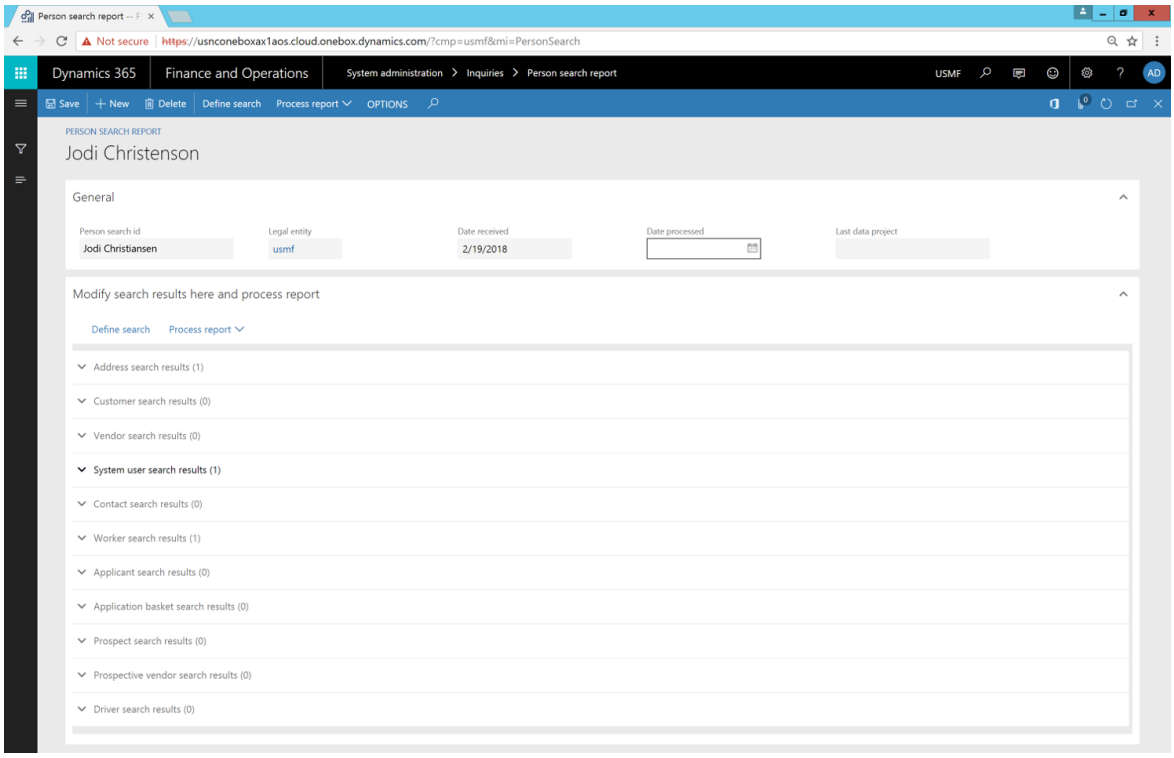


2. The search gives you three options: you can search by ID, by name, or by address. Add the type of search that you want.

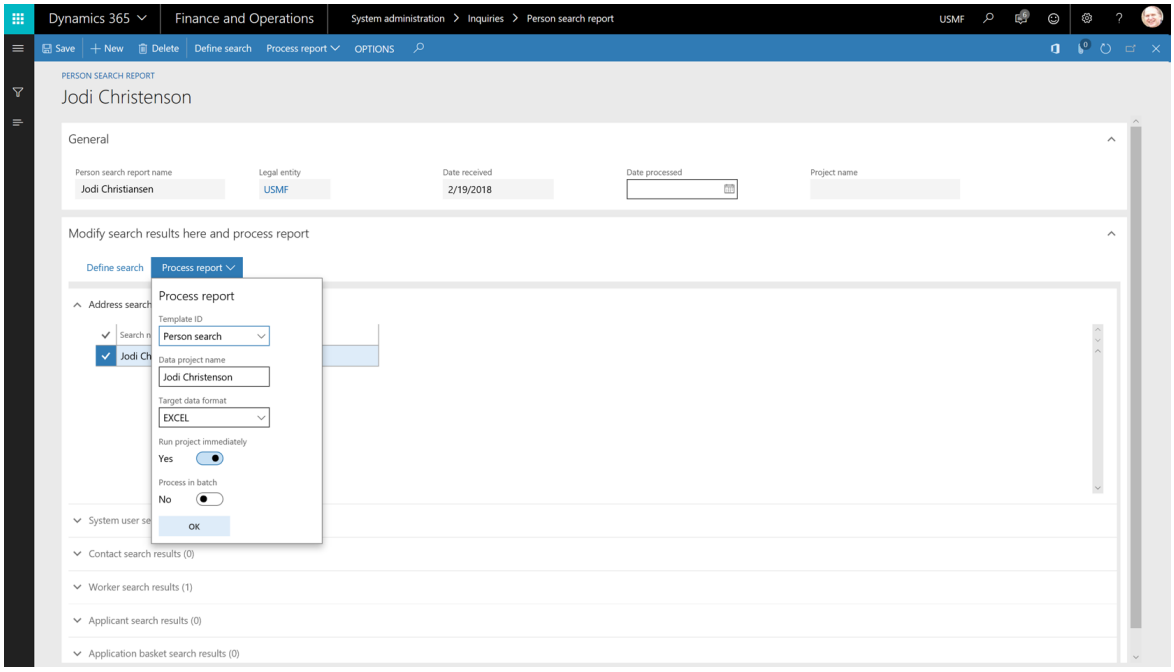


3. Run the search to show the results.

4. Verify that the results are valid. Clear any selections that return information that you don't want to include on the report.



5. Select **Process report**, and then select the Person search template.



6. Select **OK**. A data package is generated.

7. When the package has been generated, export it to your selected data format.

> **NOTE**
>
> Documents that are attached to records are not included in the data export. Attachments must be manually downloaded and shared with the individual who requested personal data.

Additional resources

You can learn more about the GDPR on the European Union's website, from information on the Microsoft Trust Center and in General Data Protection Regulation overview.

Disclaimer

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Extend the Person search report

2/18/2021 • 4 minutes to read • Edit Online

The Person search report for Finance and Operations apps is backed by an intelligent search processor that is designed to manage a collection of entities for a single person. The Person search report searches Finance and Operations data and creates a set of resulting identifiers. Each result references a search category (for example, Customer) and a result record in a related table. For information about using the Person search report, refer the Person search report topic.

> **NOTE**
>
> The Person search is available for Dynamics 365 Finance, Supply Chain Management, Commerce, and Human Resources. The Person search report is not currently available for Microsoft Dynamics AX 2012.

Add another entity to the default template

You can add any entity to the default Person search report template. Open the **Data management** workspace, and select **Templates**. Add the entity to the template. The entity that you add must include at least one of the fields that are used to filter the Person search report.

Create a new search category

1. Use the **PersonSearchResultCategory** enumeration to distinguish different categories of results, such as workers versus applicants.
2. Extend the **PersonSearchResultCategory** enumeration as needed to create new result types.

Create search processing for the new search category

In this example, you will create a new processor class.

1. Extend the **PersonSearchModule** enumeration with a new search area.
2. Create a class that extends the **PersonSearchProcessor** class and includes the **PersonSearchProcessorFactoryAttribute** attribute, with the new person search module area as a parameter.
3. In the **PersonSearchProcessor** extended class, override the **doSearch** method with your desired search logic. As shown in the following example, extend the **PersonSearchResult** table to create new table relationships.

```
PersonSearchResultCategory::Customer needs a relation:PersonSearchResult.ResultRecId =
CustTable.RecId,PersonSearchResult.ResultTableId = CustTable.TableId
```

Integrate with the Global address book (Optional)

If you want to integrate with the Global address book, insert any discovered party numbers into the **PersonSearchPartyNumberTmp** table. The **findPartyLink** method of **PersonSearchProcessor** tries to link party numbers to other search artifacts, such as users, customers, or vendors.

This method has a delegate, **onFindPartyLink**, that lets you specify additional artifacts to search, based on the

party numbers.

The **PersonSearchCriteria** tables let end users specify the parameters by which to search the system for personal data.

Create new search criteria

If you need new search criteria, follow these steps.

1. Extend the **PersonSearchCriteriaName**, **PersonSearchCriteriaAddress**, and **PersonSearchCriteriaKnownId** tables, or create your own tables.
2. Extend the **PersonSearchDialog** form to show these new data fields.
3. Use the new criteria during search processing.

The **PersonSearch** form shows the set of results that was discovered by a person search.

Show the new search category in the PersonSearch form

1. Create a new view on the **PersonSearchResult** table. This view should restrict results to only your new **PersonSearchResultCategory**.
2. Join the view to your result record, and create the view fields that are needed (for example, **PersonSearchResultCustomerView**).
3. Extend the **PersonSearchResult** table to create a new relationship to the new view. This relationship should join on the person search ID.
4. Extend the **PersonSearch** form:
   a. Add the new view as a data source.
   b. Add a new tab to the results with the result grid and the **Include/Exclude** buttons.
   c. Create event handlers for the **Include/Exclude** buttons to update the **PersonSearchResult** records.

      The **PersonSearchEventHandler::updateMarkedOnButtonClicked()** method is provided for convenience.

> **NOTE**
> If you want to see the record count in the result caption, create an event handler on the **OnQueryExecuted** view data source event. Next, call the **setResultCountOnGridCaption()** method on the **PersonSearch** form to update the count.

Each **PersonSearchEntityFilterRelation** record specifies the conditions when a filter should apply, and the filter table and field to apply.

The set of filter relations is compared to the template metadata when the package is built.

For a filter to be created, a **PersonSearchResult** record with the matching filter category must exist. After it's found, the **PersonSearchResult** references the table field where the filter value resides.

Create new filters

1. Use the Chain of Command to extend the **PersonSearchEntityFilterRelation** table.
2. Decide on the type and source of the new filter:

a. If the filter is an extended data type (EDT) or enumeration, set the **MetadataTypeId** to the data type ID.
b. If the filter is a source table field, specify the source table and source field IDs.
c. If the filter is an entity field, specify the entity field ID.

3. Decide on the filter table and filter field.

   The filter table must be available as a **PersonSearchResult** with a matching category. Otherwise, no filters will be created.

4. Insert the new filter record.

   The person search framework will automatically manage initialization of all exclusions when the form is first opened. Exclusions let you suppress the filter building functionality for specific entity fields.

Create new exclusions

1. Use the Chain of Command (COC) to extend the **PersonSearchEntityExclusion** table.
2. In the **CoC method**, specify the entity, the entity field, and whether the exclusion is active.
3. Insert the new exclusion record.

   The person search framework will automatically manage initialization of all exclusions when the form is first opened.

Additional resources

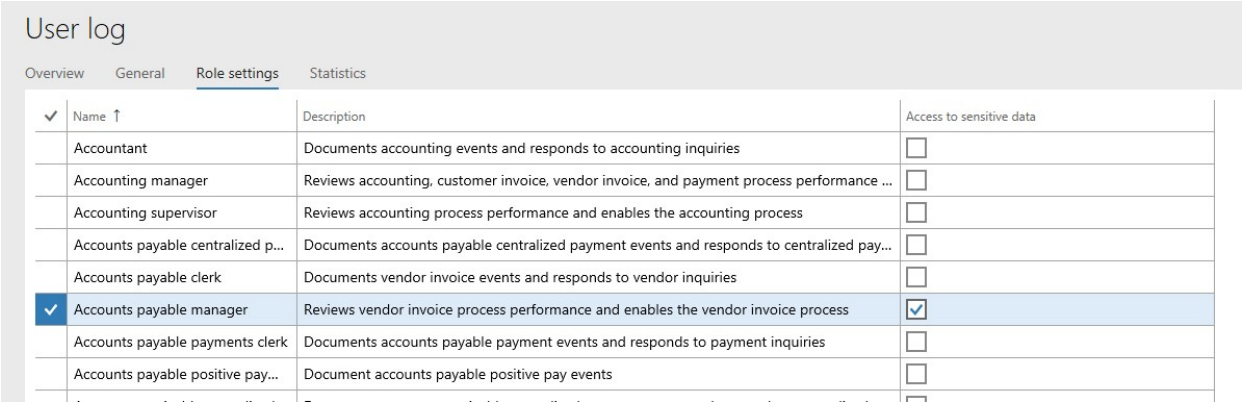
If you're extending the Person search report as part of a response to a request for data under the General Data Protection Regulation (GDPR) in the European Union, more information about that regulation is available in the General Data Protection Regulation overview.

You can learn more about the GDPR on the European Union's website and on the Microsoft Trust Center.

Manage access to sensitive data

2/18/2021 • 2 minutes to read • Edit Online

System administrators can use the **User log** page to keep an audit log of users who have logged on to the system. Knowing who has logged in can help protect your organization's data. We've enhanced the user logging capability to let the administrator identify roles that provide access to sensitive data.



Name	Description	Access to sensitive data
Accountant	Documents accounting events and responds to accounting inquiries	☐
Accounting manager	Reviews accounting, customer invoice, vendor invoice, and payment process performance ...	☐
Accounting supervisor	Reviews accounting process performance and enables the accounting process	☐
Accounts payable centralized p...	Documents accounts payable centralized payment events and responds to centralized pay...	☐
Accounts payable clerk	Documents vendor invoice events and responds to vendor inquiries	☐
Accounts payable manager	Reviews vendor invoice process performance and enables the vendor invoice process	☑
Accounts payable payments clerk	Documents accounts payable payment events and responds to payment inquiries	☐
Accounts payable positive pay...	Document accounts payable positive pay events	☐

What is sensitive data?

An organization can define what constitutes *sensitive data* in whatever way serves its needs. For some organizations, sensitive data might be any data that is related to financial or human resource data, or just data that is personal data. Some industries or some countries or regions might have a more specific definition of sensitive data that an organization can adopt for itself. It's up to each organization to decide whether and how to use the sensitive data identifier.

The sensitive data identifier enhances the user logging experience by letting your organization produce audit logs that show who in your system has access to sensitive data. This capability is helpful for organizations that might have multiple roles that have varying degrees of access to certain data. It can also be helpful for organizations that want a detailed level of auditing to track users who have had access to data that's been identified as sensitive data.

ID	Name	Type	Time ↓	Online time	RolesWithAccessToSensitiveData
Admin	Admin	Logon	11/16/2017 03:10:08 PM	0:00:00	[Auditor][System administrator]
Admin	Admin	Logon	11/16/2017 02:43:14 PM	0:23:40	[System administrator]
Admin	Admin	Logon	11/16/2017 01:59:17 PM	0:43:56	
Admin	Admin	Logon	11/16/2017 01:49:21 PM	0:00:09	

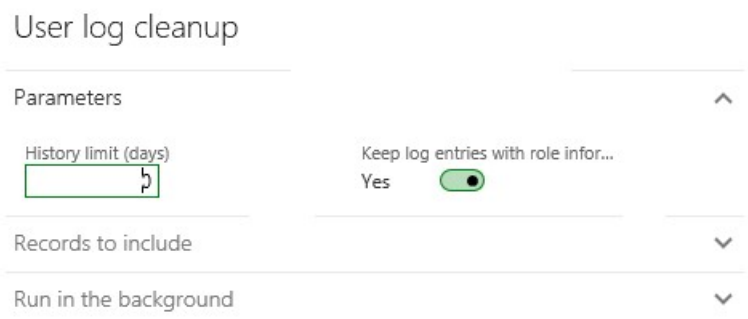
Language-specific information

The role information in the user log is language-specific and matches the current user language.

Tidspunkt ↓	Onlinetid	Roles with access to sensitive data
11/16/2017 05:29:31 PM	0:00:00	[Revisor],[Systemadministrator]
11/16/2017 05:10:15 PM	0:17:44	[Revisor],[Systemadministrator]
11/16/2017 03:47:25 PM	0:52:03	[Revisor],[Systemadministrator]
11/16/2017 03:21:56 PM	0:24:22	

Log retention

The log entries of users who have access to data that's been declared to be sensitive data can be retained separately from all other data in the log. The administrator can enable this functionality by setting an option on the **User log cleanup** page.



> **NOTE**
>
> This feature is available in version 8.0. This feature is available for Dynamics AX 2012 R3 (via KB 4074643)

Additional resources

You can learn more about the GDPR on the European Union's website and on the Microsoft Trust Center.

Disclaimer

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Respond to requests for personal data in Human Resources

2/18/2021 • 5 minutes to read • Edit Online

This topic can help both businesses that use Microsoft Dynamics 365 Human Resources, and also partners and independent software vendors (ISVs), when they comply with data subject rights (DSR) requests. For more information about the European Union's General Data Protection Regulation (GDPR) and the related resources that Microsoft provides, see General Data Protection Regulation overview.

For Human Resources, Microsoft acts as a processor. As a data processor, Human Resources provides processes and features that let you comply with your GDPR obligations as a data controller.

Rights

Data subjects have the following rights under the GDPR, and a data controller might take any of the actions that are listed under each right in response to a DSR request.

Right to view

- Use the Person search report to find and collect personal data that is subject to a DSR request. For information about using this report, see the Person search report topic.
- Use advanced search and filters to find specific personal data and export that data by using the Microsoft Office Export functionality.
- Extend the Person search report by adding an existing entity. For information that can help you extend the report, see Extend the Person search report.

Right to modify*

- Use advanced search and filters to find the data that should be corrected, and correct the data directly in Human Resources.

* You might find that some data that qualifies as personal data can't be modified directly in the product or feature. Typically, this data is part of a financial transaction or other business data that is kept "as is" for compliance with financial laws (for example, tax laws), prevention of fraud (such as security audit trail), or compliance with industry certifications. As the controller, it's your responsibility to correct inaccurate or incomplete personal data.

Right to be forgotten*

- You can delete or erase personal data where the product enables that action directly. As the controller, you should ensure any personal data that a data subject requests be erased does not conflict with other compliance obligations your organization may have around data retention, for example proof of payment or proof of tax.

* You might find that some data that qualifies as personal data can't be modified directly in the product or feature. Typically, this data is part of a financial transaction or other business data that is kept "as is" for compliance with financial laws (for example, tax laws), prevention of fraud (such as security audit trail), or compliance with industry certifications. As the controller, it's your responsibility to correct inaccurate or incomplete personal data.

Right to port

The following options are available to help you port personal data in response to a data rights request.

- Use the Microsoft Office Add-in to export personal data.
- Author a custom report that enables the export of personal data.
- Use or extend the Person search report to gather information in support of a request for a copy of the data subject's personal data.

Right to restrict processing

- Remove the employee from, for example, a course.
- Following guidance from an organization's legal counsel, the company might refuse the right to restrict processing where data is needed by the company for compliance with other legal or industry mandates.

Additional notes that apply to requests for personal data

- Personal data that is found in Microsoft Power BI is generated from the information that is entered in Human Resources and then transferred to that application for reporting purposes. Any request for personal data should be fulfilled from the information in Human Resources, by using tools such as reports, Export to Excel, and the Person search report. You should not need to do additional reporting from Power BI to fulfill a DSR request.
- Human Resources doesn't export documents that are attached to records. These attachments must be manually downloaded and shared with the individual who has made the DSR.
- If transactional data is associated with a master record, that record can't be deleted.
- Similarly, transactions that have been posted or completed can't be deleted.

Reasons why certain personal data may not be modified or deleted in Human Resources

The following table lists several reasons why personal data modification or deletion is restricted in certain scenarios.

REASON	COMMENT
Financial, tax, generally accepted accounting principles (GAAP)	A party can't be deleted, but the party's name can be updated.
Financial, tax, GAAP	A current worker's data can't be deleted, but the worker's name can be updated.
GAAP	Posted or completed transactions can't be modified.

Additional information

Only terminated workers can be deleted from Human Resources. Follow these steps to delete terminated workers.

- Delete position assignments.

  To delete a position assignment, select **Position assignments** on the **Worker** page. Select **As of date** and select **Display all records**. Drill into the position number, select **Changes timeline** > **Manage changes** > **Position worker assignments**, and remove the position assignment record that is associated with the worker that you're deleting.

- Delete fixed compensation.

  To delete fixed compensation, select **Employment history** on the **Worker** page. Select **Employment history** > **Employment** > **Fixed compensation**, and delete the fixed compensation plans for the worker.

- Delete variable compensation enrollments.

  To delete variable compensation, select **Employment history** on the **Worker** page. Select **Employment history** > **Employment** > **Variable compensation plan enrollment** and delete the variable compensation plan enrollments for the worker.

- Delete any associated checklists.

  To delete the checklists, select the **Checklists** option on the **Worker** page.

Compensation isn't assigned to contractors. Therefore, those steps can be skipped in the preceding process.

An admin can search and export personal data in Microsoft Dynamics 365 Talent: Attract and Microsoft Dynamics 365 Talent: Onboard via https://attract.talent.dynamics.com/personreport.

Additional resources

You can learn more about the GDPR on the European Union's website and on the Microsoft Trust Center.

Disclaimer

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Respond to requests for personal data in AX 2012

2/18/2021 • 2 minutes to read • Edit Online

This topic can help businesses that use Microsoft Dynamics AX 2012, implementation partners, and independent software vendors (ISVs) comply with requests from a data subject concerning their personal data under the European Union's General Data Protection Regulation (GDPR). For more information about the GDPR and the related resources that Microsoft provides, see the General Data Protection Regulation overview. Although the "Guide to the GDPR for Finance and Operations" was written for Finance and Operations apps, the rights that a data subject has under the GDPR apply to organizations using Dynamics AX 2012. The steps listed in the "Guide to the GDPR for Finance and Operations" can be used by an organization responding to a request for personal data and are also appropriate for organizations using Dynamics AX 2012, with the exception of the **Person search** report, which is not available for Dynamics AX 2012. This topic lists additional points that are specific to Dynamics AX 2012.

Applicable product updates for Dynamics AX 2012

Additional product updates and information related to GDPR that's specific to AX 2012 are available from Microsoft Dynamics Lifecycle Services (LCS). Sign-in is required for access to LCS.

Here are some example inquiries and reports that are available in AX 2012 that might help you find and report personal data:

- **Home** > **Common** > **Global address book**

  In the inquiry window, enter a person's name in the search box.

- **Accounts payable** > **Common** > **Vendor** > **All vendors**
- **Accounts receivable** > **Common** > **Customer** > **All Customers**
- **Human resources** > **Common** > **Workers**
- **Sales and marketing** > **Common** > **Customers**, **Prospects**, **Leads**, or **Contacts**

Additional resources

You can learn more about the GDPR on the European Union's website and on the Microsoft Trust Center.

GDPR data requests for Lifecycle Services (LCS)

2/18/2021 • 2 minutes to read • Edit Online

Overview

This topic offers information for complying with requests for data under the General Data Protection Regulation (GDPR) if the request requires accessing data in Microsoft Lifecycle services (LCS). For general information about this regulation and the resources Microsoft is providing to support compliance with it, see General Data Protection Regulation overview.

User management

A tenant administrator can view and manage users on the **Organization users** tile on the home page in LCS.

An administrator can view and manage all organization users on the **Organization user** page in LCS.

To export a user, the administrator can use filtering to find a specific user, select that user, and copy the name and email address of the user.

Additional resources

- Official GDPR site
- Microsoft Service Trust Portal

Lifecycle Services resources

2/18/2021 • 2 minutes to read • Edit Online

Lifecycle Services (LCS) for Microsoft Dynamics is a collaboration portal that provides an environment and a set of regularly updated services that can help you manage the application lifecycle of your implementations of the Dynamics 365 Finance and Operations apps.

- What's new in Lifecycle Services (LCS)
- Lifecycle Services (LCS) user guide
- Projects in Lifecycle Services (LCS)
- Project onboarding
- Methodologies in Lifecycle Services (LCS)
- Business process modeler (BPM) in Lifecycle Services (LCS)
- Cloud-hosted environments in Lifecycle Services (LCS)
- Manage the support experiences for Finance and Operations apps
- Configuration in Lifecycle Services overview
- Customization analysis in Lifecycle Services (LCS)
- Infrastructure estimator in Lifecycle Services (LCS)
- Issue search in Lifecycle Services (LCS)
- License sizing estimator in Lifecycle Services (LCS)
- Request for proposals (RFP) responses
- System diagnostics in Lifecycle Services (LCS)
- Upgrade analysis in Lifecycle Services (LCS)
- Usage profiler in Lifecycle Services (LCS)
- Downloadable tools in Lifecycle Services (LCS)

Additional resources

- For information about how to contact Microsoft if you have technical questions about Dynamics 365 Finance and Operations apps, or if you need help accessing Microsoft Dynamics Lifecycle Services (LCS), see Get support for Finance and Operations apps or Lifecycle Services (LCS).
- For information about how to contact Microsoft if you have technical questions about Microsoft Dynamics AX 2012 or need support, see Manage the support experiences for Finance and Operations apps.

Lifecycle Services (LCS) for Finance and Operations apps customers

2/18/2021 • 4 minutes to read • Edit Online

This topic is intended for customers who have signed up for the current versions of Finance and Operations apps. Partners who are working with customers to help them move through the lifecycle of their Lifecycle Services (LCS) project will also find this information useful.

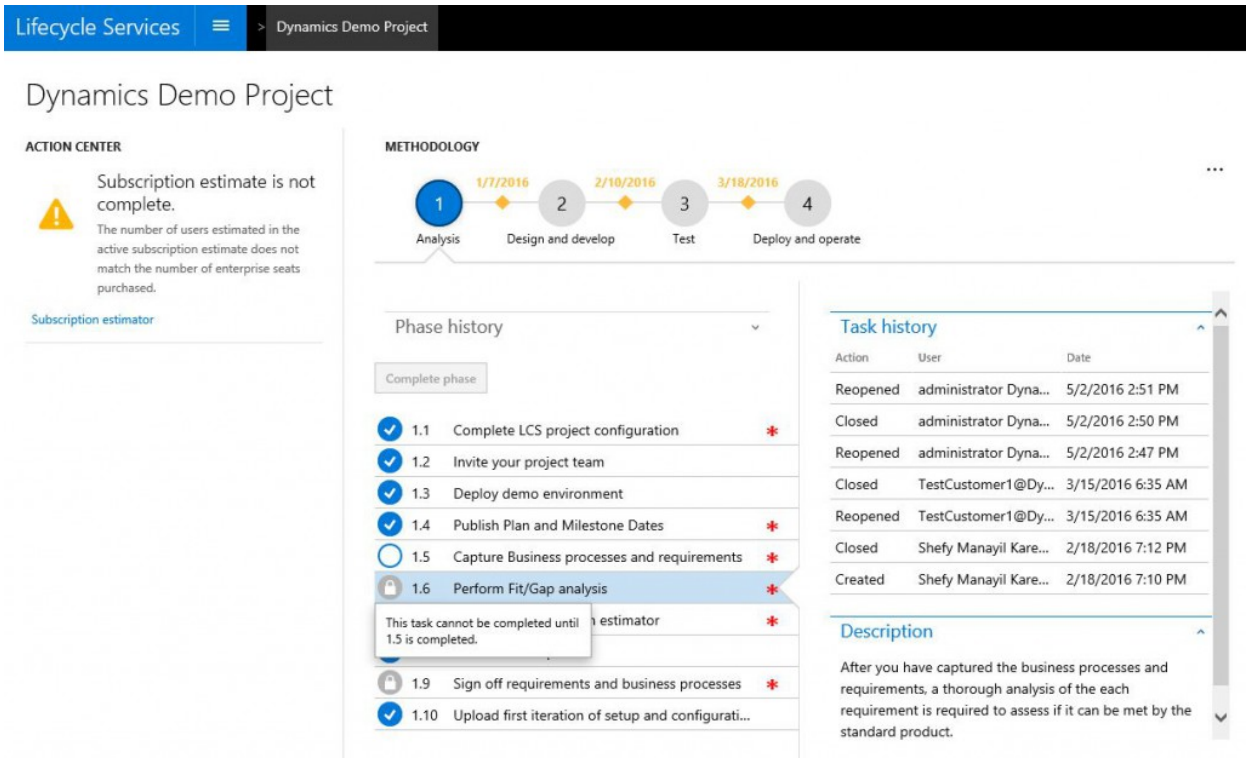
LCS workspace for the current versions of the Finance and Operations apps

When you sign up for the current versions of Finance and Operations apps, your subscription includes an Implementation project workspace. After you activate the service, the tenant administrator must sign in at https://lcs.dynamics.com by using the tenant account. The project workspace is automatically created for your organization. The workspace includes the following elements:

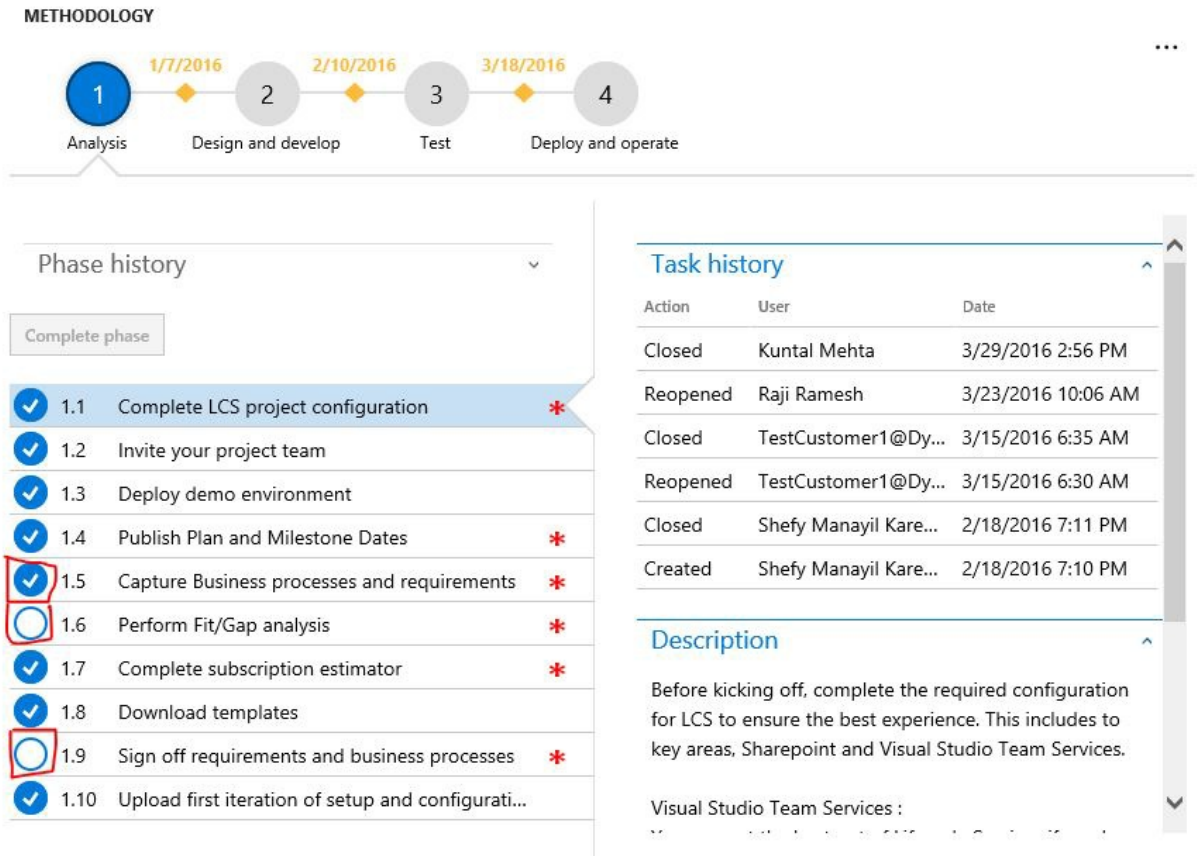
- Enabled features, based on the offer that you selected
- Environments that are deployed and managed by Microsoft
- Guidance that is provided through the Action center to help you complete required actions
- A new methodology experience that includes tasks that lock as you move through the implementation
- A more complete history that specifies who completed each methodology phase and task
- Milestones that you can use to track critical project dates
- Various services to help you with your implementation

Methodologies

As a customer, you must complete the steps that are outlined in the methodology to gain access to the production environment. Before a phase can be marked as completed, you must complete the specified mandatory tasks. Locked tasks, such as tasks 1.6 and 1.9 in the following screenshot, are unlocked after you've completed the required actions. To learn which actions must be completed before a specific task can be unlocked, click the lock icon for that task.

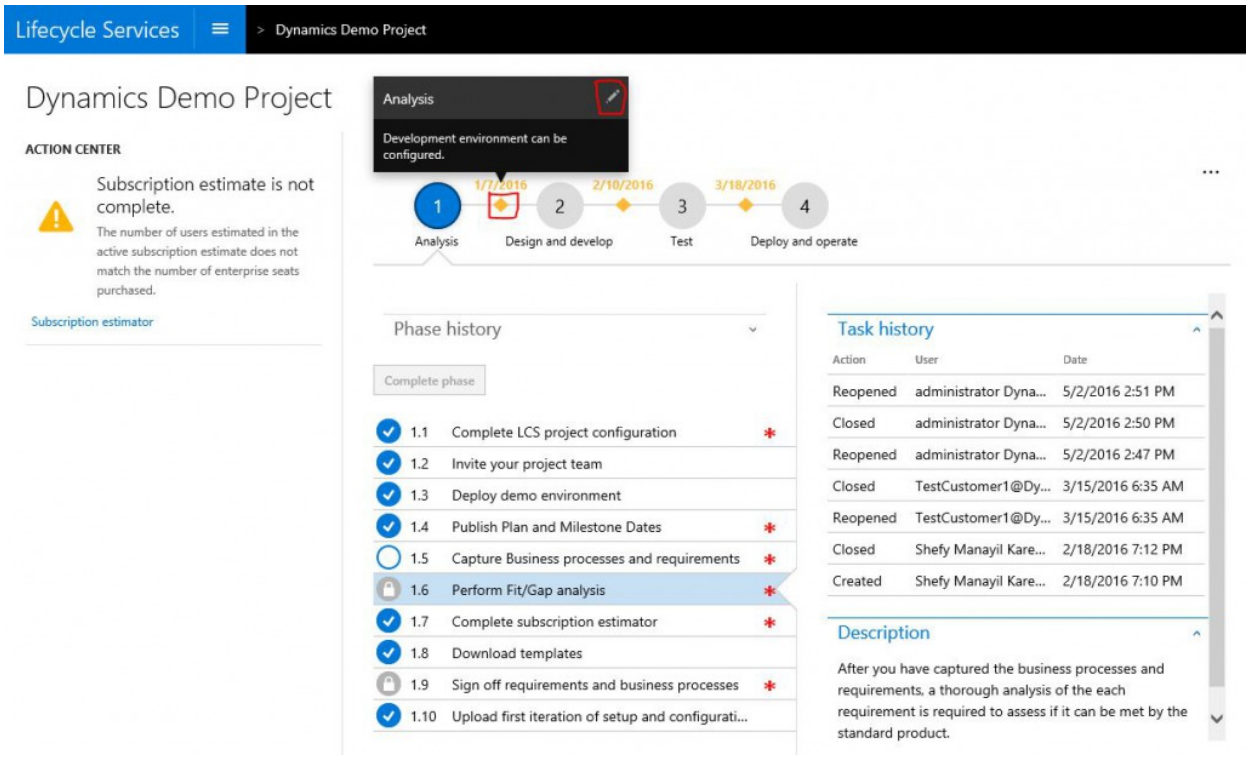


In the case of prerequisites, after you complete the required tasks, you can mark the dependent tasks as completed. For example, in the following screenshot, tasks 1.6 and 1.9 depend on task 1.5. Because task 1.5 has now been completed, the two dependent tasks can be marked as completed.

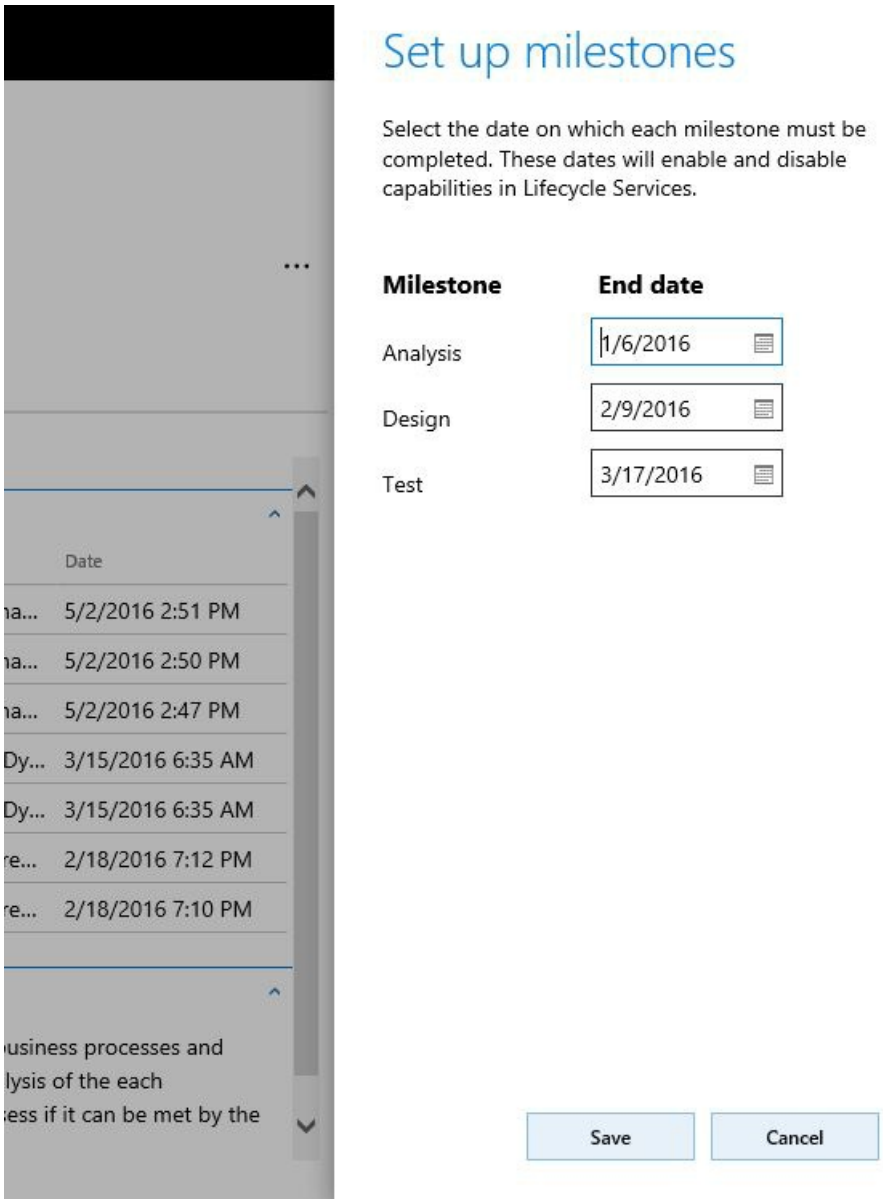


Milestones

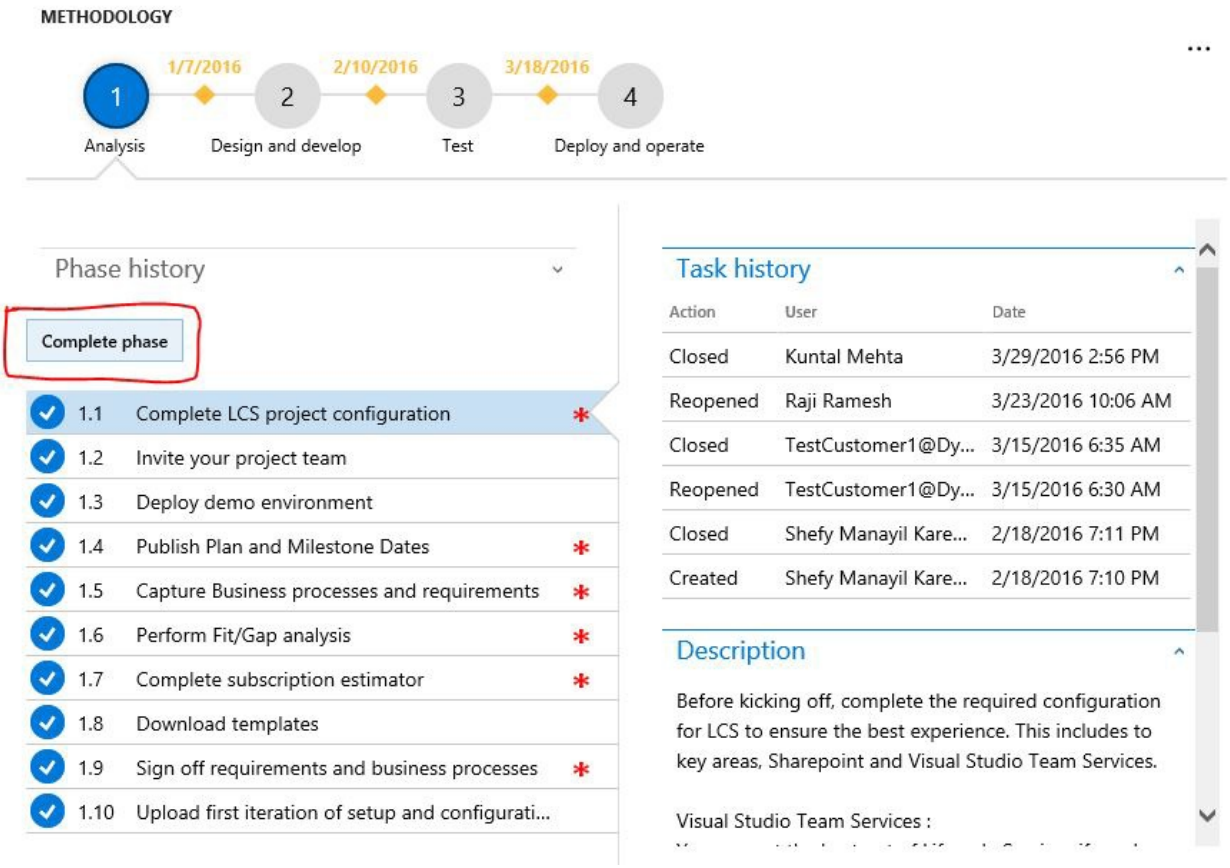
High-level milestones must be defined for a project. Milestones can help you track the deliverables that must be completed and your progress toward the milestone goals. Color indicators help you quickly learn whether you're behind schedule. For example, in the following screenshot, the milestones are yellow. To enter or update the milestone dates, click the diamond shape in the methodology, and then click the **Edit** button (pencil icon). You can change milestone dates at any time.



When you've finished entering milestones, the **Publish plan and milestone** task opens, and you can mark it as completed.

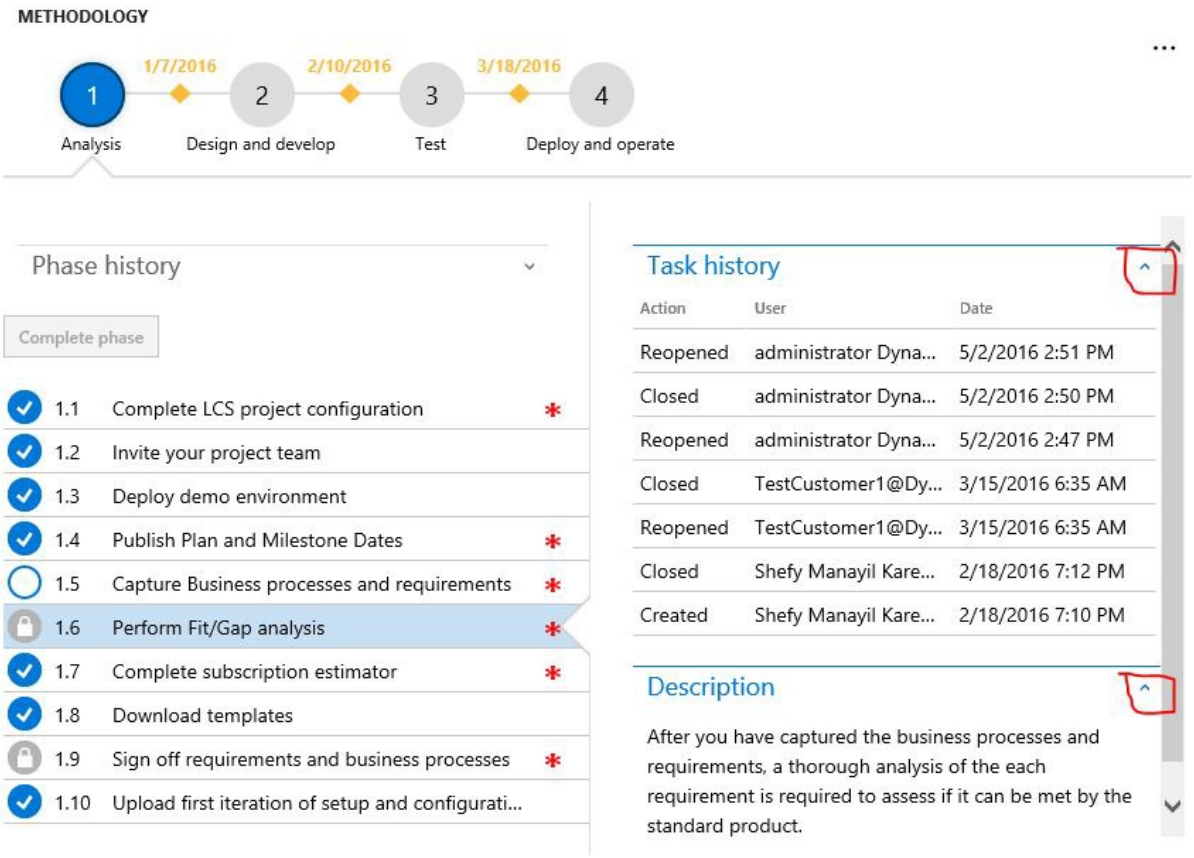


When you've completed all the required tasks in a phase, you can click **Complete phase** to mark the phase as completed. After you mark a phase as completed, next steps become available in Microsoft Dynamics Lifecycle Services (LCS).



Methodology description and history

Descriptions can help you understand what is expected of you for a specific methodology task or phase. You can expand the methodology description to learn more about each task, and then collapse the description when you've finished. The task and phase history can tell you when a task or phase was completed or reopened. If you're a project manager, this information can help you stay on top of the high-level tasks that are required for your implementations.



Subscription estimator

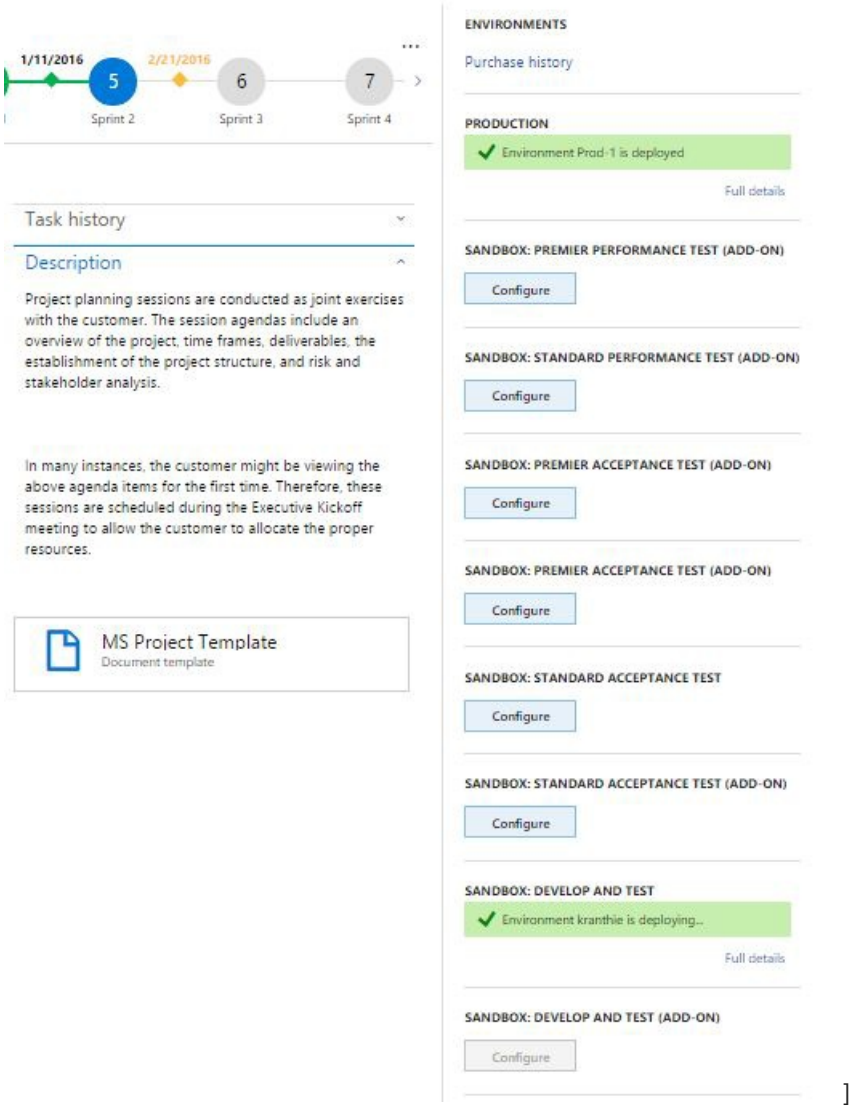
You can use the Subscription estimator tool to evaluate your subscription requirements for the current versions of the Finance and Operations apps. To use Subscription estimator, download the usage profile, which is a Microsoft Excel workbook. Then, in the workbook, complete the following worksheets:

- Deployment details
- Instance Characteristics
- Retail & Commerce

After you've completed the worksheets, enter the data from the summary sheet into Subscription estimator by clicking **+ New estimate**. You must make one estimate the active estimate. Make sure that the estimate that you mark as active is same as the offer that you bought through the VL or CSP channel.

New deployment experience

To provision your environment, you must to complete a configuration checklist. As you make progress through the methodology, environments become available to you. Click **Configure** to add deployment information.



]

Because the information that you enter determines your experience, carefully review your input. After you've entered all the required information, sign-off is required for the deployment request. The user who completes the sign-off becomes the system administrator on the instance. Verify that the correct user completes the sign-off for the deployment. After the sign-off is completed, the Microsoft site reliability engagement team reviews the request. After the team has reviewed the information that you entered, it initiates the provisioning. If the information isn't correct, the team will contact you. After the provisioning is completed, the status is updated to indicate that the environment has been deployed, as shown in the following screenshot. If the provisioning takes longer than expected, the Microsoft site reliability engagement team reviews the status and takes appropriate actions. These actions might include contacting you. After the environment is provisioned, click **Full details** to open the **Detailed environment** page, where you can sign in to the system, view the monitoring status, or view relevant updates.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Lifecycle Services (LCS) for Finance and Operations apps partners

2/18/2021 • 2 minutes to read • Edit Online

This article explains how partners can get started with Microsoft Dynamics Lifecycle Services (LCS).

As a Finance and Operations partner, you can access the current version by following the steps in Sign up for preview subscriptions.

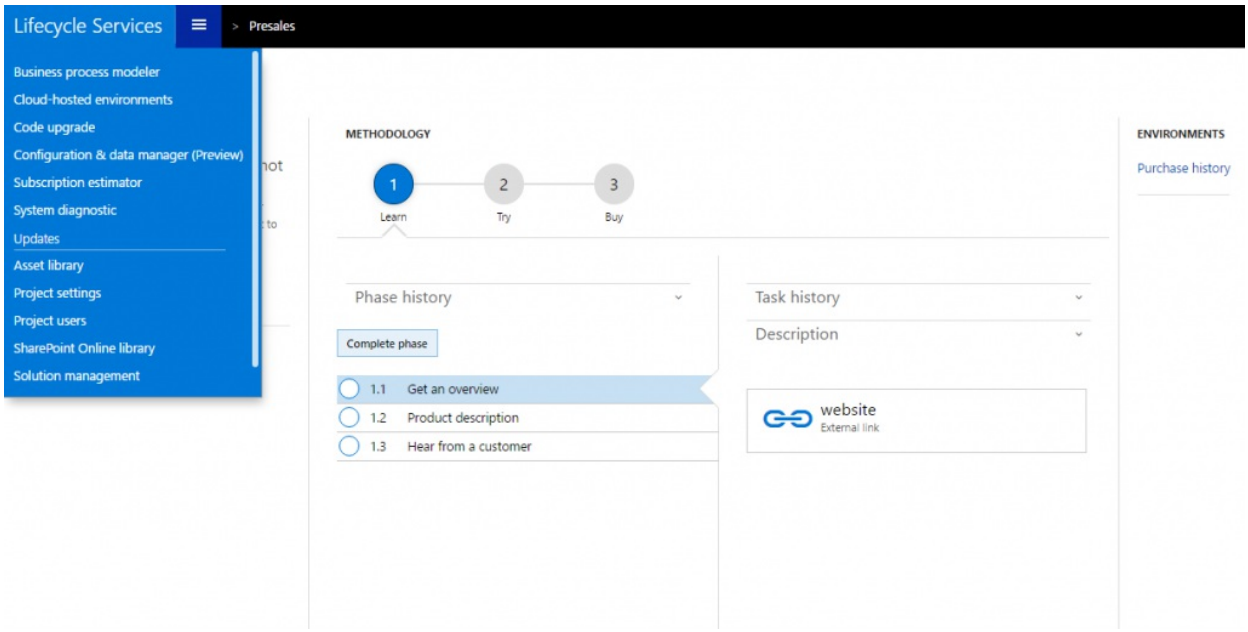
Projects for partners in LCS

After you sign up as a partner for the current version, you can create two types of projects:

- Prospective presales
- Migrate, create solutions, and learn the Dynamics 365 Finance and Operations apps

Prospective presales project

Work with new prospects to help them understand the business processes that are available, and to help them evaluate their subscription needs. Note that only partners can provision a new cloud environment.



Migrate, create solutions, and learn the project for Finance and Operations apps

Create a Microsoft Dynamics AX 2012 project workspace. You can use the project to support the upgrade from Dynamics AX 2012 to the current version. You can also use the project to learn how to use the current version or create solutions in Microsoft Dynamics AX Lifecycle Services (LCS).

Projects for customers in LCS

For every customer who signs up for LCS, an Implementation project workspace is automatically created during the sign-up process. As a partner, you can't create an Implementation project. For more information about the Implementation project workspace, see Lifecycle Services (LCS) for Finance and Operations apps customers. Services within project workspaces behave in the same manner. However, an important difference between an Implementation project and other project types is the ability to configure the current version of Finance and Operations, which is managed by Microsoft.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

What's new or changed in Lifecycle Services (LCS)

2/18/2021 • 2 minutes to read • Edit Online

Microsoft Dynamics Lifecycle Services (LCS) provides a cloud-based collaborative workspace that customers and their partners can use to manage projects from pre-sales through implementation and operations. LCS provides checklists and tools to help you manage your project, based on the phase of the project and the industry that you're working in. It also provides a dashboard, so that you have a single location where you can obtain up-to-date project information.

To get started with LCS, see the Lifecycle Services (LCS) user guide.

> **IMPORTANT**
> LCS features and service changes will no longer be announced via blog posts. Descriptions of LCS features are provided in the release plans.

The following sections list the features that are included in LCS releases.

June 2020 - wave 1

AREA	FEATURE	STATUS
Business process modeler (BPM)	Download task recording (AXTR)	General availability
Service updates	View canceled updates	General availability

May 2020 - wave 2

This release contains general performance improvements and minor bug fixes.

May 2020 - wave 1

AREA	FEATURE	STATUS
Issue Search improvements	Lifecycle Services Issue search improvements	General availability

April 2020 - wave 2

AREA	FEATURE	STATUS
Environment actions	Lifecycle Services support for dual-write capabilities	Preview
Environment actions	Removing Remote Desktop access to Tier 2-5 Standard Acceptance Test (or sandbox) environments	Preview

April 2020 - wave 1

This release contains general performance improvements and minor bug fixes.

March 2020 - wave 2

AREA	FEATURE	STATUS
Admin APIs	RESTful APIs for database export	Preview
Environment actions	Apply data upgrade packages for AX2012 customers on sandbox environments	General availability
Environment actions	Platform update 20 required for database movement operations	General availability

March 2020 - wave 1

This release contains general performance improvements and minor bug fixes.

February 2020 - wave 2

AREA	FEATURE	STATUS
Environment actions	Platform update 20 required for database movement operations	General availability

February 2020 - wave 1

AREA	FEATURE	STATUS
Admin APIs	Database movement RESTful APIs	General availability

LCS releases before November 2019

For information about LCS releases that occurred before November 2019, see the blog posts that the Lifecycle Services team published on the Dynamics 365 blog.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Known issues

2/18/2021 • 3 minutes to read • Edit Online

This topic explains how to view the known issues that you might experience when you use Finance and Operations.

You can find known issues in Finance and Operations by using the Issue search tool in Microsoft Dynamics Lifecycle Services (LCS). You can sign in to LCS by using your CustomerSource or PartnerSource account. Then, in the **Issue search** field, enter the search text that is shown in the following table.

> **NOTE**
> Be sure to enter the text exactly as it appears in the table. Otherwise, you might get different search results.

Application releases

RELEASE	VERSION	BUILD NUMBER	AVAILABILITY	SEARCH FOR THIS TEXT
Microsoft Dynamics 365 for Finance and Operations	8.0	8.0.30.8020	April 2018	Known issue in Dynamics 365 for Finance and Operations version 8.0
Microsoft Dynamics 365 for Finance and Operations, Enterprise edition	7.3	7.3.11971.56116	December 2017	Known issue in Dynamics 365 for Finance and Operations 7.3
Microsoft Dynamics 365 for Finance and Operations, Enterprise edition	July 2017	7.2.11792.56024	June 2017	Known issue in Dynamics 365 for Finance and Operations (July 2017)
Microsoft Dynamics 365 for Operations	1611	7.1.1541.3036	November 2016	Known issue in Dynamics 365 for Operations version 1611 with platform update 3
Microsoft Dynamics AX	7.0.1	7.0.1265.23014	May 2016	Known issue in May 2016 release
Microsoft Dynamics AX	7.0	7.0.1265.3015	February 2016	Known issue in February 2016 release

Platform releases

RELEASE	VERSION	BUILD NUMBER	AVAILABILITY	SEARCH FOR THIS TEXT
Microsoft Dynamics 365 for Finance and Operations, Enterprise edition	Platform update 15	7.0.4839	March 2018	Known issue in Dynamics 365 for Finance and Operations, Enterprise edition platform update 15 (March 2018)
Microsoft Dynamics 365 for Finance and Operations, Enterprise edition	Platform update 12	7.0.4709	November 2017	Known issue in Dynamics 365 for Finance and Operations, Enterprise edition platform update 12 (November 2017)
Microsoft Dynamics 365 for Finance and Operations, Enterprise edition	Platform update 11	7.0.4679.35176	October 2017	Known issue in Dynamics 365 for Finance and Operations, Enterprise edition platform update 11 (September 2017)
Microsoft Dynamics 365 for Finance and Operations, Enterprise edition	Platform update 10	7.0.4641.16233	August 2017	Known issue in Dynamics 365 for Finance and Operations, Enterprise edition platform update 10 (August 2017)
Microsoft Dynamics 365 for Finance and Operations, Enterprise edition	Platform update 9	7.0.4612.35162	July 2017	Known issue in Dynamics 365 for Finance and Operations, Enterprise edition platform update 9 (July 2017)
Microsoft Dynamics 365 for Finance and Operations, Enterprise edition	Platform update 8	7.0.4565.16212	June 2017	Known issue in Dynamics 365 for Finance and Operations, Enterprise edition platform update 8 (June 2017)
Microsoft Dynamics 365 for Operations	Platform update 7	7.0.4542.16189	May 2017	Known issue in Dynamics 365 for Operations platform update 7 (May 2017)
Dynamics 365 for Operations	Platform update 6	7.0.4509.16180	April 2017	Known issue in Dynamics 365 for Operations platform update 6 (April 2017)

RELEASE	VERSION	BUILD NUMBER	AVAILABILITY	SEARCH FOR THIS TEXT
Dynamics 365 for Operations	Platform update 5	7.0.4475.16165	March 2017	Known issue in Dynamics 365 for Operations platform update 5 (March 2017)
Dynamics 365 for Operations	Platform update 4	7.0.4425.16161	February 2017	Known issue in Dynamics 365 for Operations platform update 4 (February 2017)
Dynamics 365 for Operations	Platform update 3	7.0.4307.16141	November 2016	Known issue in Dynamics 365 for Operations version 1611 with platform update 3
Microsoft Dynamics AX	Platform update 2	7.0.4230.16130	August 2016	Known issue in Dynamics AX platform version 7.2 (August 2016)
Microsoft Dynamics AX	Platform update 1	7.0.4127.16103	May 2016	Known issue in May 2016 release
Microsoft Dynamics AX	7.0	7.0.4030.16079	February 2016	Known issue in February 2016 release

Removed or deprecated features in Lifecycle Services (LCS)

2/18/2021 • 2 minutes to read • Edit Online

This topic describes features that have been removed or deprecated for Microsoft Dynamics Lifecycle Services (LCS).

- A *removed* feature is no longer available in the service.
- A *deprecated* feature isn't in active development and might be removed in a future update.

This list is provided so that you can consider these removals and deprecations as you do your own planning.

October 2019 announcements

Flowchart diagrams in Business process modeler

Reason for deprecation/removal	We are deprecating the flowchart diagrams component in Business process modeler (BPM), because the legacy design caused low usage.
Replaced by another feature?	No
Areas affected	Business process modeler
Status	Deprecated: The flowchart diagrams component in BPM is expected to be removed in 2020. The following functionality will be unavailable: • All flowcharts will be read-only and unavailable for editing. The shape properties that are associated with flowchart activities will also be unavailable. These flowcharts include both the default flowcharts that are automatically generated and customized flowcharts that are modified based on those default flowcharts. • The process steps will be read-only and unavailable for editing. • The legacy fit/gap analysis feature will be unavailable. Therefore, no gap list will be automatically created or available for export. **Note:** This feature had previously been deprecated and replaced by Microsoft Azure DevOps integrations. • The version history of the flowchart will be unavailable.

Lifecycle Services (LCS) user guide

2/18/2021 • 4 minutes to read • Edit Online

Microsoft Dynamics Lifecycle Services (LCS) provides regularly updated services. The goal of LCS is to deliver the right information, at the right time, to the right people, and to help ensure repeatable, predictable success with each roll-out of an implementation, update, or upgrade. LCS is available to customers and partners as part of their support plans. If you're a Microsoft Dynamics AX 2012 customer, you can sign in by using your CustomerSource or PartnerSource credentials. If you're a customer of the newest version of the Dynamics 365 Finance and Operations apps, you can sign in by using your Microsoft Azure Active Directory (Azure AD) credentials. Go to LCS.

Tools that are provided in LCS

The following table lists the tools that are provided in LCS and describes the phases that each tool applies to.

TOOL	DESCRIPTION
Projects	Projects are the key organizer for your experience in LCS. Projects let you invite your partners to collaborate with you, and they also let you track progress.
Methodologies	Methodologies provide a tool that you can use to ensure more repeatable, predictable implementation projects. You can use one of our methodologies or create your own. By using a methodology, you can easily track and report on your progress.
Business process modeler	Business process modeler lets you create, view, and modify standard process flows. By using Business process modeler, you can achieve the following goals: standardize process flows; align your business processes with industry-standard processes, as described by the American Productivity & Quality Center (APQC); identify fit and gaps between user requirements and the default functionality that Microsoft Dynamics products provides.
Cloud-hosted environments	Cloud-hosted environments is a tool that you can use to deploy Microsoft Dynamics environments on Azure. When you use Cloud-hosted environments, you must select the type of environment to deploy, such as a demo, developer/test, or production environment. Based on your selection, the Cloud-hosted environments tool provisions the appropriate number of virtual machines (VMs) in Azure. These VMs have Microsoft Dynamics components (and all their prerequisites) already installed on them.
Cloud-powered support	Cloud-powered support helps you manage support incidents. It lets you create a VM in Azure that has the same hotfixes installed as your local environment. You can reproduce and record an incident on the VM, and then submit the incident to our support team. Support follows up by investigating and, if possible, testing a fix on the VM, and then sends the fix back to you for verification.

TOOL	DESCRIPTION
Configuration and data manager (preview)	Configuration and data manager (preview) lets you copy a configuration from one instance to another. You can copy from and to environments that meet the following criteria: they are managed as part of an LCS project; they run the Data Import/Export Framework.
Customization analysis	Customization analysis validates model files against best practices and provides a report of potential areas for improvement.
Issue search	Issue search helps you find existing solutions and workarounds for known issues in Microsoft Dynamics products. You can see which issues have been fixed, which issues remain open, and which issues have been resolved as "won't fix."
Asset library	The Asset library is a storage location for the various assets that are associated with a tenant in LCS.
Get updates	Get updates is a tool that customers to access the updates that are available for their environments.
Environment monitoring	Environment monitoring is a set of tools that helps you monitor, diagnose, and analyze the health of the environments that you manage.
Translation service	The Microsoft Dynamics 365 Translation Service (DTS) is hosted in LCS. It's designed to enhance the experience for partners and independent software vendors (ISVs) when they translate their solutions or add a new language for the supported Dynamics products.
Regulatory updates	Regulatory updates is a tool that lets customers, partners, and ISV solution providers stay up to date about regulatory updates by setting up alerts through LCS.
System diagnostics	System diagnostics is a tool that helps administrators monitor AX 2012 environments.
Updates	The **Updates** page hosts the details of updates that are available for an AX 2012 environment. It also provides access to groups of updates that can be used for slipstream installations.
Upgrade analysis	Upgrade analysis helps you plan your upgrade to the latest version by analyzing code artifacts from Microsoft Dynamics AX 4.0, Microsoft Dynamics AX 2009, or AX 2012.
Usage profiler	Usage profiler is a data-gathering tool that helps you describe your projected or current usage of an. The usage profile that is generated can be used for various purposes, such as hardware sizing and support.
Downloadable tools	The **Downloadable tools** page provides a set of tools that were previously hosted on Microsoft Dynamics InformationSource.

TOOL	DESCRIPTION
License sizing estimator	License sizing estimator helps you estimate the number of licenses that are required. It provides a shared workspace that lets you model default and customized roles, and then automatically calculate the required client access licenses (CALs).

Additional resources

The LCS team is also blogging on the Lifecycle Services Engineering blog. Subscribe to our RSS feed to keep up with our posts and announcements.

Access Lifecycle Services

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Project onboarding

2/18/2021 • 5 minutes to read • Edit Online

Project onboarding is a self-paced, wizard-driven onboarding experience that guides project users in Microsoft Dynamics Lifecycle Services (LCS) through the process of setting up the key configuration components for a new implementation project for Dynamic 365 Finance, Dynamics 365 Supply Chain Management, or Dynamic 365 Retail. This wizard can also be accessed during and after the implementation, and can be used to update the information as required.

Microsoft relies on the information that you provide. You must provide the most current and accurate data as you complete Project onboarding. After you complete Project onboarding, you can deploy environments and continue with the project implementation.

To access Project onboarding, sign in to LCS, and then, on the main menu, select **Project onboarding**.

> **NOTE**
> Project onboarding is available for implementation projects and must be completed before any of the Microsoft-managed environments are deployed. For more information about implementation projects, see Lifecycle Services (LCS) for Finance and Operations apps customers.

For more information about the onboarding process, see Onboard an implementation project, and watch the Finance and Operations: Onboarding to Dynamics 365 TechTalk.

Onboarding steps

Each step in Project onboarding is designed to give you guidance about the project implementation or to gather information about the project context, so that the FastTrack team can better serve you. By providing accurate information in the wizard, you help Microsoft understand your implementation plan, so that it can provide appropriate guidance.

You can move through the wizard either by using the **Next** and **Back** buttons, or by selecting each step directly. For some steps, the right column of the page shows additional contextual information that will help you complete the step.

Welcome

The **Welcome** page provides general guidance and information that you will need to complete Project onboarding.

Project overview

- Provide the overview information for the implementation project.
- Describe the vision and goals for the project in a few sentences. This information will help Microsoft understand the goals that you want to achieve and how you define success for the project.
- Provide the Partner MPN ID, which you can get from the implementation partner team. If a partner is not involved or not yet identified, choose the appropriate option in the implementation partner drop-down list. Note that providing accurate partner data is a pre-requisite for FastTrack Program assignment. You could miss the opportunity for valuable services if you do not provide the correct partner information. Once partner identified, you need to update the MPN ID.

- Specify the estimated number of user licenses after full roll-out including current licenses. This number can differ from the current license purchase. If no change is planned, provide the current user license count. If a license type isn't applicable, enter **0** (zero).
- If your implementation project is a demo project, or if you're moving from another tenant, provide the details.

Project scope

- Provide information about how you plan to scope the implementation in terms of features and products. This information will help Microsoft understand your implementation and provide any guidance that is required.
- After you set the option for specific features to **Yes** in this step, you must provide additional data that is related to those features. This data will help Microsoft have a better engagement with you during the implementation. The additional data fields are mandatory.

Define your team

- Verify that all project team members are invited and configured.
- Set the **Primary contact for FastTrack** option to **Yes** for at least two users who have an active email address in the user list. If this option isn't set to **Yes** for any team member, FastTrack will reach out to all team members for implementation guidance during your implementation. If necessary, you should nominate at least one customer and one partner team member to be contacted by FastTrack.
- Each team member will be assigned a project security role and an implementation role. The project security role is relevant to access to the LCS project workspace, and the implementation role is relevant to the individual team member's role on the implementation team. We highly recommend that you include representatives from the customer among the project team members who have a monitored email address.

For more information, see Configuring project security and Roles in a Dynamics 365 implementation.

Define milestone dates

- Define all mandatory milestone steps. Milestones are associated with the methodology of the project. If the milestone dates haven't yet been decided, use tentative dates.
- Update the milestone dates as plans change. Microsoft uses the milestone dates to provide appropriate guidance for each milestone. To edit milestone dates, select the pencil symbol.

Associate LCS with Azure DevOps

- Connect LCS and Azure DevOps to maintain the application lifecycle.
- Enter the root URL of your Azure DevOps account and the personal access token that you obtained from Azure DevOps. The Azure DevOps account should belong to the customer.

Configure Azure DevOps

- Map work items between LCS and Business process modeler (BPM).
- Acknowledge the setup.
- If you choose to use a custom process template, follow these best practices:
    - Don't remove any existing work item types.
    - Don't remove any existing state for a work item type.
    - Don't add any required fields to a work item type.

FastTrack

- The **FastTrack** page introduces the FastTrack program. It explains what the program consists of and how your implementation can benefit from it.
- We strongly recommend that you watch the existing TechTalks and subscribe to upcoming TechTalks as Microsoft continues to share best practice guidance and information about the changes that are occurring in the product and platform.

Next steps

The **Next Steps** page provides additional resources about the most critical aspects of the implementation. You can access this page at any time during the implementation.

Complete onboarding

- Complete Project onboarding so that you can move on to the next steps in your implementation.
- You can complete Project onboarding only after all previous steps have been completed. If any previous steps haven't been completed, the **Complete onboarding** button won't be available.
- Any skipped steps are marked with an asterisk (*). You can also view any missing steps.
- After you complete Project onboarding, you can continue to update information, such as project scope values.

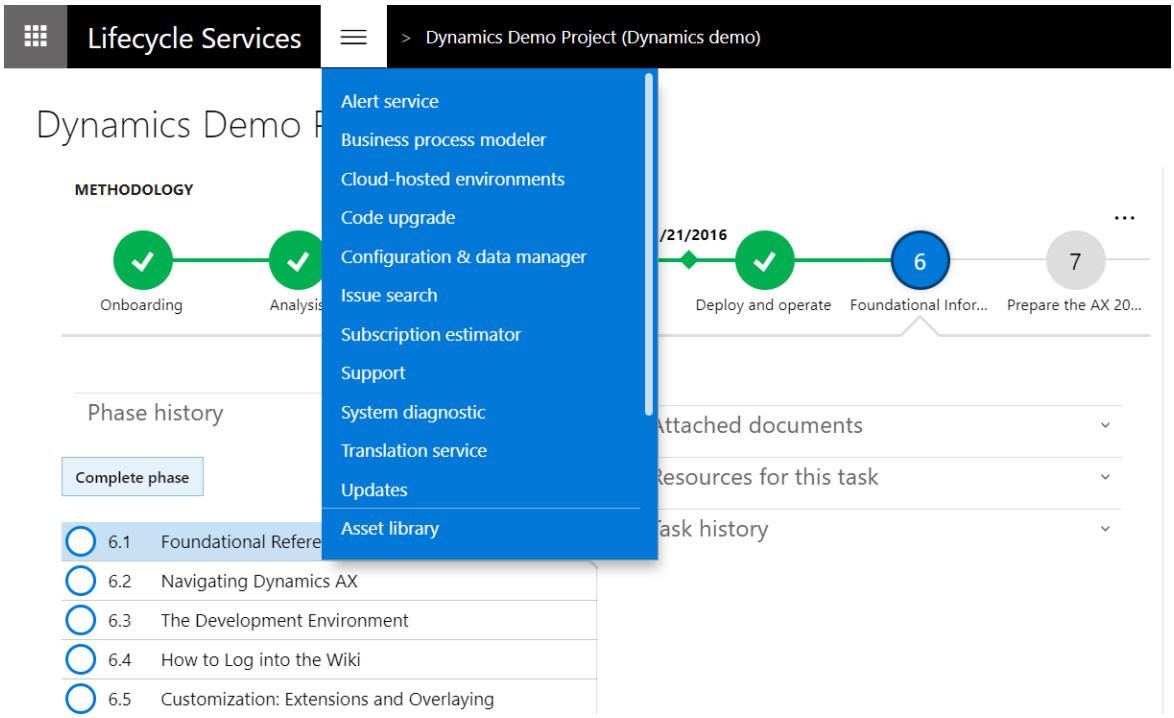
Subscription estimator in Lifecycle Services (LCS)

2/18/2021 • 2 minutes to read • Edit Online

Subscription estimator is a tool that is available in Microsoft Dynamics Lifecycle Services (LCS). Microsoft uses this tool to estimate the initial size of the production environment that must be provisioned for a customer. Before customers can request deployment of a production environment, they must estimate their peak workloads in terms of transaction counts and then upload that information to LCS. By using the details of user licenses and transaction counts to infer subscription requirements, the Subscription estimator tool helps ensure that the provisioned environment meets the customer's business requirements.

Follow these steps to use the Subscription estimator tool.

1. In LCS, open the project that is associated with the implementation project.
2. At the top of the page, select the hamburger icon, and then select **Subscription estimator**.



3. Download the sample usage profile.
4. Answer the required questions on each tab. If you're a Commerce customer, be sure to answer the questions on the **Retail and Commerce** tab.
5. Save the usage profile locally.
6. To upload the usage profile, select **New estimate**, name the estimate, and then upload the usage profile.
7. After the upload is completed, select **Mark as Active** to activate an estimate. An active estimate is required in order to configure a production deployment.

When there is a valid active estimate, the **Configure** button becomes available. You can use this button to request a production environment deployment.

NOTE

Although you can have multiple estimates, one estimate must be marked as **Active**. After the production environment has been deployed, or deployment of the environment has received sign-off, the active estimate is locked. To mark a different estimate as the active estimate, create a support request by using the Support portal in LCS.

Frequently asked questions

Question: Why isn't the **Configure** button for deploying a production environment available, even though there is an active estimate? And why does a warning message appear in the Action center on the project dashboard?

Answer: If you have multiple implementation projects, the **Configure** button might not be enabled and a warning message will appear in the Action center regarding an insufficient number of licenses. Log a support request, and the support team can help resolve this issue.

Question: Why does an error occur when I mark an estimate as **Active**?

Answer: When you mark an estimate as **Active**, you might receive the following error message:

- **Estimate created but does not meet requirements** – This error occurs if transaction lines that are entered aren't within the limits of the Subscription estimation tool. To resolve this error, create a support request, and attach the usage profile. Your instance can then be manually sized.

If you receive any other error message or encounter any other issue, create a support request, and attach your active estimate so that the support team can address the issue.

Additional resources

Subscriptions, LCS projects, and Azure Active Directory tenants FAQ

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Configure Lifecycle Services (LCS) security

2/18/2021 • 3 minutes to read • Edit Online

Security in Microsoft Dynamics Lifecycle Services (LCS) is controlled at both the organization level and the project level. Not all members of an organization have access to all projects. Additionally, the members of a project might not all be members of the same organization.

Currently, users can sign in by using the Microsoft Azure Active Directory (Azure AD) credentials that they created in the Microsoft 365 portal when they signed up. Users who are administrators for their organization in Azure AD will be administrators in Lifecycle Services (LCS).

For Microsoft Dynamics AX 2012, organization-level access to LCS is controlled by the association of a person's Microsoft ID with an organization in CustomerSource or PartnerSource. Therefore, users of CustomerSource or PartnerSource automatically have access to their organization's workspace in LCS, and can view all projects that they have been invited to participate in. Users who are administrators for their organization in CustomerSource and PartnerSource will be administrators in LCS.

Although an administrator can invite users who don't have CustomerSource or PartnerSource credentials to be members of an organization in LCS, we don't recommend this approach. Users who are invited to be members of an LCS organization aren't provided with credentials in CustomerSource or PartnerSource.

Project-level access to LCS is by invitation. You can invite members of your organization to be project owners and team members. Additionally, you can invite users who aren't part of your organization, and who don't have accounts in Azure AD or CustomerSource or PartnerSource, to be team members.

> **IMPORTANT**
> We strongly recommend that you manage all users within your company at the organization level. In that way, you help ensure that their relationship with your organization is correct in CustomerSource, PartnerSource, and Azure AD. Additionally, you help ensure that users can access the benefits that are available to your organization.

Manage LCS organization users

Only an administrator can manage users. Follow these steps.

1. In CustomerSource, PartnerSource, or Azure AD, associate all the users in your organization who require access to LCS with your organization. Users might have to wait up to two business days before they can sign in to LCS.
2. Add your users to the appropriate projects in LCS.

Invite a user to an LCS project

1. Sign in to LCS.
2. Select the project to add the user to.
3. Select the **Project users** tile, and then, on the **Project users** page, select the plus sign (+).
4. Enter the user's email address, select the correct security role, and then select **Invite**.

NOTE

For implementation projects, you can select the implementation role for the invited user. If you set **Allow FastTrack to contact** to **Yes**, then Microsoft FastTrack team may reach out to you based on your implementation role and the stages of the implementation project.

Configuring project security

You can invite users from inside or outside your organization to join your project as users. The following table describes the roles that are available for users.

ROLE	DESCRIPTION
Project owner	Members of this role have access to all tools in LCS, can add other users in any role, and can delete the project.
Environment manager	Members of this role have access to all tools in LCS and can manage cloud-hosted environments.
Project team member	Members of this role have access to all tools in LCS but can't manage cloud-hosted environments.
Project team member (prospect)	Members of this role have limited access to all tools in an LCS project. Prospects are users who have been added to a project, but who don't have an account in VOICE or an Azure AD account. You can identify that a user is a prospect, because **prospect** is listed as the organization.
Operations user	Members of this role have access to the following tools in LCS: • System diagnostics • Issue search • Cloud-powered support • Updates • Cloud-hosted environments

After you've configured security for one project, you can import the users to another project.

Configure implementation roles

If you have an implementation project, you will have the option to specify project user's implementation roles. For more information, see Roles in a Dynamics 365 implementation.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Issue search in Lifecycle Services (LCS)

2/18/2021 • 3 minutes to read • Edit Online

This article provides information about the Issue search tool on Microsoft Dynamics Lifecycle Services (LCS). It explains how to search for product issues and regulatory features, and describes the information that is provided for each status.

Prerequisites

None

Search for product issues and regulatory features

You can use Issue search to search for product issues, and determine whether an issue has been resolved, is open, or has a workaround. You can also search for regulatory features, and determine whether a feature is available or is planned in a future release. Finally, you can find regulatory white papers, certifications, and registrations.

1. Go to Microsoft Dynamics Lifecycle Services (LCS).
2. Select a project to work in.
3. Click the **Issue search** tile.
4. Enter search terms. You can enter a keyword or group of keywords, or a Microsoft Knowledge Base (KB) number. You can also use a dollar sign ($) to indicate an Application Object Tree (AOT) object path in the format $*ObjectType**Object* or $\\*ObjectType*\\*Object*#*Method* (for example, **$\Classes\Tax#Save**). Standard search operators such as **AND** and **OR** are supported.

You can filter the results list for resolved or open issues, workarounds, and issues that won't be fixed or are postponed. You can also filter by the application version. By default, the results are sorted by relevance. However, you can sort by date ascending, date descending, version ascending, or version descending instead. By default, all status and product version filters are selected. **Note:** Results for Microsoft Dynamics AX 2009 are included in Microsoft Dynamics AX 2012 projects only when you search for regulatory features. The following table describes the information that is provided for each status when you search by product issue.

STATUS	COLOR	DESCRIPTION OF RESULTS

STATUS	COLOR	DESCRIPTION OF RESULTS
Resolved	Green	The KB number, title, version, description of the problem and change, and hotfix are provided. A list of objects that are affected by the hotfix is also provided whenever possible. • To download the hotfix, click **Download hotfix**. • To determine what code was changed in a hotfix, click **View changes**. Added code is indicated by a green highlight, and deleted code is indicated by a red highlight. **Important:** The list of code changes is provided for reference only. Code changes should not be manually inserted into a higher development layer. Otherwise, they become unsupported customized objects that the partner or customer assumes full responsibility for.
Open	Red	A bug number, title, version, and description of the problem are provided.
Workarounds	Brown	The issue number, title, version, problem description, and mitigation are provided.
Postponed	Gray	A bug number, title, version, and description of the problem are provided. If a workaround is available, it's described. A bug that has this status has been evaluated and won't be fixed at this time.
By design	Gray	A bug number, title, version, and description of the problem are provided. An explanation of the design is provided whenever possible. A bug that has this status has been evaluated, and it has been determined that this functionality is working as designed.
Not reproducible	Gray	A bug number, title, version, and description of the problem are provided. An explanation of the problem is provided whenever possible. A bug that has this status has been evaluated, and a fix won't be issued at this time.

STATUS	COLOR	DESCRIPTION OF RESULTS
Will not be fixed	Gray	A bug number, title, version, and description of the problem are provided. If a workaround is available, it's described. A bug that has this status has been evaluated and won't be fixed.

The following table describes the information that is provided for each status when you search by regulatory feature.

STATUS	COLOR	DESCRIPTION OF RESULTS
Resolved	Green	For feature updates: The KB number, title, version, description of the problem, and hotfix are provided. A list of objects that are affected by the hotfix is also provided whenever possible. • To download the hotfix, click **Download hotfix**. • To read the KB article for the hotfix, click **Read KB article**. For white papers, certifications, registrations, and reports: The title, product version, and description of the white paper, certification, registration, or report are provided. • To read the documentation, click **Read documentation**.
Open	Red	A bug number, title, version, planned release date, and description of the problem are provided.

Issue details

The code changes are provided for reference only and should not be manually inserted into a higher development layer. Otherwise, they become unsupported customized objects that the partner or customer assumes full responsibility for.

Configuration in Lifecycle Services overview

2/18/2021 • 2 minutes to read • Edit Online

You can use the Configuration manager to copy from and to Dynamics AX 2012 R3 environments that meet the following criteria:

- Managed as part of a Lifecycle Services project
- Running System diagnostics
- Running the Data Import/Export Framework

> **IMPORTANT**
>
> This feature is **not** supported for production use. Configuration manager (beta) relies on entities from the Data Import/Export Framework in your environment. Because these entities do not currently include all the functionality in AX 2012 R3, some configuration data is not copied between environments.

For more information, see:

- Set up Configuration manager
- Copy configurations by using Configuration manager

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Set up Configuration manager

2/18/2021 • 3 minutes to read • Edit Online

> **IMPORTANT**
>
> This feature is **not** supported for production use. Configuration manager (beta) relies on entities from the Data Import/Export Framework in your environment. Because these entities do not currently include all the functionality in AX 2012 R3, some configuration data is not copied between environments.

Before you begin

Before you begin, your environment must include the following components:

- A running version of AX 2012 R3 that has been configured for your business. For more information about how to install AX 2012 R3, see Install Microsoft Dynamics AX 2012.
- A running instance of the Data Import/Export Framework. For more information about how to install the Data Import/Export Framework, see Install the Data import/export framework (AX 2012 R#).

  > **IMPORTANT**
  >
  > You must deploy the DMFEntityExecutionStatusService and DMFService service groups to enable to Configuration manager (beta) to connect to Data Import/Export Framework.

- An AX 2012 R3 project in Lifecycle Services.

  > **WARNING**
  >
  > Copying configurations between environments can be a destructive operation. All project owners have the right to configure and perform these operations. Make sure that only trusted individuals are set as project owners.

Create Data Import/Export Framework source data formats in AX 2012 R3

You must create both a Dynamics AX and a CSV source data format to manage configurations in all environments where you intend to export and import configurations.

Create an AX source data format

Complete the following procedure in the environment that you intend to export a configuration from.

1. Click **Data import export framework** > **Setup** > **Source data formats**.
2. Click **New**, and name the source AX.
3. Verify that the type is **AX**.
4. Repeat this procedure in the environment that you intend to import the configuration to.

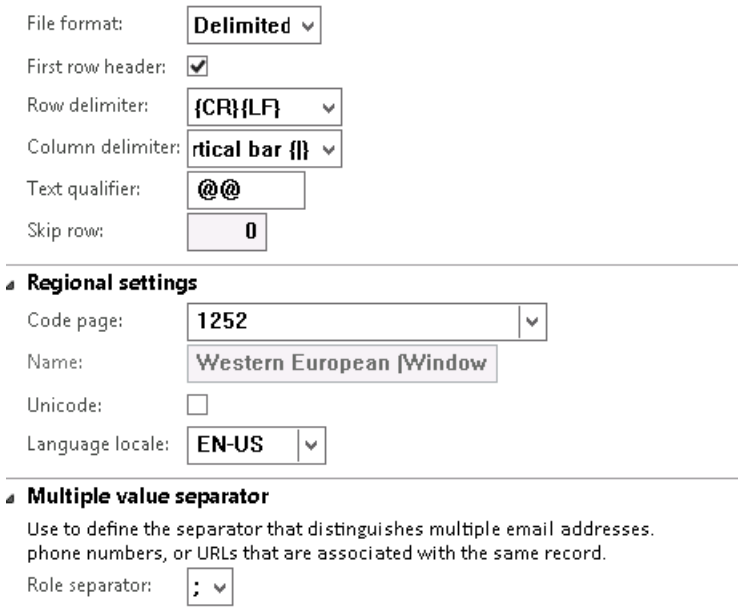
Create a CSV source data format

Complete the following procedure in the environment that you intend to export a configuration from.

1. Click **Data import export framework** > **Setup** > **Source data formats**.

2. Click **New**, and name the source CSV.
3. Verify that the type is **File**.
4. On the **General** tab, set the following values.

SETTING	VALUE
File format	Delimited
First row header	Selected
Row delimiter	{CR}{LF}
Column delimiter	Vertical bar {
Text qualifier	@@
Skip row	0
Code page	1252
Language locale	EN-US
Multiple value separator	;



5. Repeat this procedure in the environment that you intend to import the configuration to.

Install and configure the local component of the System diagnostics (Lifecycle Services)

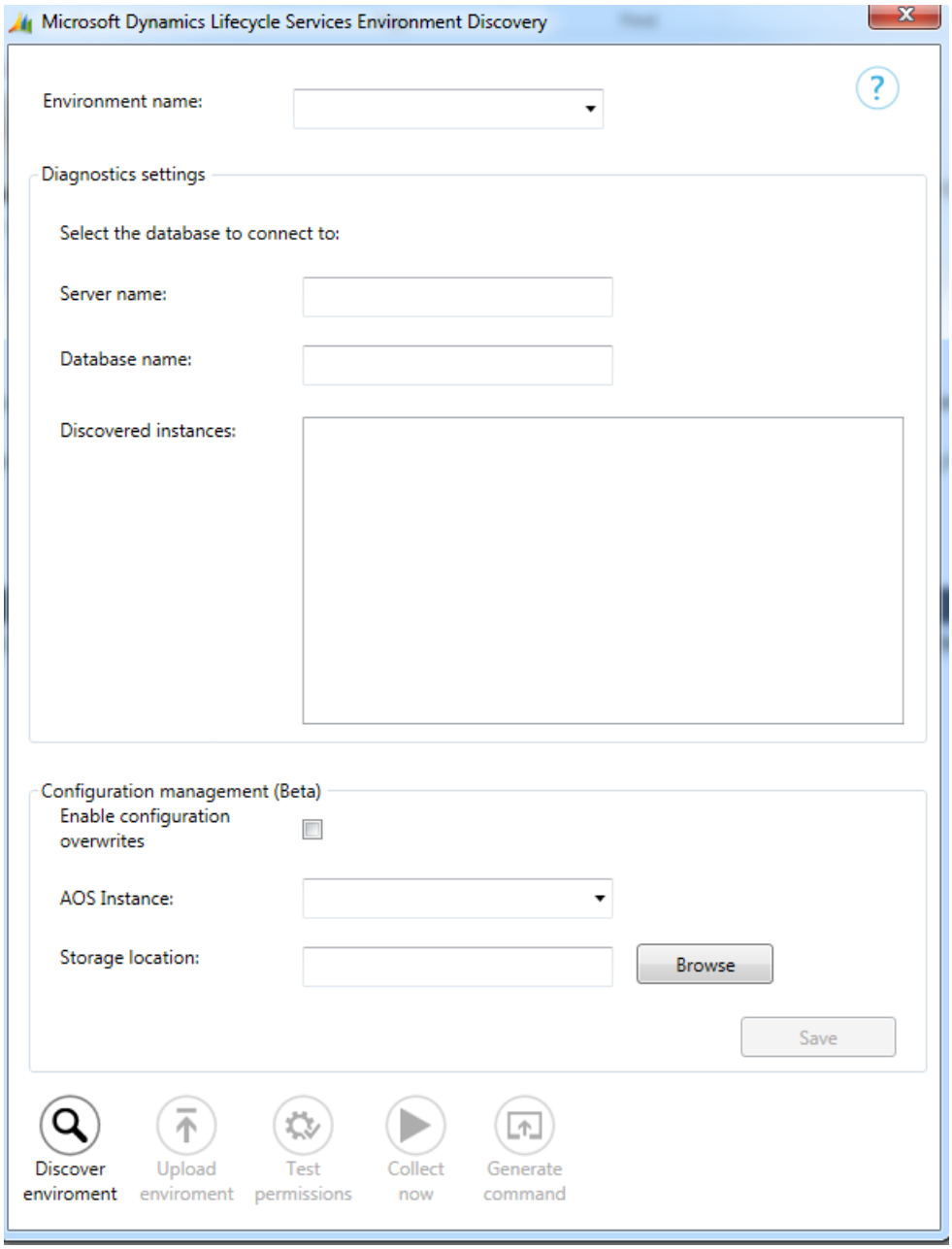
Complete the following procedure in the environment that you intend to export a configuration from.

IMPORTANT

If you have already installed the local component of the System diagnostics for your project and environment, you must uninstall it by using **Add/Remove programs**. Only one instance of Microsoft Dynamics AX Application Object Server (AOS) per environment can be used with Configuration management, regardless of the number of instances that are discovered.

1. Install the local component of the System diagnostics. For details, see Install and run System diagnostics. Important: For this beta release, we require that you add the service account for the System diagnostics to the sysadmin role in AX 2012 R3.
2. Click **Start** > **Microsoft Dynamics AX Lifecycle Services Diagnostic Service Discovery**.
3. In the **Environment Discovery** window, enter a name for the environment, and the fully-qualified name of the Microsoft SQL Server instance and database. Then click Discover environment.
4. After discovery is completed, enter the values in the **Configuration management (Beta)** section, click **Save**, and then click **Upload environment**.

OPTION	VALUE
Enable configuration overwrites	Select this option to enable configurations to be copied and overwritten for the specified AOS instance. **Important:** We strongly recommend that you disable configuration overwrites when you have fully configured an environment.
AOS instance	Specify the AOS instance that can be copied from or overwritten.
Storage location	Specify the location where configurations are stored locally. The AOS service account and the Data Import/Export Framework service account must have read and write access to this location. **Important:** We strongly recommend that you use the same directory that is used for the Data Import/Export Framework.Be aware that the shared directory might contain sensitive data, depending on what you are importing and exporting. Make sure that as few users as possible, in addition to the AOS service account and the Data Import/Export Framework service account, have access to the location.



5. Repeat this procedure in the environment that you intend to import a configuration to.

Next steps

The environment is now ready for you to copy and manage configurations. For more information, see Copy configurations by using Configuration manager.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Configure the code upgrade service in Lifecycle Services (LCS)

2/18/2021 • 5 minutes to read • Edit Online

This topic explains how to configure the **Code upgrade** tile in Lifecycle Services (LCS) to migrate your solution to the latest version of the Dynamics 365 Finance and Operations apps.

Overview

The code upgrade tool operates by connecting to your Azure DevOps project, locating your Trunk\Main branch, branching to a new branch that will be named as Releases\<version number>, and then performing the code upgrade there. After this process is complete, you can synchronize your developer environment to this new branch under Releases\<version number> and resolve conflicts. When you have compiled and tested your upgraded code you can merge the new branch back into Trunk\Main, using source control explorer in Visual Studio and the process is complete.

Dynamics 365 for Finance and Operations version 8.0 and newer, does not allow customization via overlayering of Microsoft models. Before you upgrade, you must have a plan to refactor your customizations into extensions. For more information, see the Extensibility home page and Relax model restrictions to refactor overlayering into extensions.

Process

Create the Trunk\Main folder structure

For the code upgrade service to recognize your source code, your Azure DevOps project must contain a Team Foundation Version Control (TFVC) code repository. In addition, the code repository folder structure must conform to the following strict pattern.

- For code and metadata: //Trunk/Main/Metadata
- For Visual Studio project and solution files: //Trunk/Main/Projects

You can create new folders directly in the Azure DevOps web interface under **Repos**.

> **NOTE**
>
> - Folder names are case sensitive, that is, you must use Main and not MAIN, or the code upgrade service will not recognize the folder.
> - Azure DevOps projects use Git version control by default. You will need to add a TFVC repository.
>   1. Go to Project settings, then Repositories.
>   2. Select New repository.
>   3. In the Type field, select TFVC, and then click Create.

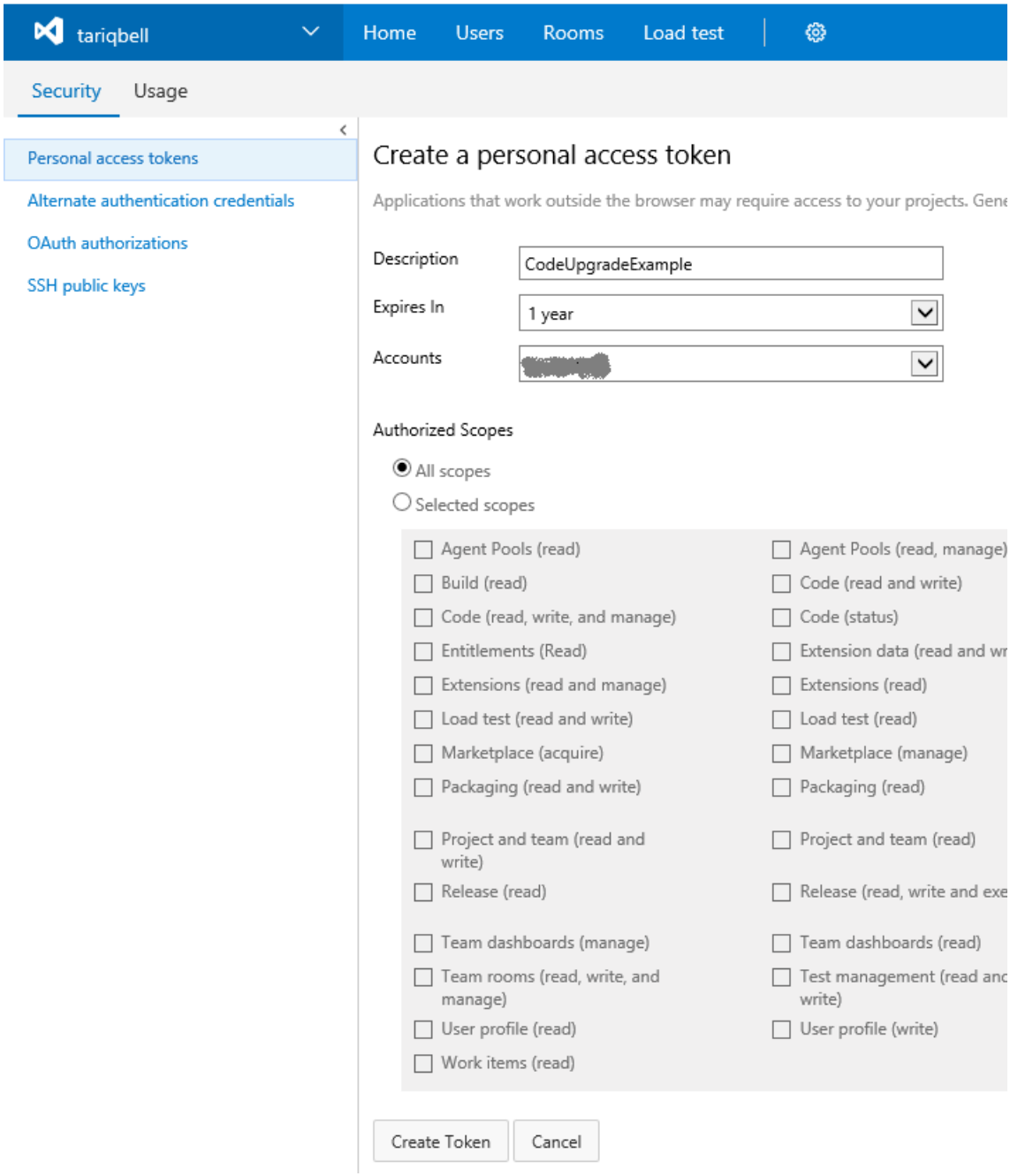
Create a personal access token

To connect to an Azure DevOps project, LCS is authenticated using a personal access token. Use the following steps to create a personal access token in Azure DevOps. If you have already configured your LCS project to connect to your Azure DevOps project, you can skip this section.

1. Sign in to visualstudio.com and locate your Azure DevOps project.

2. In the top-right corner, hover over your name, a menu appears, select **Security**.

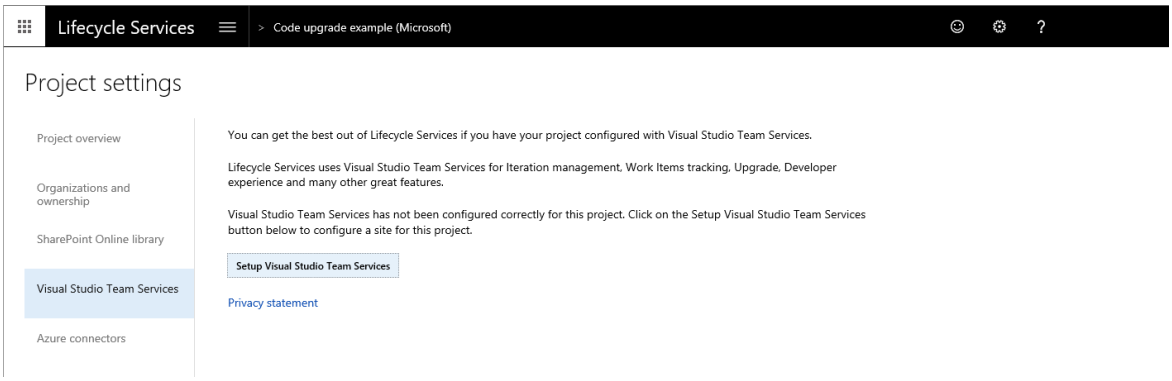
3. Select **Add** to create a new personal access token, give it a name, and then enter the amount of time that you want the token to last for. Select **Create Token**.



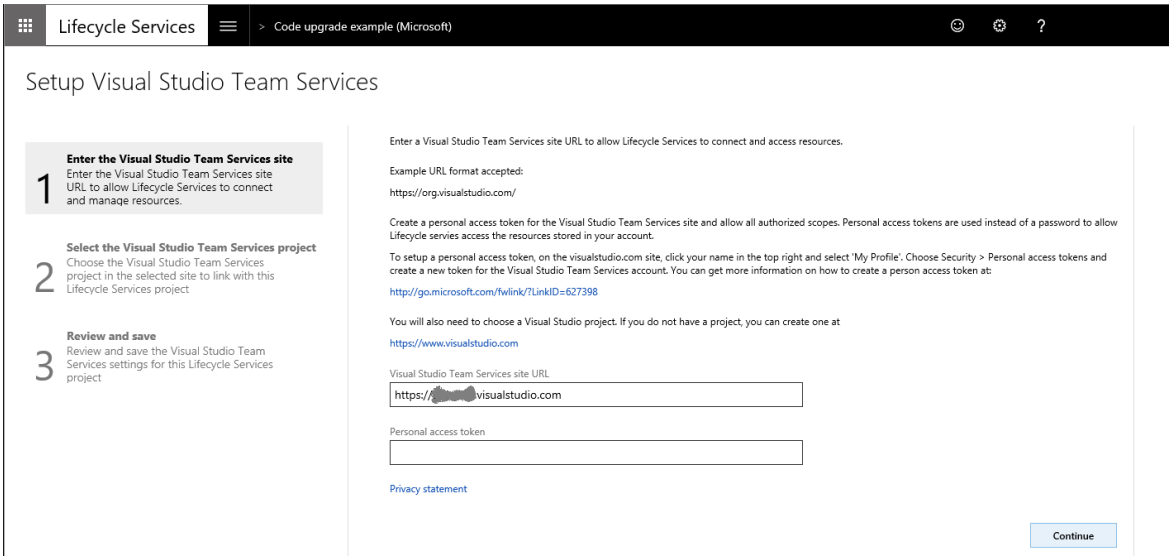
4. Copy the token to your clipboard. You will not be able to find the token details after this step is completed, so be sure that you have copied the token before navigating away from this page.

Configure your Lifecycle Services project to connect to Azure DevOps

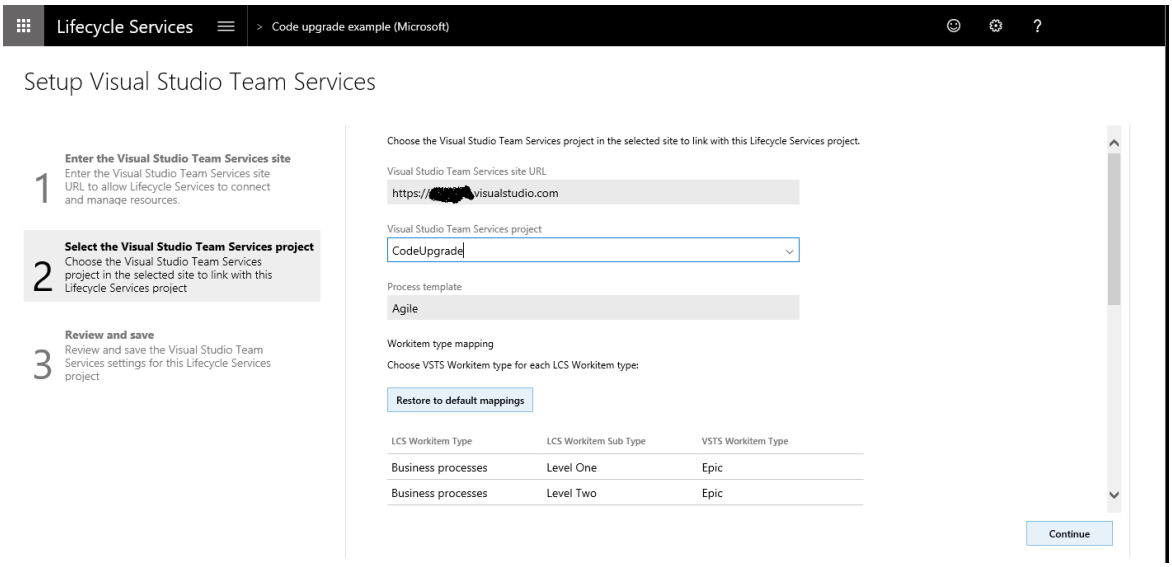
1. In your LCS project, go to the **Project settings** tile, select **Visual Studio Team Services**, and then select the **Setup Visual Studio Team Services** button. This configuration is needed by many LCS tools, if you have already configured LCS to connect to your Azure DevOps project, you can skip this section.



2. Enter the root URL for your Azure DevOps organization and the access token created earlier, and then select **Continue**.



3. Select the project within your Azure DevOps organization that you want to connect to, and select **Continue**.

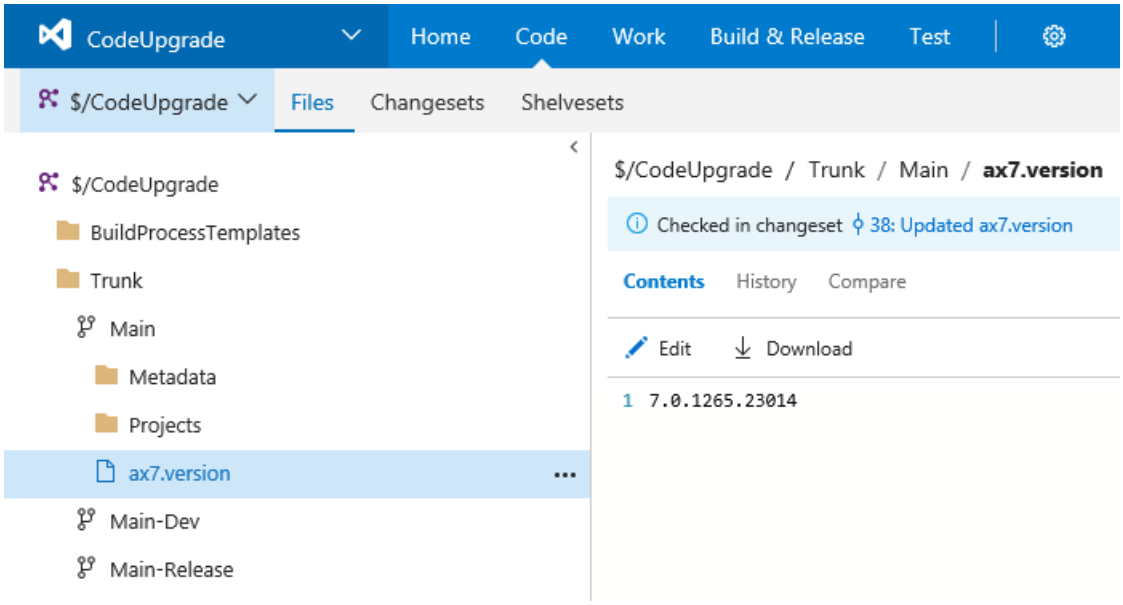


4. On the **Review and save** page, select **Save**.

Create an ax7.version file

> **NOTE**
>
> If you are migrating from AX 2012, you can skip this step.

The code upgrade tile in LCS automatically finds the version that you are migrating from, by reading the ax7.version file under the Main folder in your source control. You must create this file manually, either in Visual Studio or through the Azure DevOps web portal, as shown below. This file is not needed if you are migrating your code from Dynamics AX 2012 R3 or an earlier version. The version number entered here must be the application version (not the platform version). Take care to enter the correct version number here as entering an incorrect version number in this file may cause your code upgrade run to fail.



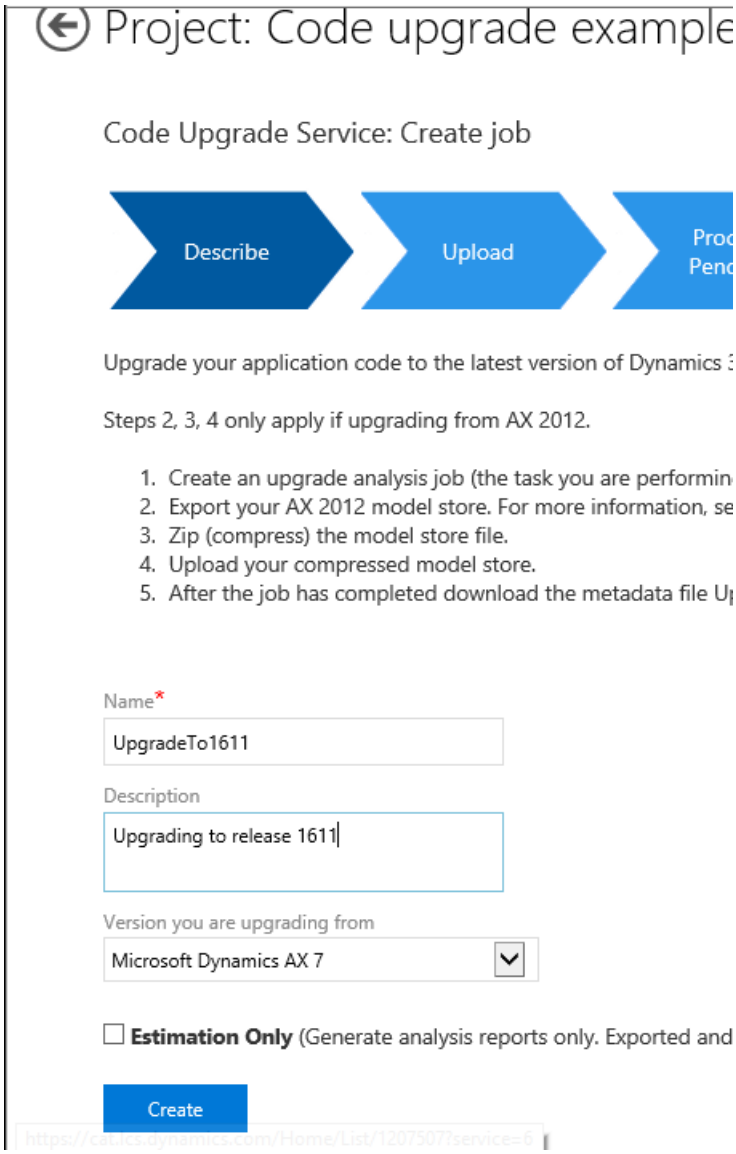
For more information about how to identify which application version you have, see Overview of Microsoft Dynamics AX build numbers.

Execute the code upgrade tile

1. In your LCS project, select the **Code upgrade** tile.

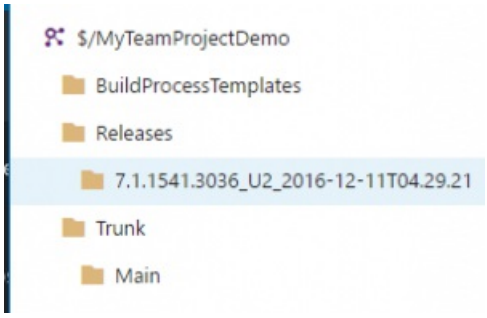
2. In the bottom-left corner of the screen, select **Add**, and then enter a name and description. Select the version you are upgrading from as Microsoft Dynamics AX 7, and then select **Create**.

   - If you are upgrading your code from Dynamics AX 2012 R3, select the version you are upgrading from. You will be prompted to upload a zipped version of your Dynamics AX 2012 R3 model store file.
   - If the **Estimation Only** check box is selected, the tool only generates a report and does not check in or create a new code branch in Azure DevOps for you. You should use this option if you want to evaluate the potential size of the work involved in upgrading before you commit to the actual upgrade.



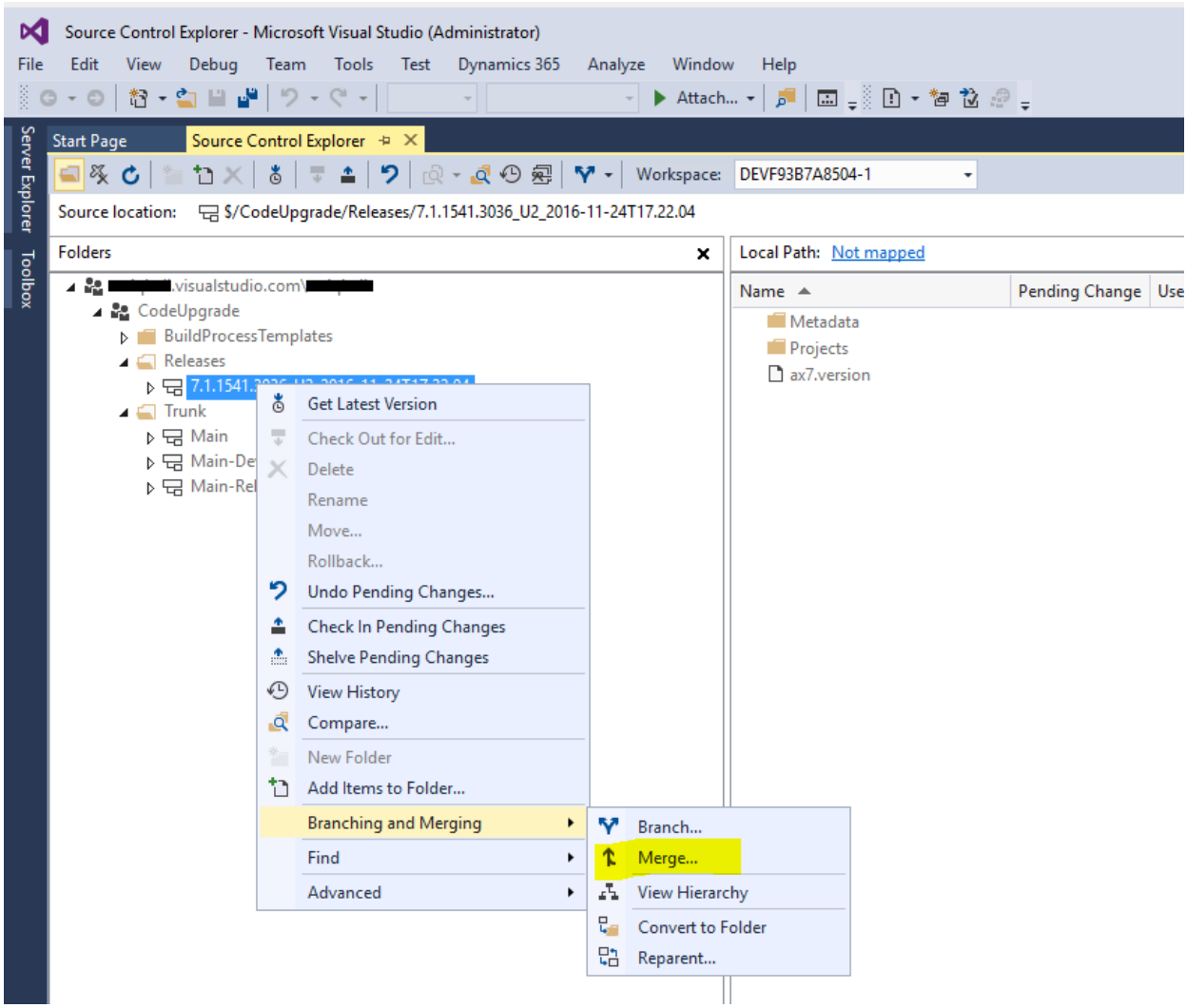
3. Select **Analyze code** in the bottom-right corner. The code upgrade process will start. This process typically takes 40 minutes for a large solution to complete. When complete, return to the **Code upgrade** tile in LCS to view the results.

4. The code upgrade service creates a new branch and checks in the upgraded code to your Azure DevOps project. After the upgrade process is complete, your code will exist in a new branch under the **Releases** folder. The branch name is suffixed with the date and time of the upgrade.



Merge Releases back into Trunk\Main

Once the upgraded code in Releases\<version number> compiles successfully and you have completed your code migration and testing, you are ready to merge this branch back into Trunk\Main. To merge, on your development environment in Visual Studio, open the Source control explorer pane then right-click on the **Releases\<version number>** branch, and in the context menu go to **Branching and Merging**, and then on the submenu select **Merge**.



The Source Control Merge Wizard opens, which guides you through merging the Releases\<version number> branch back into Trunk\Main.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Create or update methodologies

2/18/2021 • 2 minutes to read • Edit Online

Lifecycle Services (LCS) for Microsoft Dynamics provides methodologies that you can use to ensure a more repeatable and predictable implementation project experience. You can use one of the provided methodologies or create your own. With a methodology, you can easily track and report on your progress.

A methodology consists of phases, tasks, and milestones. Each phase can have any number of tasks, some of which are mandatory. When all of the tasks in a phase are completed, the phase can be marked as complete. You can also create a milestone for when you anticipate a phase to be completed. The following methodologies are included in an LCS project:

- Implementation
- Sure Step
- Learn development
- Migrate and create solutions
- Consume solutions

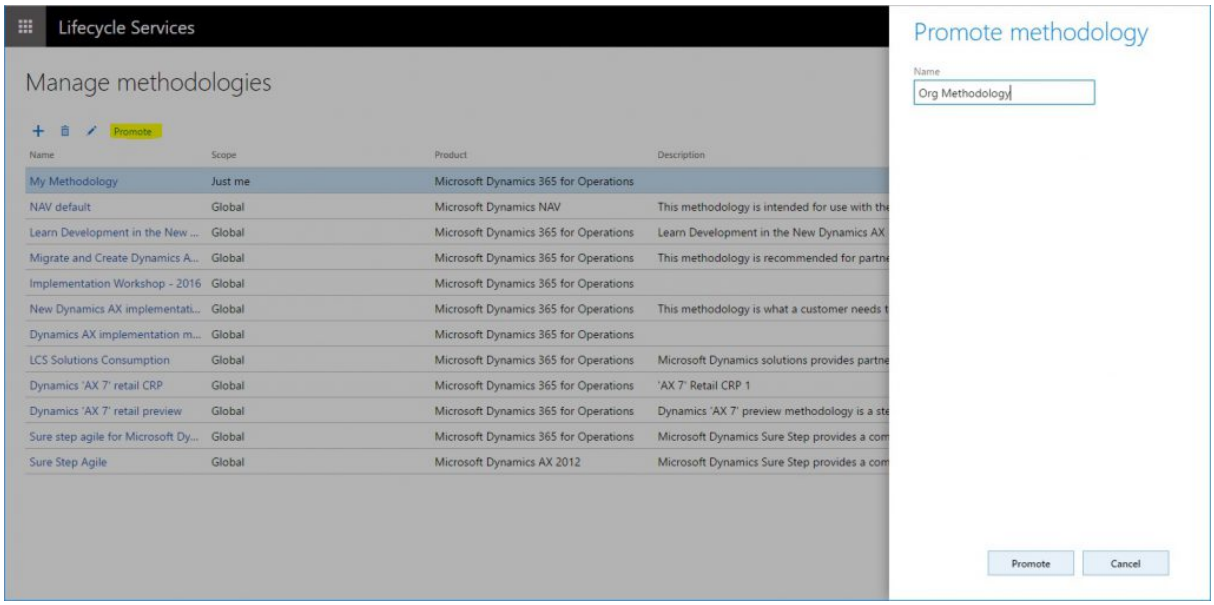
Note: There is a known limitation where new changes that are published to the Microsoft Methodology are not pushed to existing projects. Only new projects get these changes.

Add or update methodologies

A partner or a project administrator can create new methodologies or make changes to an existing methodology for their organization or within the scope of a specific project. These additions and changes can be made at the project level or at the organization level. Use the following procedures to create and save a new methodology, update existing methodologies, and when appropriate, promote a new methodology or methodology changes to the organization level.

Create a new methodology

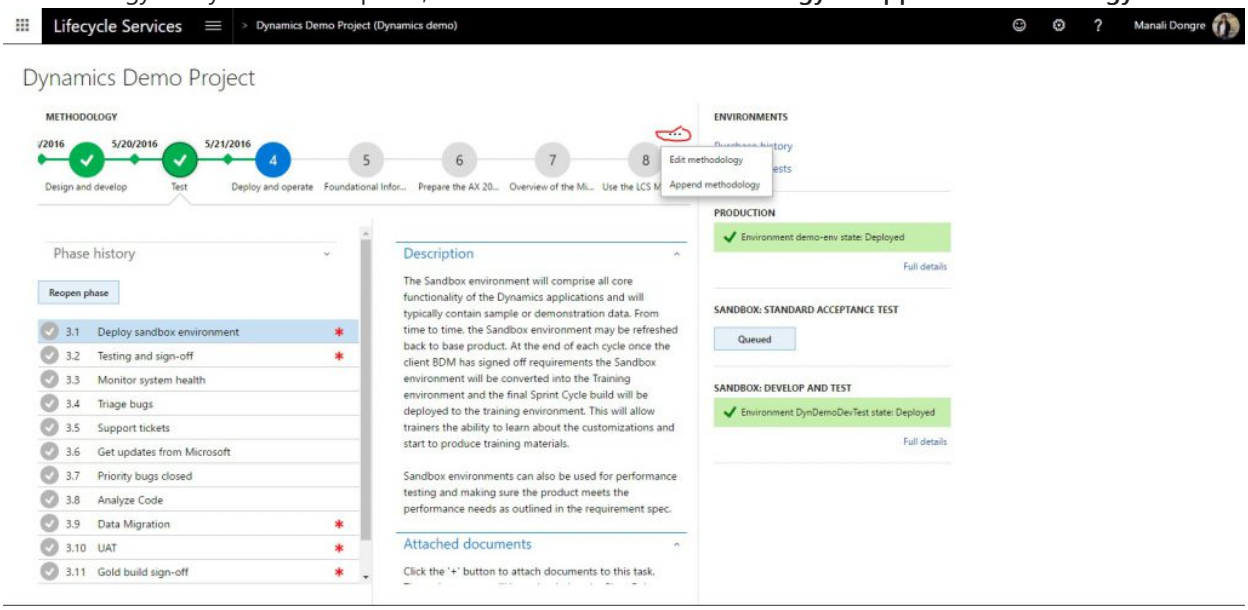
1. On the Lifecycle Services dashboard, on the right side of the screen, click **Manage methodologies**.
2. On the **Manage methodologies **page, click the plus sign (+).
3. In the **New methodology** pane, enter a name and description for the new methodology. Click **Confirm**.
4. **Optional:** After you have confirmed the methodology, you can promote it to the organization level by selecting the methodology in the grid, and then selecting **Promote**.



Note: You must be an admin in your organization to promote a methodology to the organization level.

Change or update a methodology

There are two ways to make changes to a methodology. You can append an existing methodology or you can make changes to a methodology in the scope of a project. From the LCS project dashboard, select the methodology that you want to update, and then select **Edit methodology** or **Append methodology**.



If you select to edit the methodology, you can make the following changes:

- Add a new phase.
- Add a new task.
- Edit a phase or task. (Some phases and tasks can be edited, but others are enforced by Microsoft and are therefore locked and can't be edited.)
- Copy a phase or task.
- Reorder phases and tasks.
- Delete a phase or task.

Business process modeler (BPM) in Lifecycle Services (LCS)

2/18/2021 • 2 minutes to read • Edit Online

Business process modeler (BPM) in Microsoft Dynamics Lifecycle Services (LCS) is a tool that you can use to create, view, and modify repeatable implementations that are based on business process libraries. BPM helps you align your business processes with industry-standard processes that are described by the American Productivity & Quality Center (APQC). You can perform fit-gap analysis between your business requirements and the default processes in Finance and Operations apps. Additionally, you can add new business processes that aren't already defined.

BPM is compatible with the following products:

- **Microsoft Word** – You can generate documentation for business processes.
- **Microsoft Visio** – You can export business process maps to Visio files.

> **NOTE**
>
> The information in this topic is specific only to Finance and Operations apps. For information about Business process modeler and Microsoft Dynamics AX 2012, see Business process modeler (BPM) in Lifecycle Services (LCS).

Prerequisites

To effectively use BPM, you must have Microsoft Office 2010 and a Microsoft Azure DevOps project.

Getting started

Follow these steps to access BPM.

1. Go to LCS.
2. Sign in, open a project, and then select the **Business process modeler** from the drop-down menu. The **Business process libraries** page has three sections:
   - **Project libraries** – This section contains business processes that a user has created or added.
   - **Corporate libraries** – This section contains custom business processes that someone in your organization has published.
   - **Global libraries** – This section contains cross-industry standard business processes, typically published by Microsoft.
3. To copy a standard business process library from the **Global libraries** section to the **Project libraries** section, select the upper-right corner of the tile in the **Global libraries** section, and then select **Copy**.
4. After the business process library has been added to the **Project libraries** section, select the tile to view the business process library.

Additional resources

- Create, edit, and browse Business process modeler (BPM) libraries
- Synchronize BPM libraries with Azure DevOps
- Complete tasks in Business process modeler (BPM)

- Work with activity diagrams in Business process modeler libraries
- Business process libraries in Business process modeler (BPM)
- Flowcharts in Business process modeler (BPM)

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Business process libraries in Business process modeler (BPM)

2/18/2021 • 2 minutes to read • Edit Online

This article explains how to view a standard business process library in Business process modeler, how to copy and modify a business process library, and how to export information about the business process library to Microsoft Word.

This topic explains how to view a standard business process library, how to copy a business process library, modify it, and how to export information for the business process library to Microsoft Word.

View and copy a standard business process library

You can select a standard business process library on the Business process library page. To open this page, sign in to Lifecycle Services, open a project, and then click the Business process modeler tile.

> **IMPORTANT**
> The business process libraries that are available depend on the industry that was selected when the Lifecycle Services project was created.

To select a standard business process library to start with, follow these steps:

1. Sign in to Lifecycle Services, open a project, and then click the **Business process modeler** tile.
2. In the **Global libraries** or **Corporate libraries** section, right-click a library.
3. On the app bar, click **Copy**. The library is added to the **My libraries** section.
4. In the **My libraries** section, click the library to display the business process library.

Search for a process within a library

You can search for a relevant business process within a business process library.

1. Sign in to Lifecycle Services, open a project, and then click the **Business process modeler** tile.
2. Open a business process library.
3. In the search field, enter a search term or phrase, or a $ followed by the AOT name of an object. For example, $LedgerJournalTransDaily.

Modify a business process library

You can modify a business process library if it is associated with a project as described in the previous procedure. To modify a business process library, follow these steps:

1. Sign in to Lifecycle Services, open a project, and then click the **Business process modeler** tile.
2. Open a business process library.
3. Make changes to the business process library.
   - To change an existing library node, right-click the node to display the app bar, and then click **Edit**. Make changes, and then click **Save**.
   - To add a library node, drag a node from the **Activities** list to the library. Right-click the new node and then click **Edit** to change the name and other information for the node. Make changes, and then click

Save.

- To delete a library node, right-click the node, and then click **Delete**.

Export a business process library to Word

You can export information about a business process library, and all the flowcharts that are associated with it, to a Word document. To export a business process library to Word, follow these steps:

1. Sign in to Lifecycle Services, open a project, and then click the **Business process modeler** tile.
2. Open a business process library.
3. In the **Core Business Processes** list, right-click a top-level node in the library.
4. On the app bar, click **Doc**, and save the document.

Create, edit, and browse Business process modeler (BPM) libraries

2/18/2021 • 4 minutes to read • Edit Online

This topic provides information about how to create, edit, and browse Business process modeler (BPM) libraries. It's important to note You can browse a BPM library that is a global library or a corporate library. However, before you can edit and work with a BPM library, it must be part of your project in Microsoft Dynamics Lifecycle Services (LCS). Libraries that are distributed by Microsoft appear under **Global libraries**, whereas libraries that are published by your organization appear under **Corporate libraries**.

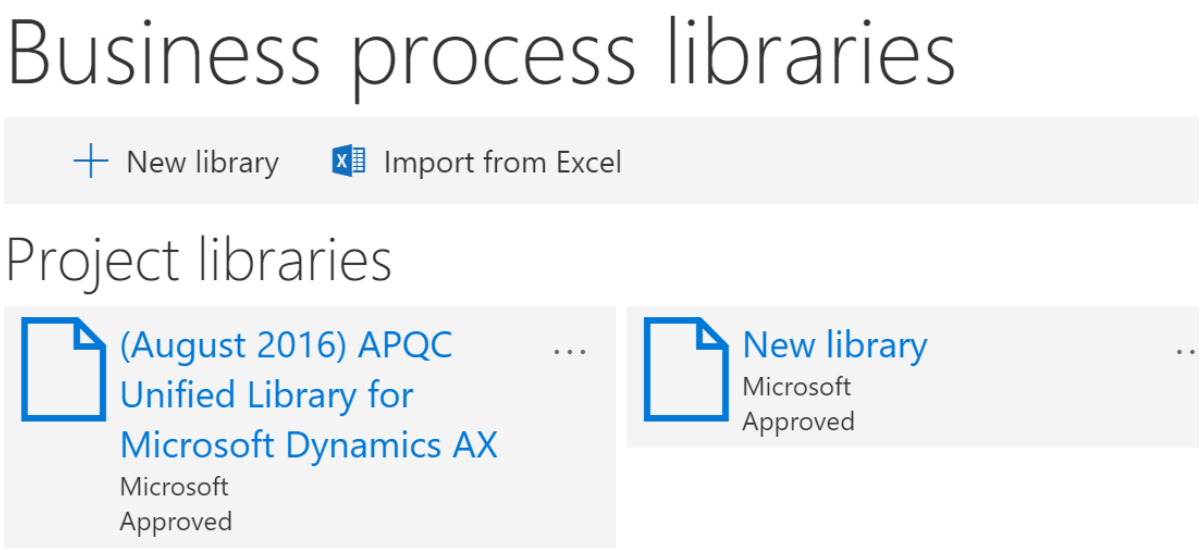
> **NOTE**
>
> BPM localization is not supported. If you edit in the new BPM client in any language other than EN-US, your changes will only display when you view the BPM in the language in which the changes were made. To view any changes made in EN-US, you must synchronize with Visual Studio Team Server before the changes will display.

Create a BPM library

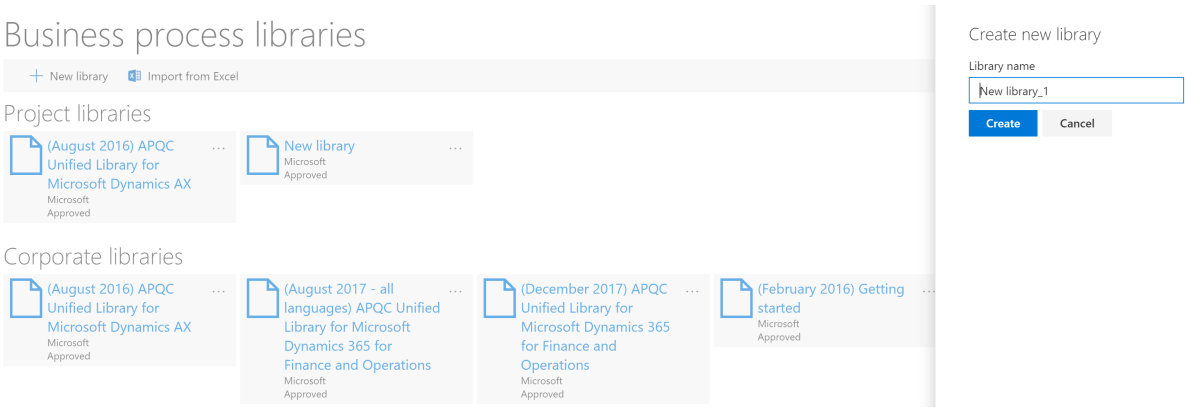
There are several ways to author a BPM library. You can do so from scratch either building directly in the client or by importing an Excel template. Additionally, you can copy an existing library. This section walks through each of these methods.

Use the BPM client

1. On the **Business process libraries** page, select **New library**.

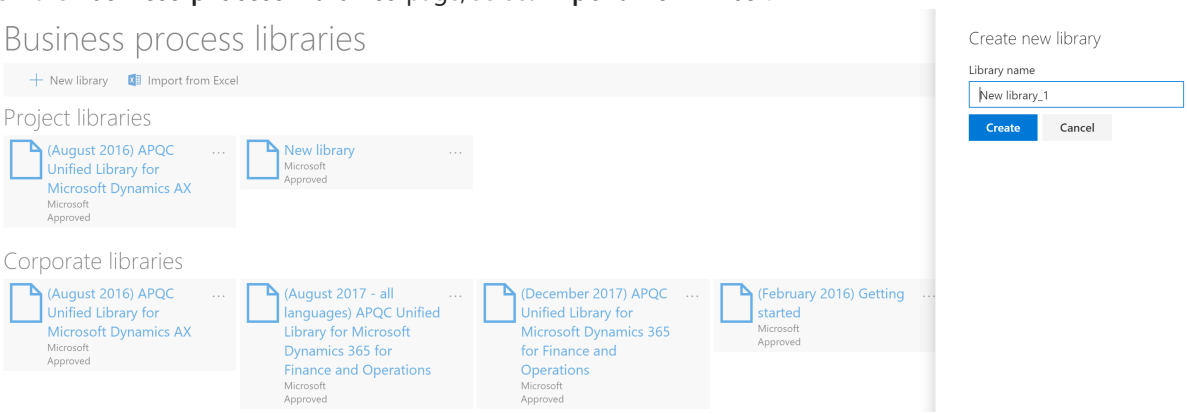


2. Enter a name for the new library, and then select **Create**.

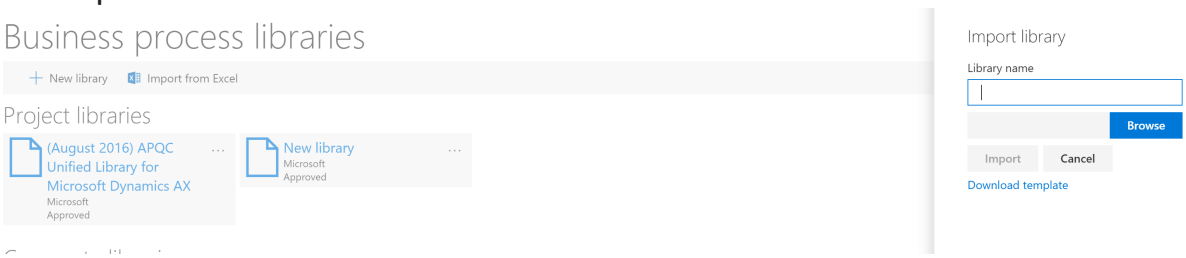


Use Excel Import

1. On the **Business process libraries** page, select **Import from Excel**.

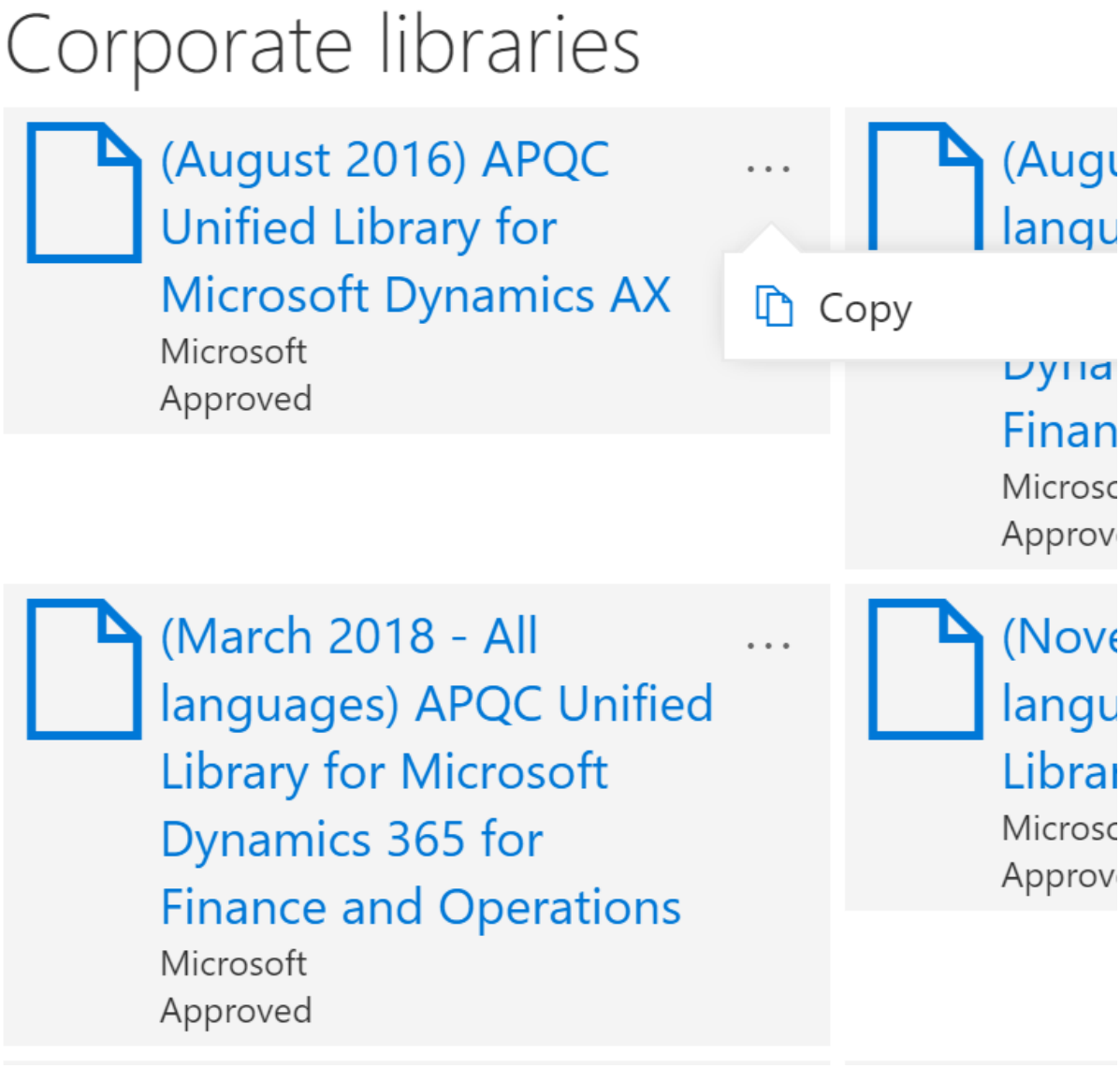


2. Select **Download template** from the pane. Once downloaded, open the file.
3. The template has several columns, most importantly **Id** and **Parent Id**. Associate each line with a new Id number, if you'd like to make a line a child item, add the Id of the line you'd like it to fall under in the parent Id column. the
4. Once complete, save the template and return to BPM.
5. Using the import pane, select **Browse** to upload the updated template, enter a name for the new library, and select **Import**.

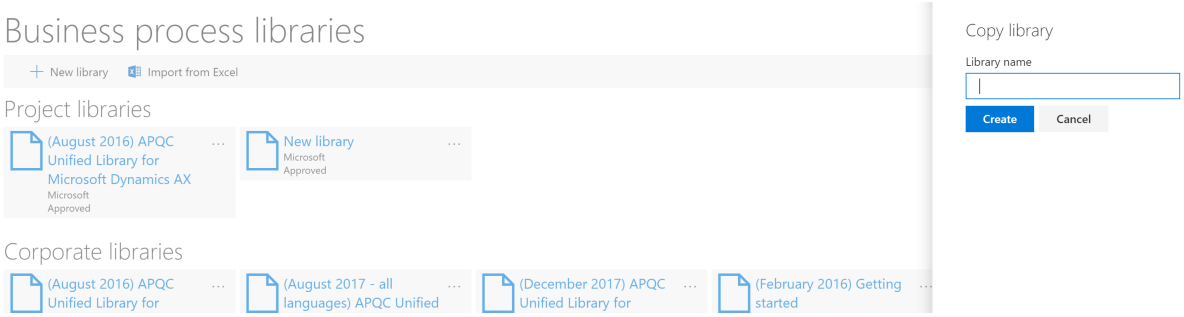


Copy a library

1. Open the **Business process libraries** page.
2. On the tile for the library that you want to copy, select the ellipsis button (...), and then select **Copy**.

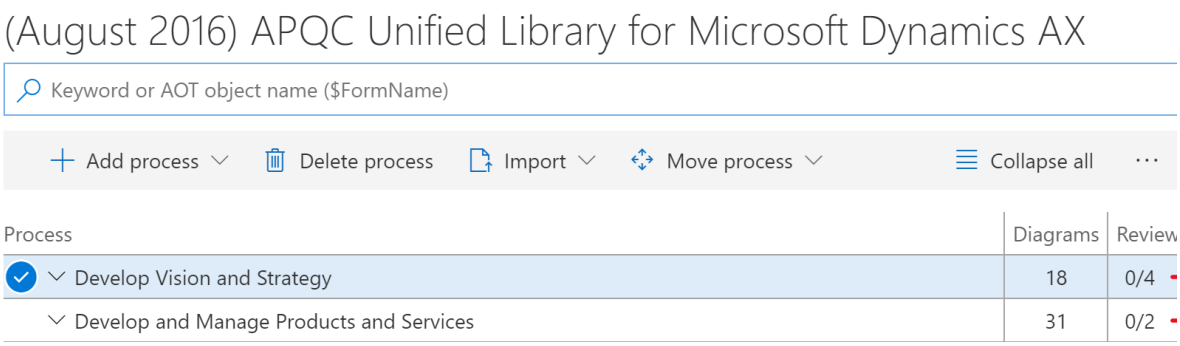


3. Enter a name for the library, and then click **Create**.

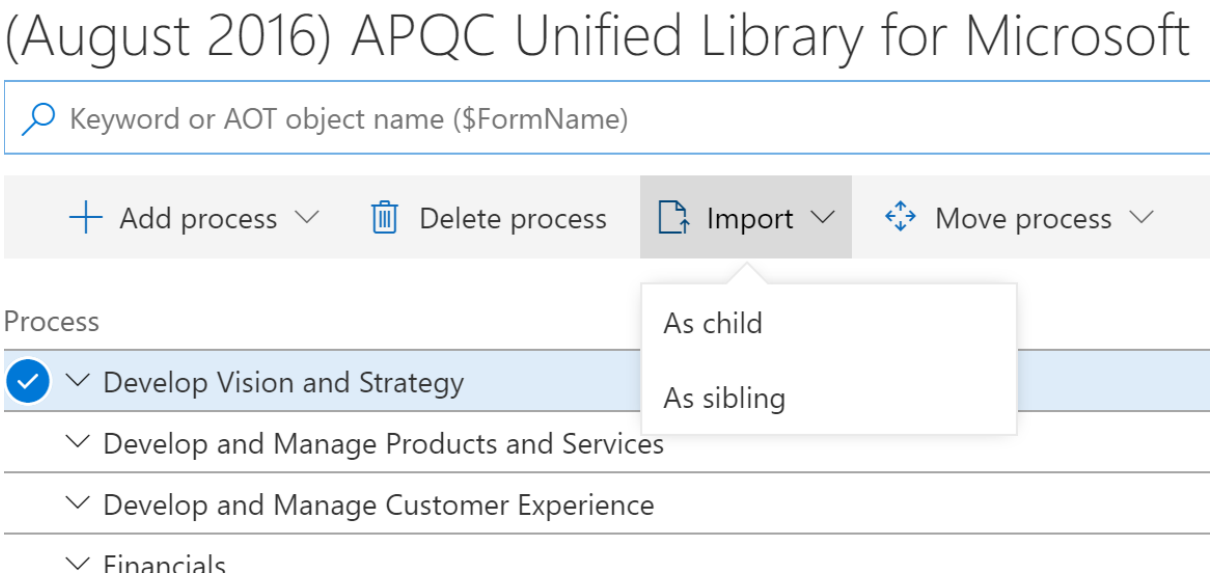


Import a sections of another library

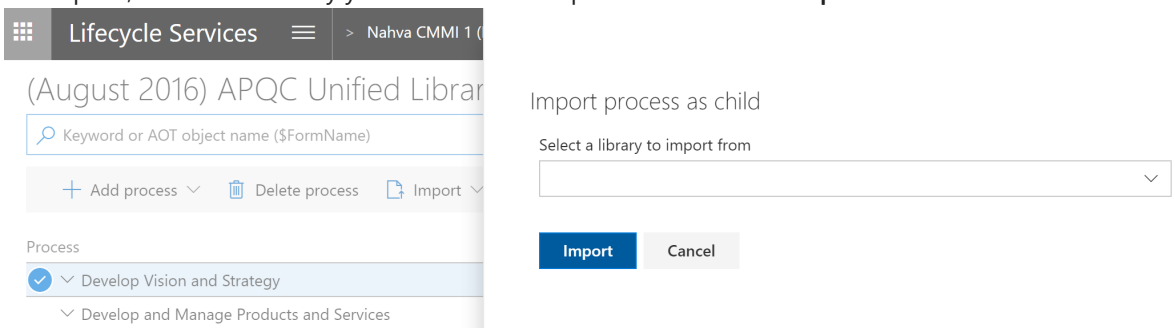
1. Openn the **Business process libraries** page, and then open the library you want to edit.
2. Navigate to the line you would like to import to and select **Import**.



3. Select **As child** or **As sibling**.

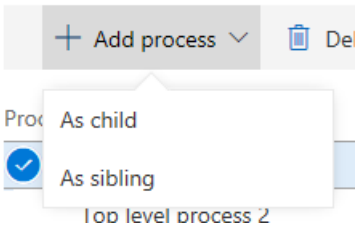


4. In the pane, choose the library you would like to import from and click **Import**.



Add a new process

1. In the BPM library, select an existing process.
2. Select **Add process**. You can select to add the process as a child or a sibling of the selected process node. In this way, you can create a semantic hierarchy of business processes.



Edit the properties of a process

1. In the BPM library, select the process node to edit.
2. In the right pane, on the **Overview** tab, click **Edit mode**.

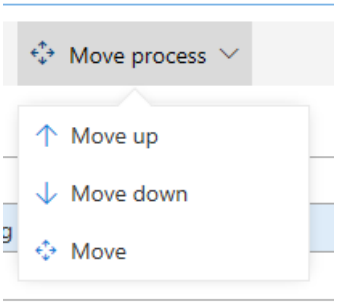
3. Enter a name and description for the process node.
4. Select the industries and the countries or regions that the process applies to. You can also add keywords and links. Keywords let you define categories, work streams, or other metadata. Links (URLs) let you reference external sites or documentation.

5. When you've finished editing the properties, click **Save**.

Move a process

You can move a process node or assign it to another parent node in the BPM hierarchy.

1. Select the process node to move, and then click **Move process**. You can select to move the process up or down, or you can select **Move** to see more options.



2. If you selected **Move**, you can browse the hierarchy, select a node to move the process to, and then select **Move as child** or **Move as sibling**. To cancel the move operation, click **Cancel**.

Delete a process

To delete a business process, select the process to delete, and then select **Delete**.

Copy a global or corporate library to your project

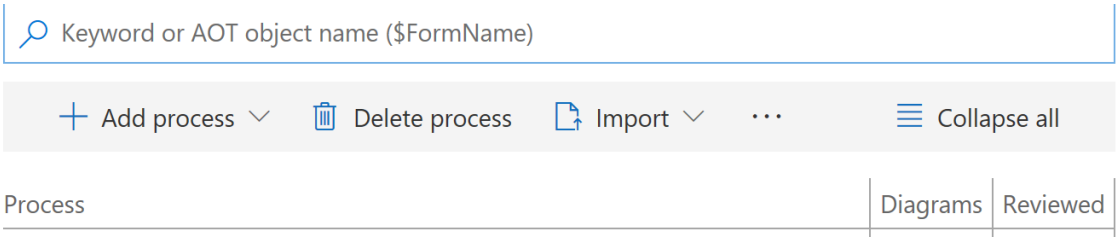
You can browse a BPM library that is a global library or a corporate library. However, before you can edit and work with a BPM library, it must be part of your project in Microsoft Dynamics Lifecycle Services (LCS). Libraries that are distributed by Microsoft appear under **Global libraries**, whereas libraries that are published by your organization appear under **Corporate libraries**.

Browse a BPM library

1. On the **Business process libraries** page, double-click the tile for the library that you want to browse.
2. In the BPM library, select a process to view its substeps.

Process	Diagrams	Reviewed
Develop vision and strategy	22	0/3
Develop and Manage Products and Services	37	0/2
Manage product and service portfolio	28	0/7
Evaluate performance of existing products/services against market opportunities		0/1
Define product/service development requirements		0/2
Perform discovery research		0/3
Confirm alignment of product/service concepts with business strategy		0/4
Manage product and service life cycle		0/3
Manage product and service master data	11	0/18
Manage product configuration models	17	0/2
Develop products and services	9	0/5

3. Use the buttons on the toolbar to add, delete, or import processes as a child or a sibling. You can also select **Collapse all** to view only parent processes.

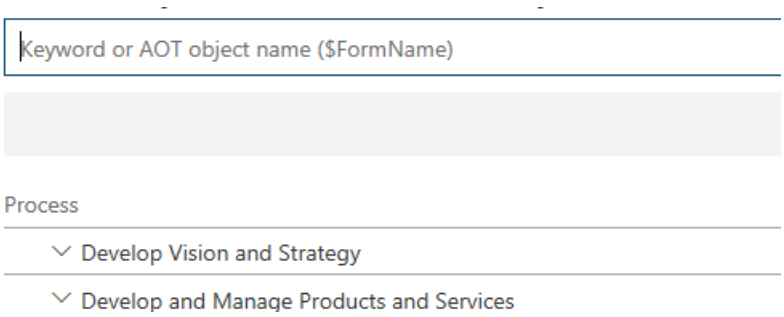


Search a BPM library

You can search for words or phrases in your BPM library. The search functionality searches the names and descriptions of business processes.

- To search for a *word*, enter the search word in the search box, and then press Enter.
- To search for a *phrase*, put double quotation marks around the search phrase.

  For example, enter **technology** (word) or **"information technology"** (phrase) in the search box.
- You can also search for Application Object Tree (AOT) elements that are part of the task recordings that are in your library. Typically, these AOT elements are the names of pages or menu items. When you search for an AOT element, prefix it with a dollar sign ($). For example, enter **$CustTable** in the search box.



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

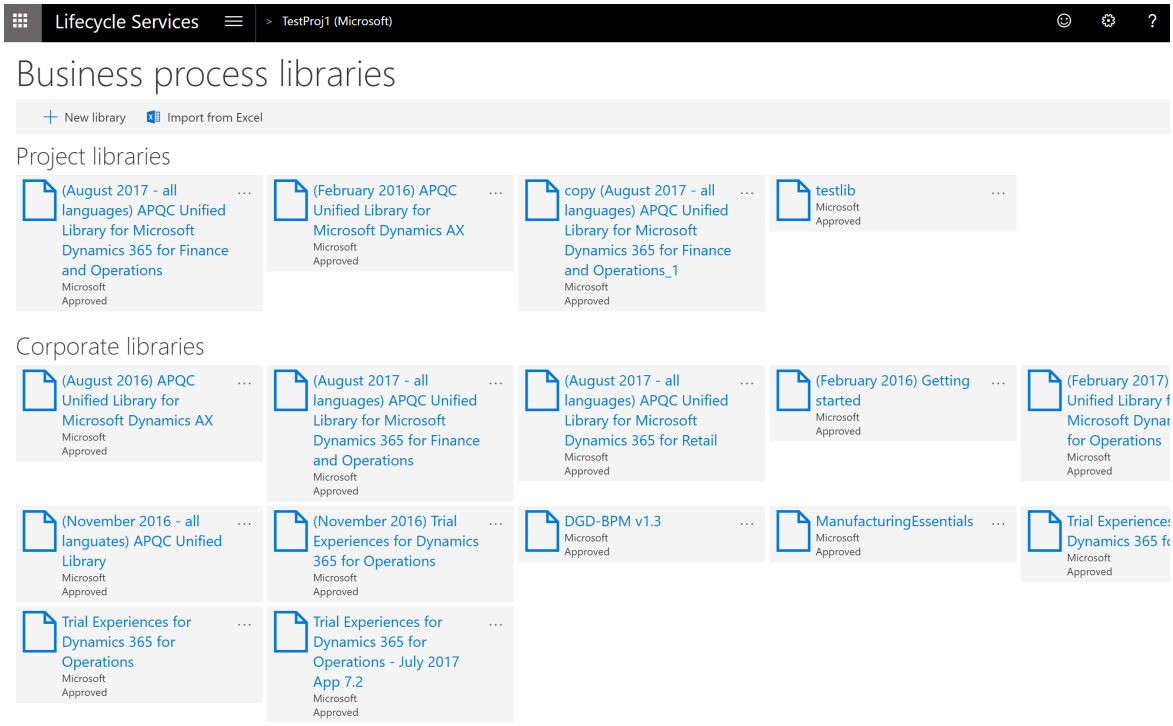
The survey will take about seven minutes. No personal data is collected (privacy statement).

Complete tasks in Business process modeler (BPM)

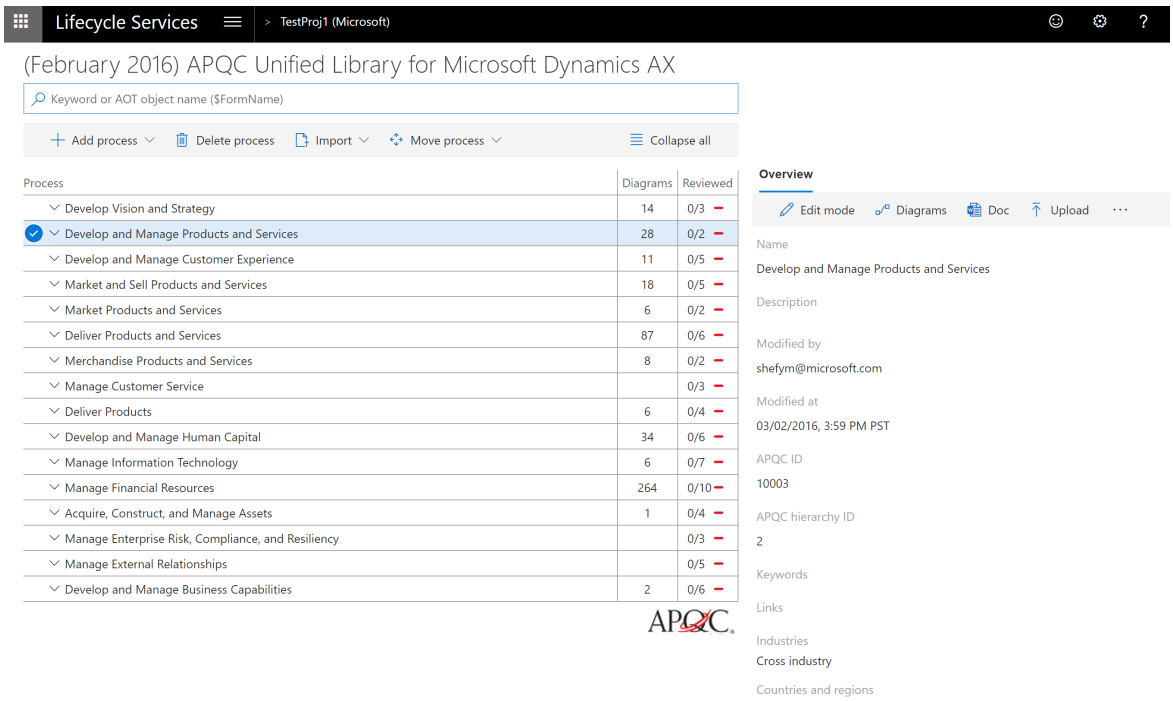
2/18/2021 • 2 minutes to read • Edit Online

Upload a task recording

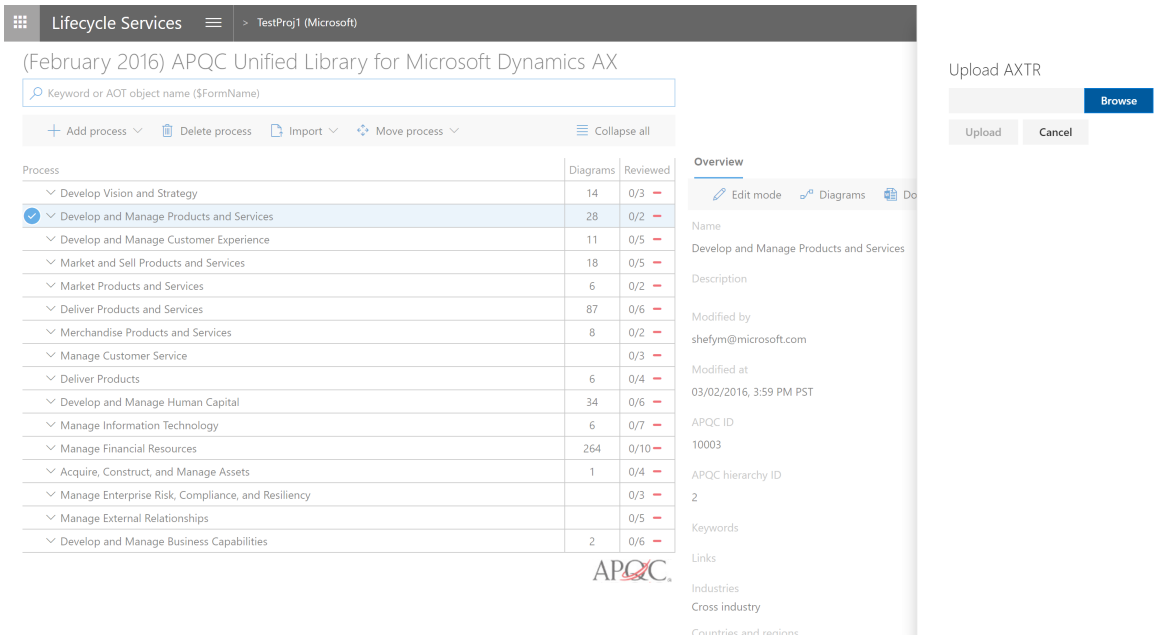
1. In Microsoft Dynamics Lifecycle Services (LCS), in your project, on the **Business process libraries** page, select the library to upload the task recording to.



2. Select the process to upload the task recording to.



3. On the **Overview** pane, select **Upload**. Select **Browse** to find and select the file to upload, and then select **Upload**.



Download a task recording

You can download a task recording (AXTR file) that has been uploaded to a BPM process.

1. In your LCS project, on the **Business process libraries** page, select the library to download the task recording.
2. Select a process that has task recording uploaded.
3. On the **Overview** pane, select **Download** to save the task recording (AXTR).

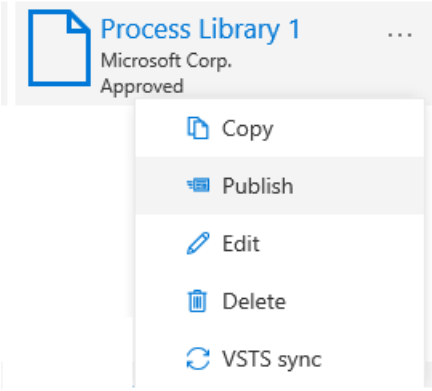
Export a methodology to Word

1. In your LCS project, on the **Business process libraries** page, select the library to export.
2. Select the process to export, and then, in the right pane, select **Doc** to begin the download.

> **NOTE**
>
> The methodology will begin from the process step that you selected.

Publish a BPM library

- In your LCS project, on the **Business process libraries** page, on the tile for the library that you want to copy, select the ellipsis button (...), and then select **Publish**.



Distribute a BPM library

When you distribute a BPM library, the library will be available to all users who are a part of your organization. In other words, it will be available to all users who sign in to LCS by using your organization's domain (for example, all users who have an @contoso.com account).

1. Ask the customer to invite you to their project.
2. Sign in to the customer's LCS project by using your organization's account.
3. On the **Business process libraries** page, copy the library from the **Corporate libraries** pane to the customer's project.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Work with activity diagrams in Business process modeler libraries

2/18/2021 • 2 minutes to read • Edit Online

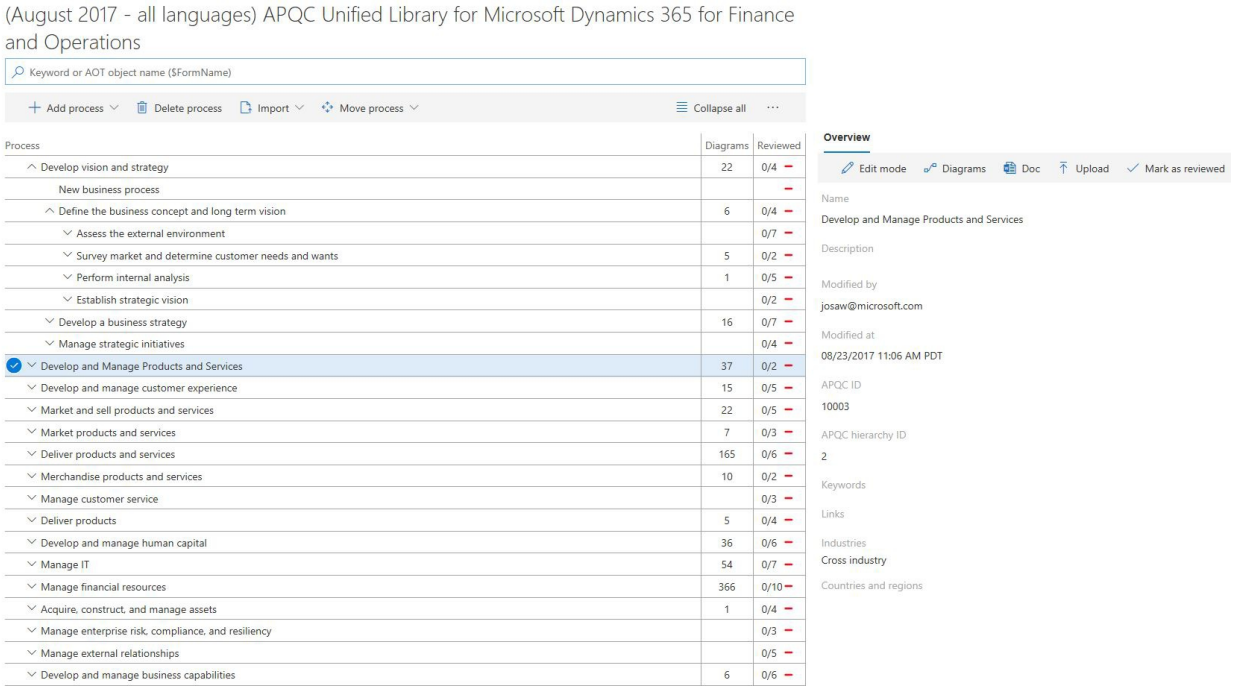
You can associate an activity diagram with a business process. Activity diagrams are used to describe how a business process or task is completed in a proposed software solution.

There are two types of activity diagrams:

- **Task recordings** – Business processes that are associated with task recordings for Finance and Operations, include activity diagrams and process steps that are automatically generated.
- **Microsoft Visio** – You can associate a business process with a Visio diagram by manually uploading a Visio file.

Browse activity diagrams

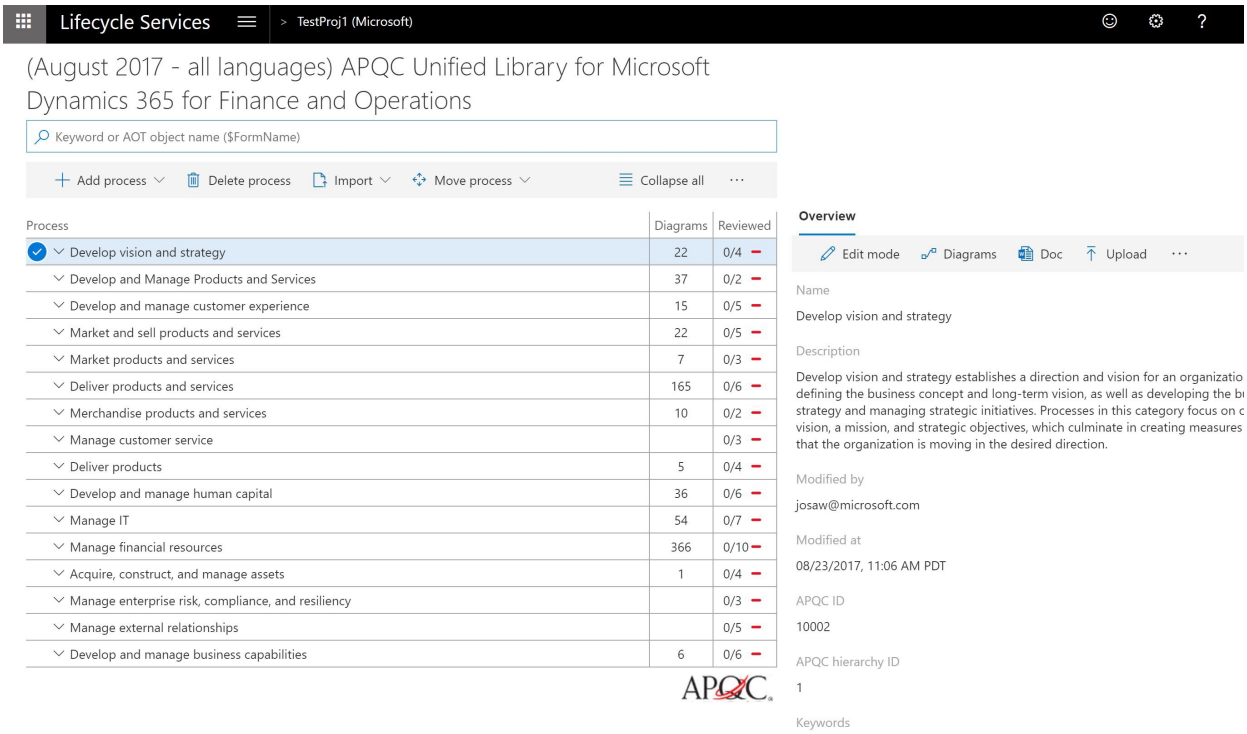
The **Diagrams** column in your BPM library indicates whether a particular business process is associated with an activity diagram. The number in the column indicates the number of child processes that include diagrams. The symbol next to the number indicates whether the current node or process is associated with a diagram. These indicators don't apply to Visio diagrams.



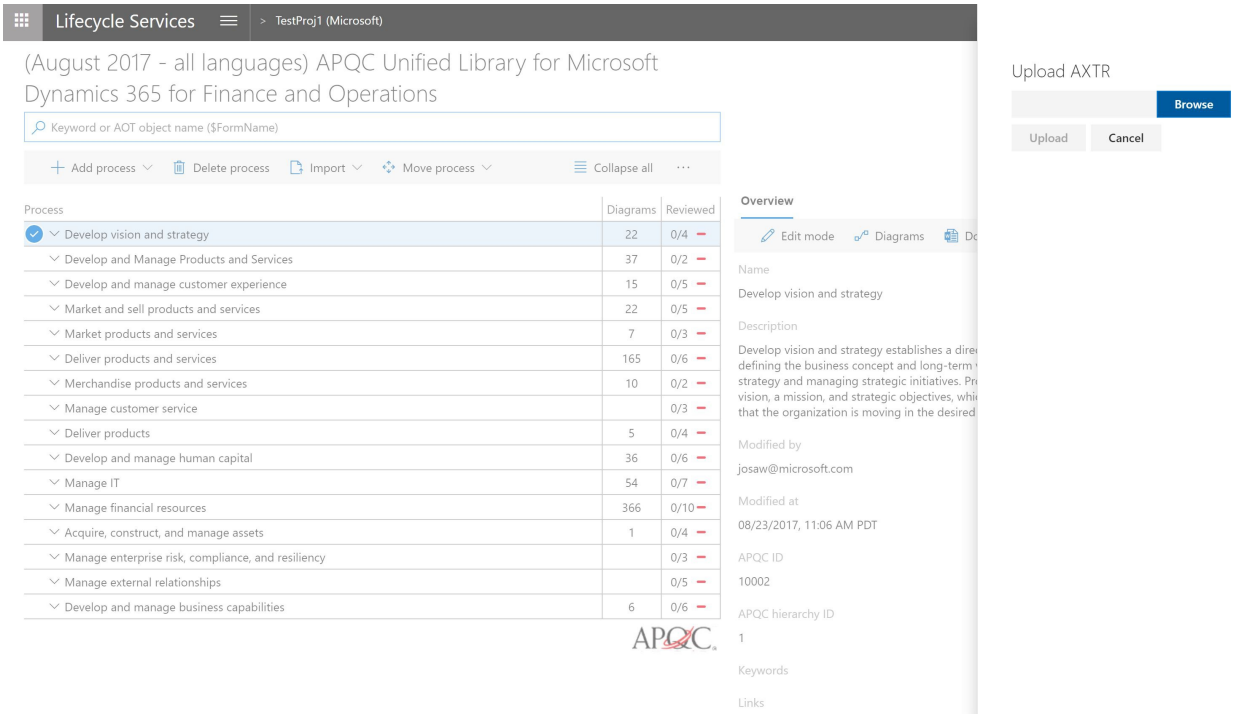
To view an activity diagram, select the business process, and then, in the right pane, on the **Overview** tab, select **Diagrams**. The **Flowchart** page appears.

Upload Task Recording

To upload a task recording, open the business process library that you want to upload to. Select the process step that you want to upload the task recording to, and then click **Upload**.



In the right pane, click **Browse** to choose a file, and then click **Upload**.



Activity diagrams that are created from task recordings

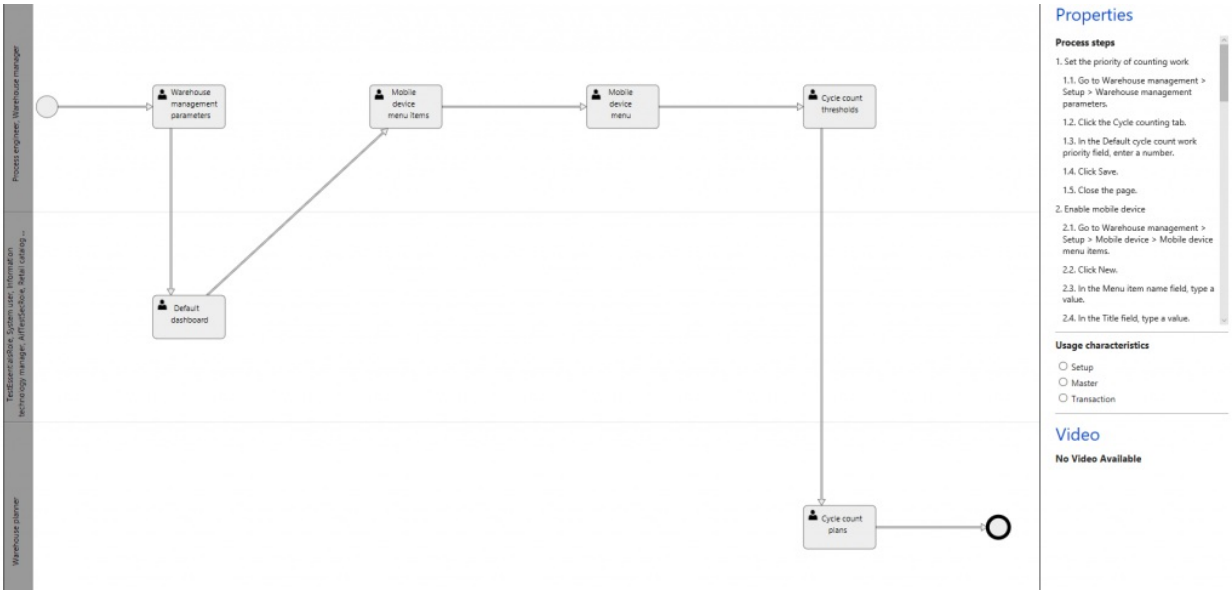
> [!IMPORTAMT] Flowchart diagrams in Business process modeler have been deprecated. To learn more about the deprecation, see Flowchart diagrams in Business process modeler.

You can create a task recording in your environment and save it directly to Microsoft Dynamics Lifecycle Services (LCS). In this way, you can associate the task recording with a business process in a BPM library. For more information, see Connecting the help system and Create documentation or training with Task Recorder.

The Task recorder tool lets you create a distributable recording file. Recording files have the .axtr file name extension. You can associate a business process in BPM with a task recording by manually uploading the recording file.

To upload a recording file, select the business process, and then, in the right pane, on the **Overview** tab, select **Upload**.

BPM automatically generates an activity diagram and detailed process steps for all task recordings that are created. The following illustration shows an example.



Visio files

You can associate a business process with a Visio diagram. Typically, this functionality is used for high-level processes that can't be represented by a task recording.

> **NOTE**
>
> BPM supports .vsd and .vsdx files. However, it doesn't support .vsdm files (macro-enabled Visio drawing files). If a .vsd file contain macros, BPM disables the execution of the macros.

To view or upload a Visio file, follow these steps.

1. Select the business process, and then, in the right pane, on the **Overview** tab, select **Diagrams**.
2. On the **Flowchart** page, select the **Visio** tab. For more information, see the "Unconnected flowcharts" section in Flowcharts in Business process modeler (BPM).

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Synchronize BPM libraries with Azure DevOps

2/18/2021 • 6 minutes to read • Edit Online

You start the implementation stage of a project by synchronizing a Business process modeler (BPM) library with your project in Microsoft Azure DevOps. In this way, you can review processes and associate requirements with business processes. By synchronizing a BPM library with a Azure DevOps project, you can also track the progress of your implementation project in Azure DevOps, and can associate various work items with requirements and business processes. These work items include bugs, tasks, backlog items, tests, and documents.

Currently, BPM-Azure DevOps synchronization doesn't support custom work item types or synchronizing business processes with custom work item types. If you try either of these, you will receive a warning. If you choose to ignore the warning and attempt a Azure DevOps sync with a custom template, you can avoid synchronization issues by verifying the following for the template:

- Does not delete any work item type
- Does not delete any state of a work item type
- Does not add any required fields to a work item type

To learn more about Azure DevOps, go to www.visualstudio.com/team-services.

LCS project settings: Set up Azure DevOps

If you've already set up Azure DevOps from Microsoft Dynamics Lifecycle Services (LCS), you can skip the procedures in this section.

Create a personal access token

To connect to a Azure DevOps project, LCS is authenticated by using a personal access token. Follow these steps to create a personal access token in Azure DevOps.

1. Go to https://www.visualstudio.com, sign in, and find your Azure DevOps project.
2. In the upper-right corner, hold the pointer over your name, and then, on the menu that appears, select **Security**.
3. Select **Add** to create a new personal access token.
4. Enter a name for the token, and then specify how long the token should last.
5. Select **Create Token**.
6. Copy the token to your clipboard.

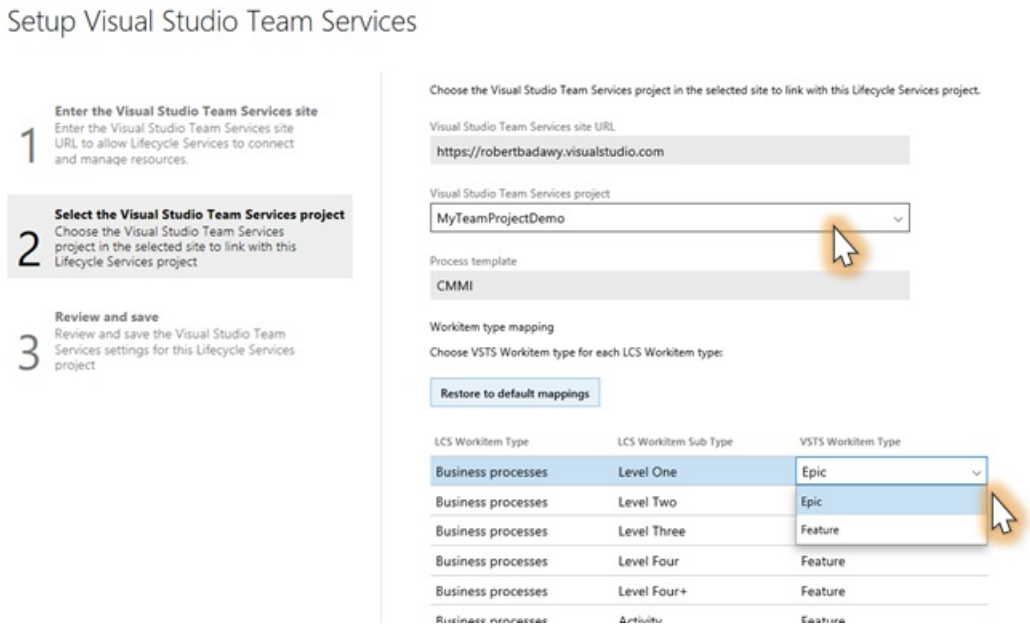
> **NOTE**
> You won't be able to find the token details again after you complete this step and move away from the page. Therefore, make sure that you've copied the token before you move away from the page.

Configure your LCS project to connect to Azure DevOps

1. In your LCS project, select the **Project settings** tile.
2. Select **Azure DevOps**, and then select **Setup Azure DevOps**. This configuration is required by many LCS tools. If you've already configured LCS to connect to your Azure DevOps project, you can either skip

this procedure or select **Change** to change the existing configuration.

3. Enter the root URL for your Azure DevOps account, and the personal access token that you created earlier, and then select **Continue**.
4. Select your Azure DevOps project.
5. Specify the mapping between LCS/BPM items and the associated Azure DevOps work item types.



6. Select **Continue**, review your changes, and then select **Save**.

Synchronize a BPM library with a Azure DevOps project

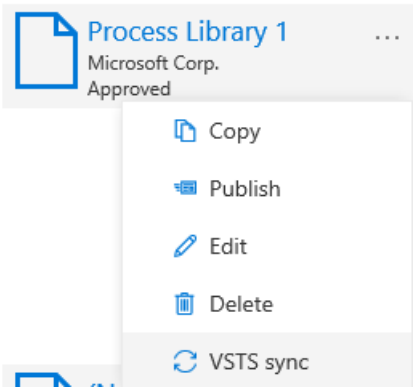
After you've set up the connection between the LCS project and a Azure DevOps project, you can synchronize a BPM library with the Azure DevOps project. When you synchronize a BPM library with a Azure DevOps project, a Azure DevOps work item is created for each business process line in the BPM library. In addition, the hierarchy of business processes in BPM is reflected in the hierarchy of work items in Azure DevOps. The type of work items that are created in Azure DevOps depends on the settings of your LCS project.

This synchronization is a one-way synchronization. Changes in LCS are reflected in Azure DevOps, but changes in Azure DevOps aren't reflected in LCS.

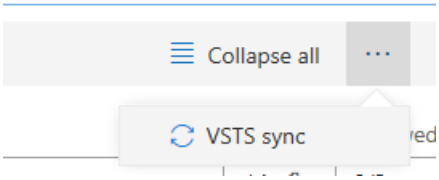
The following information is synchronized:

- Business process names
- Business process descriptions
- Keywords (as tags)
- Countries or regions (as tags)
- Industries (as tags)

To synchronize a BPM library with a Azure DevOps project, on the **Business process libraries** page, on the tile for the library that you want to synchronize, select the ellipsis button (...), and then select **Azure DevOps sync**.



You can also start Azure DevOps synchronization from the toolbar in a BPM library. Select the ellipsis button (...), and then select **Azure DevOps sync**.



> **NOTE**
> BPM localization is not supported. If you edit in the new BPM client in any language other than EN-US, your changes will only display when you view the BPM in the language in which the changes were made. To view any changes made in EN-US, you must synchronize with Visual Studio Team Server before the changes will display.

Turn off synchronization of BPM with Azure DevOps

To turn off synchronization, on the **Business process libraries** page, select the library that you want to stop synchronizing, select the ellipsis button (...), and then unselect **Azure DevOps sync**.

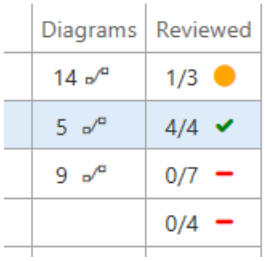
Review processes and add requirements

During the project phase where you're gathering requirements, you can use the BPM library to review business processes and tasks, and to identify requirements. In BPM, you can mark business processes as reviewed to track the review process.

To mark a process or one of its child processes as reviewed, select the process in BPM, and then, in the right pane, on the **Overview** tab, select **Mark as reviewed**.

When a business process is marked as reviewed, the **Reviewed** column is updated. This column shows the following information:

- A fraction indicates how many direct child processes have been reviewed.
- A symbol indicates how completely the process and its child processes have been reviewed:
  - **Green check mark** – The process and all its child processes have been fully reviewed.
  - **Yellow circle** – The process and its child processes have been partially reviewed.
  - **Red dash** – The process and its child processes haven't been reviewed.



While you're reviewing a business process that is connected to Azure DevOps, you can add a requirement directly to your Azure DevOps project.

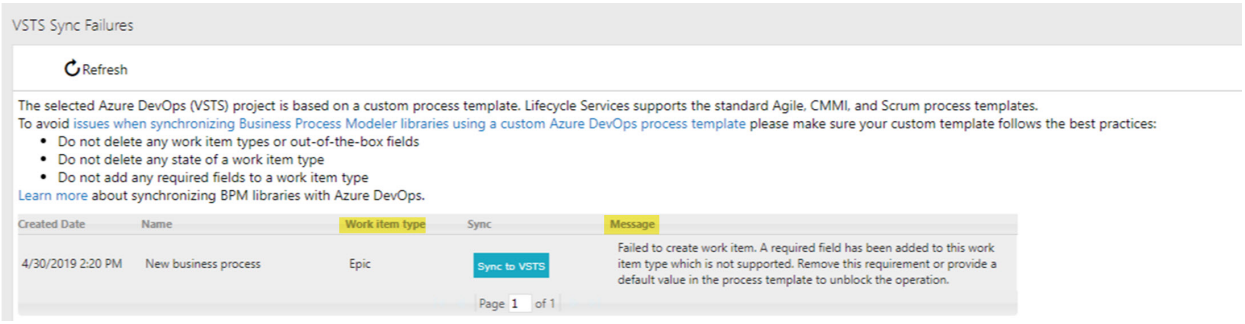
1. Select a business process.
2. In the right pane, on the **Requirements** tab, select **Add requirement**.
3. Enter a name, description, and type, and then select **Create**.

   In Azure DevOps, a requirement work item is created that is associated with the current business process.

To go to the Azure DevOps work items that are associated with the current business process, on the **Requirements** tab, select the appropriate links.

Common syncing errors

If the BPM to Azure DevOps synchronization fails, you will see the failed process name, work item type, and an error message.



Here are some common causes and suggested actions to resolve the error.

POSSIBLE CAUSE	ERROR MESSAGE	SUGGESTED SOLUTION
Required field added	Failed to create work item. A required field has been added to this work item type, which is not supported. Remove this requirement or provide a default value in the process template to unblock the operation.	Remove the required field or provide a default value.
Work item type disabled	Failed to create work item. The work item type has been disabled in the process template. Enable the work item type to unblock the operation.	Enable the work item type in the process template

POSSIBLE CAUSE	ERROR MESSAGE	SUGGESTED SOLUTION
Couldn't find work item to update	Failed to update work item. The work item does not exist, or you do not have permissions to read it. Check the PAT configuration in the project settings or restore the work item if it has been deleted directly from the DevOps project.	Restore the work item from the recycle bin if it was deleted, or create a new Personal Access Token (PAT) and make sure that it has full permissions.
Personal Access Token is expired	Failed to sync with Visual Studio Team Services. The request response is: Unauthorized. Please check that the PAT is setup correctly and still valid, try again and contact support if the error persists.	Create a new Personal Access Token (PAT) from Azure DevOps and update the PAT value in your LCS Project settings.
Generic error	Failed to sync with Visual Studio Team Services. The request response is: {0}. Please check that the PAT is setup correctly and still valid, try again and contact support if the error persists.	Contact customer support with the request response that caused the syncing error.

Create and automate user acceptance tests

2/18/2021 • 8 minutes to read • Edit Online

You can use Task recorder and Business process modeler (BPM) to create user acceptance test libraries. Task recorder is a powerful tool to record test cases and organize them by business process using BPM. As a Microsoft partner you can use BPM to distribute test libraries to your customers via LCS and LCS solutions. If you are a customer, use BPM to author and distribute test libraries across different projects and team.

Because BPM can be synchronized with Azure DevOps (formerly known as Visual Studio Team Services), you can automatically create test cases (including test steps) in your Azure DevOps project. Azure DevOps can then serve as your test configuration and test management tool where you can create targeted test plans and test suites, manage the execution of tests and investigate results. For more information about testing with Azure DevOps, see What are test plans, test suites, and test cases?

This topic walks through the process of creating and executing acceptance test suites to be used for manual or automated testing.

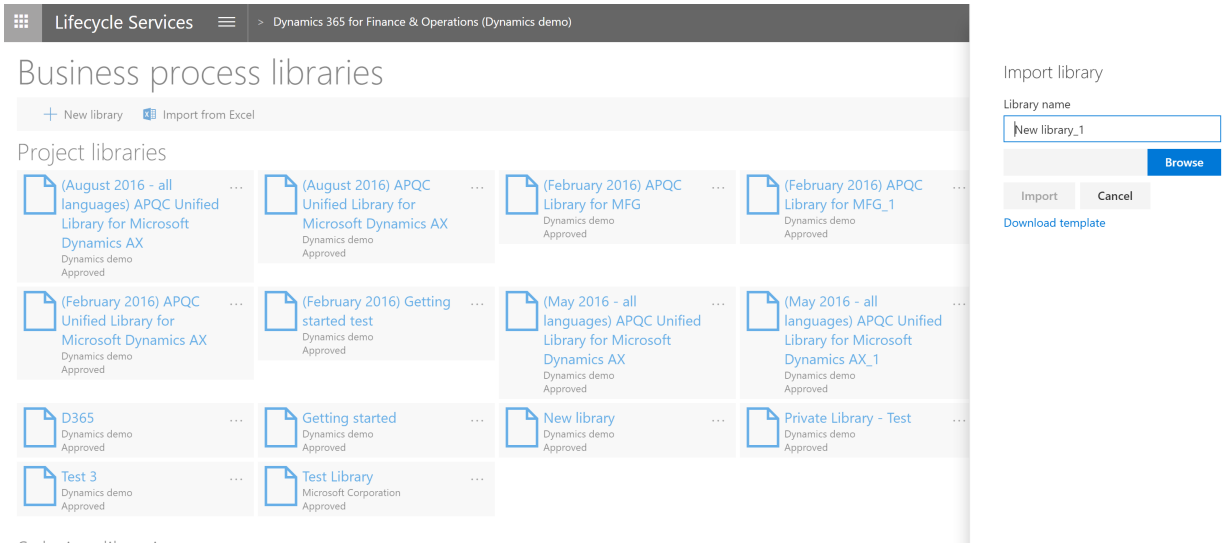
Create a Scenario Acceptance Testing BPM library

BPM is a great LCS tool to describe a hierarchy of business processes and user tasks. LCS also allows Microsoft partners and customers to author and distribute BPM libraries across LCS projects via the Asset library. This section describes how to take advantage of BPM to define your acceptance test library.

Create a BPM library

There are several ways to create a Business process modeler (BPM) library. For more information about how to create libraries in BPM, see Create, edit, and browse Business process modeler (BPM) libraries.

For illustration purposes, this topic uses a library that contains common business processes, such as create an expense report and approve order requests. The library was created by using the Excel import functionality.



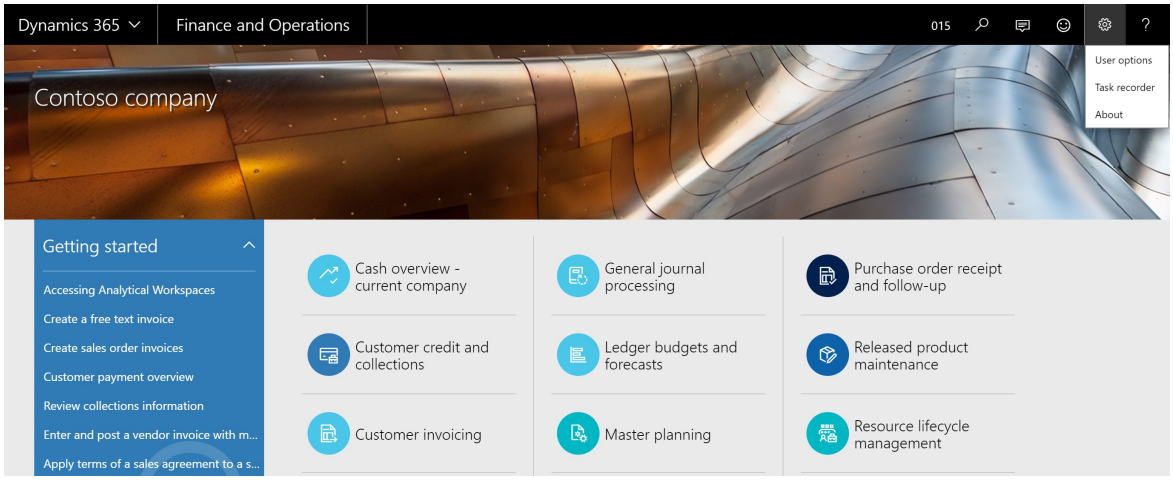
Record test cases and save to BPM

After you have created a BPM library, you'll need to use Task recorder to create your test cases and then upload the cases to BPM. There are several ways to do this.

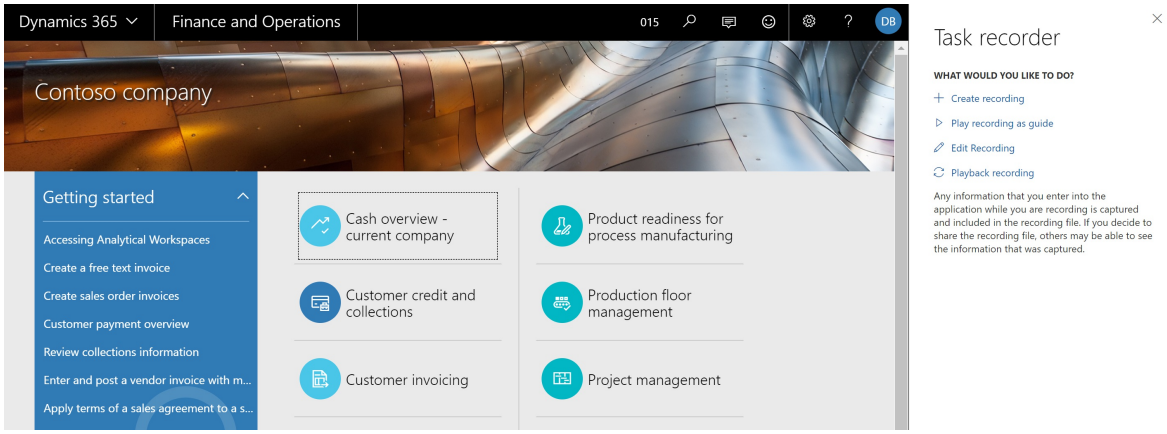
If you're using a BPM library that already has all of the necessary task recordings (test cases) attached, you can skip this step. Otherwise, follow the instructions below to create new task recordings.

Create and save a new task recording

1. Open the client and sign in.
2. Select the company that you want to use while recording.
3. Go to **Settings** > **Task recorder**.



4. Click **Create a new recording**.
5. Enter a name for the recording, and then click **Start**. Recording begins the moment that you click **Start**.
6. When the recording is complete, in the Task recorder pane, click **Stop**.
7. To save the task recording to an attached BPM, click **Save to Lifecycle Services**.



8. Select the library that you want to save the recording to, and then click **Save**. Otherwise, select **Save to Disk** and follow the steps in the next section, "Upload an AXTR file to BPM."

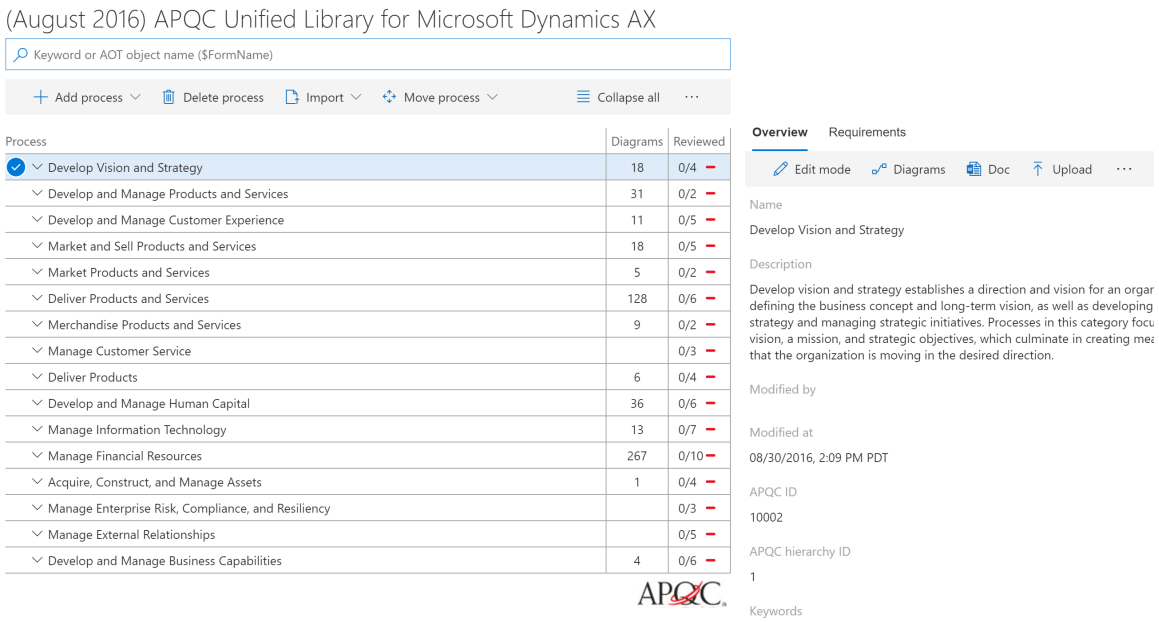
> **NOTE**
>
> To enable the effective execution of your tests using automation tools, make sure all of your task recordings start on the main dashboard of your application. For end-to-end processes that are performed by more than one user, we recommend that you divide your task recordings into user-specific tasks. This simplifies the maintenance of test cases and allows you to execute test cases in the context of security roles, which is a best a practice.

Upload an AXTR file to BPM

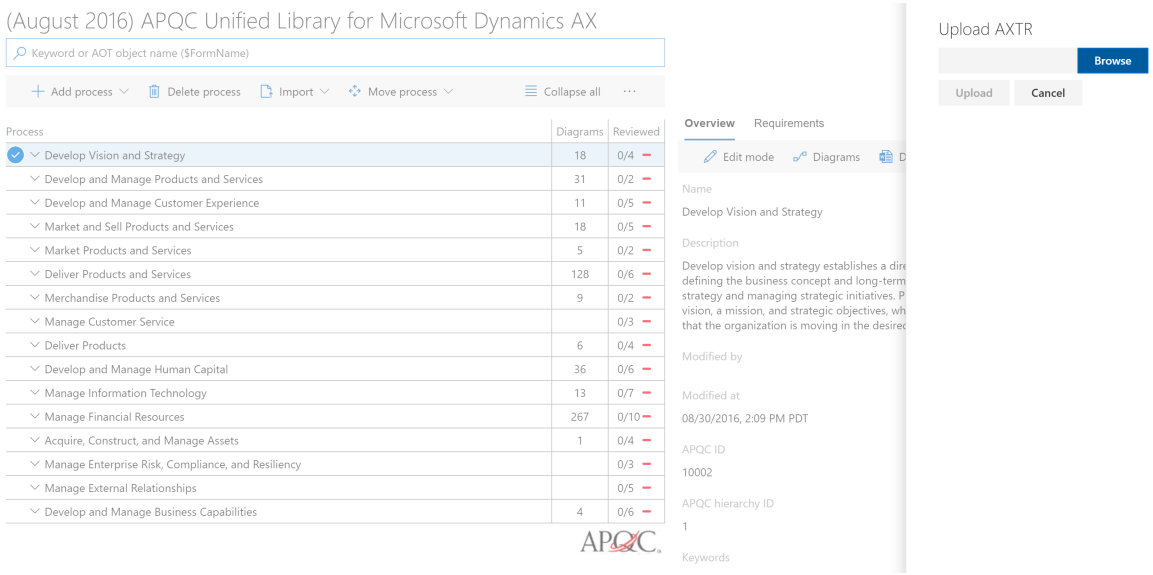
If you have saved your recordings (AXTR files) to disk, follow these steps to upload them to BPM.

1. In Lifecycle Services (LCS), in your project, on the **Business process libraries** page, select the library to upload the task recording to.
2. Click **Author and edit** and in the lines, locate and select the process to upload the task recording to.

3. In the right pane, click **Upload**.



4. Click **Browse** to find and select the file to upload, and then click **Upload**.



Save an existing task recording to BPM

1. To attach an existing task recording, sign in to the client.
2. Go to **Settings** > **Task recorder**.
3. Select **Edit Task Recording** and attach the file by either saving directly to LCS or downloading the AXTR and then uploading to BPM.

Guidelines for recording test cases

Follow these guidelines when authoring and recording your test cases, especially if you are planning to automate test execution. The process and tools described in this article apply to business process acceptance tests. They are not meant to replace component and unit testing that is typically owned by developers.

- Author a limited number of test cases that, when combined, cover complete end-to-end processes.
- Focus on business processes that have been customized.
- An individual test case (recording) should cover one or two business tasks only, typically executed by one person. This simplifies task recording maintenance. Do not combine a complete end-to-end business process such as "Procure to Pay" or "Order to Cash" into one large task recording. For example, instead of having RFQ > Purchase Order > Product Receipt > Vendor Invoice > Vendor Payment as one test case, divide the process into three or four test cases. You will have the opportunity to combine these tests into an ordered

test suite later.

- A test case should have at least one validation. Try to validate critical fields that cover the impact of other fields. For example: Validation of totals on sales or purchase orders cover the unit price/quantity/discount/tax ...etc.
- Avoid printing a report in a test case. If a test case needs to print a report, it should be selected on screen.
- 80+% of test cases should be of transactions or source documents. Master data should be limited to up to 20% of test cases only.

Synchronize and configure your test plan in Azure DevOps

An acceptance test library is your starting point. It typically contains all test cases (task recordings) of a particular application organized by business process. During a particular test pass, you usually do not need to execute all test cases. What test cases you select depends on the phase of your implementation or the nature of the update you are planning to apply to your production environment. Azure DevOps enables you to organize your test cases in test plans and test suites. A test plan contains one or more test suites (A subset of your test library); test cases can belong to more than one test suite.

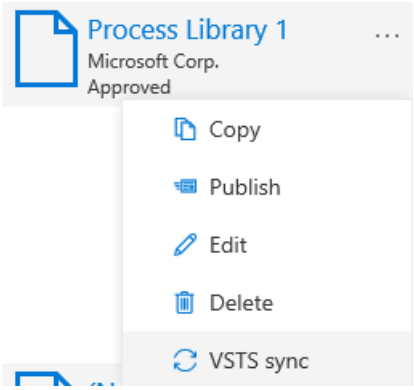
Once you have selected your acceptance testing BPM library, synchronize it with Azure DevOps and create your test plan and test suites.

Sync with Azure DevOps

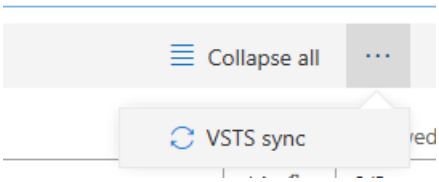
Synchronize your BPM library with your Azure DevOps project. For more information, see Synchronize BPM libraries with Azure DevOps.

After configuration is complete, synchronize the BPM library with a Azure DevOps project.

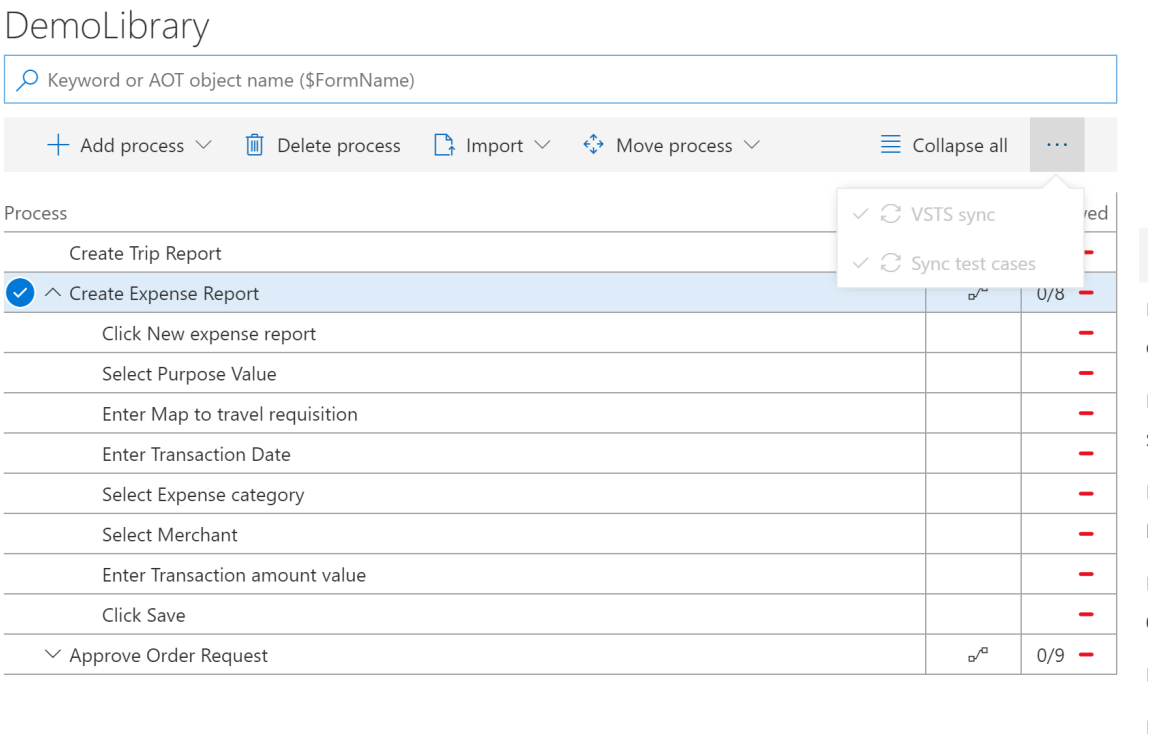
1. On the **Business process libraries** page, on the tile for the library that you want to synchronize, select the ellipsis button (...), and then select **Azure DevOps sync**.



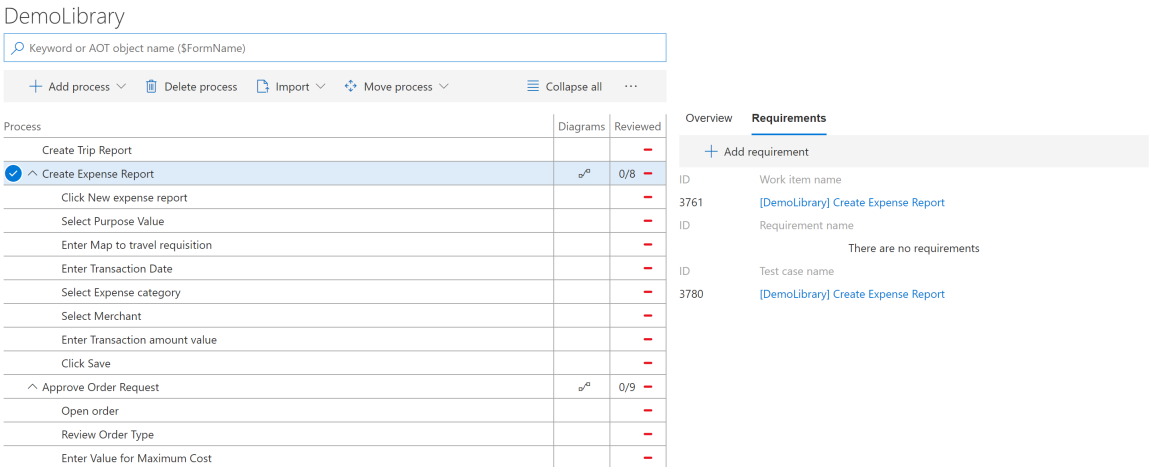
   You can also start Azure DevOps synchronization from the toolbar in a BPM library. Select the ellipsis button (...), and then select **Azure DevOps sync**.



2. After Azure DevOps synchronization is complete, select the ellipsis button (...), and then select **Sync test cases**.



3. When this step is complete, your task recordings will become test cases in Azure DevOps and a link will appear under the **Requirements** tab.

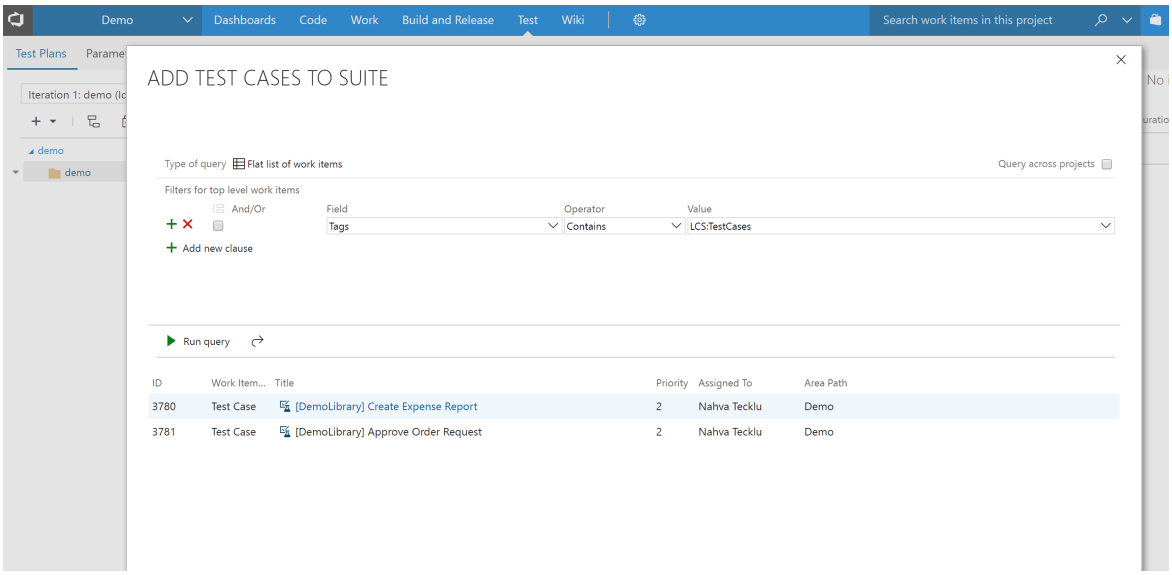


In addition to the test steps, the task recording XML file is attached to the Azure DevOps test case. This file will be needed if you want to automate test execution.

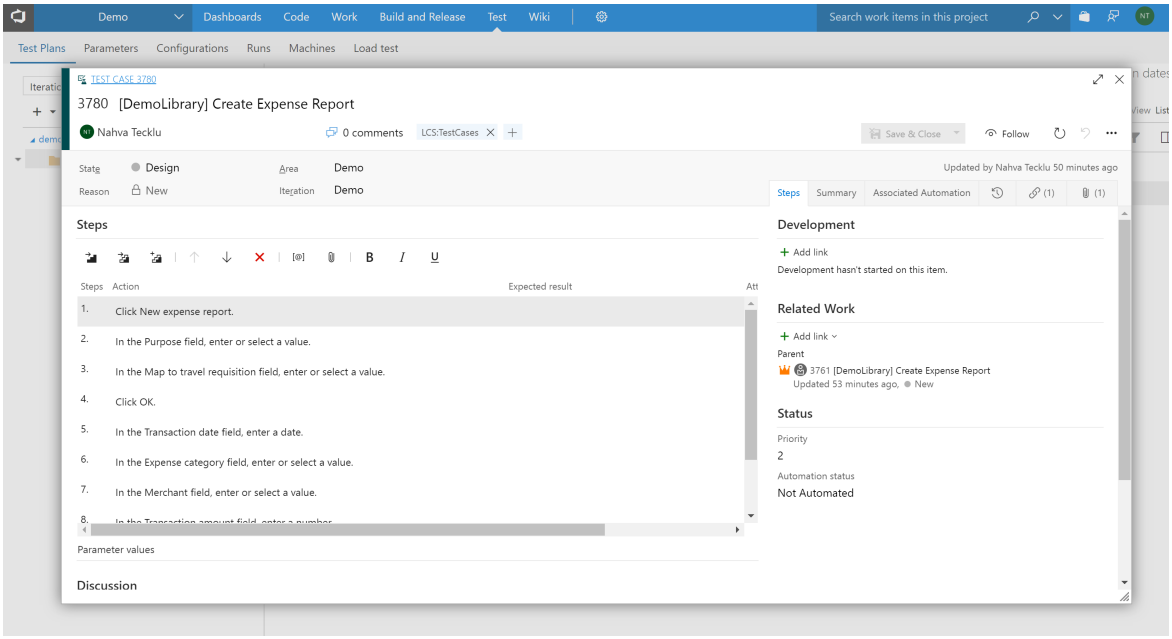
Create a test suite in Azure DevOps

Next, you will need to create a test plan and test suite in Azure DevOps. This will allow you to execute an ordered suite of test cases and easily manage, investigate, and track the results.

1. Sign in to Azure DevOps and select the project and test plan that you want to test in.
2. On the toolbar, select **Test** > **Test Plans**.
3. In the left pane, select **+**, and then select **Static suite**.
4. Enter a name for the suite.
5. Click **Add existing** and query the tag **LCS:Test Cases**.
6. Click **Run** > **Add test cases**.



7. Select the test case to view details and the attached XML file.

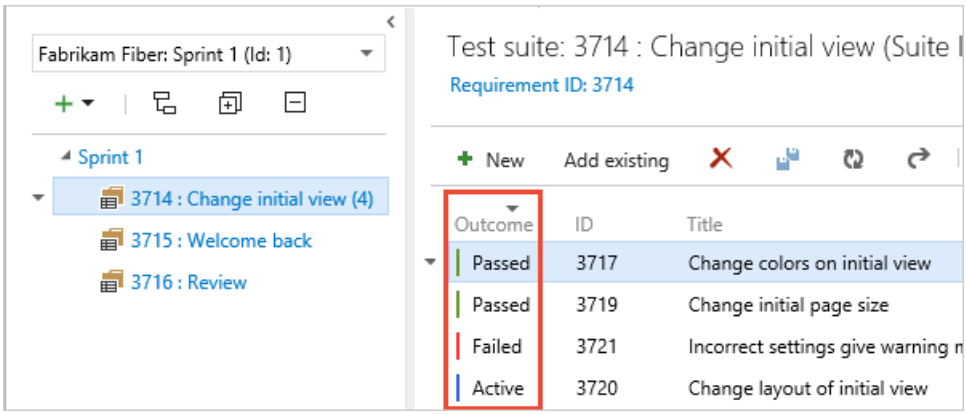


> **NOTE**
>
> This example shows how to create one comprehensive acceptance test suite with all test cases added. Instead, you should create various test suites under the same test plan and then use custom queries to add specific test cases to a test suite. A test case can belong to more than one test suite.

Execute your tests

Run manual test cases

After you have a test suite, you are ready to use it for regression testing after updates have been made to your application in a sandbox or test environment. You can run the test cases in your test suite manually or play the task recordings that are part of the test suite and use Azure DevOps to mark the test cases as passed or failed.



Azure DevOps also provides a tool, **Test Runner**, to manage manual test case execution. For more information about using Test Runner, see Run manual tests.

We recommend that you take advantage of Azure DevOps as it provides a rich set of management features not only for testing, but result management and mitigation.

Run automated test cases

The platform for Finance and Operations provides developers with tools to author test cases based on task recordings and use Azure DevOps to manage the automated execution of these test cases.

Developers can use the build and test automation capabilities of **build and test** environments. For details, see the Continuous delivery home page.

Functional power users can automate the execution of their test cases using the **Regression suite automation tool**. For more information, download the tool and read the Regression suite automation tool.

Investigate test runs

Once an automated run is complete, on the Azure DevOps toolbar, select **Test > Runs** (or **Test Plans > Runs**) to investigate your test run. Select the desired test run to investigate test case failures and errors. You can also go to your test suite in Azure DevOps to see the latest results associated with your test cases. For more information on testing and test management in Azure DevOps, see the Azure DevOps documentation.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Flowcharts in Business process modeler (BPM)

2/18/2021 • 2 minutes to read • Edit Online

IMPORTANT

Flowchart diagrams in Business process modeler have been deprecated. To learn more about the deprecation, see Flowchart diagrams in Business process modeler.

You can use Business process modeler in Microsoft Dynamics Lifecycle Services (LCS) to define and store business process flowcharts for an organization. This topic explains how you can view the default connected flowcharts, export a connected flowchart as a Visio file, and upload and view unconnected flowcharts.

- Connected flowcharts are the automatically generated flowcharts based on data recorded in Task recorder and uploaded to Business process modeler, this also includes the process steps from the task recording.
- Unconnected flowcharts are uploaded directly from Visio.

View a connected flowchart

Default connected flowcharts are available for many nodes in the industry-standard libraries. You can view a connected flowchart to determine whether it meets your needs.

To view a connected flowchart, follow these steps:

1. Sign in to Lifecycle Services, open a project, and then click **Business process modeler**.
2. In the **Project libraries** section, select a library to display it.
3. Expand the business process library and then click a library node that has a flowchart icon associated with it: 

   The flowchart is displayed. Each activity in the process is represented by a shape in the diagram. Process steps are displayed in the right pane.

Export a flowchart as a Visio file

You can export a business process model flowchart to a Visio file.

1. Sign in to Lifecycle Services, open a project, and then click **Business process modeler**.
2. In the **Project libraries** section, select a library to display it.
3. Expand the library and then select any library node that has a flowchart icon associated with it.
4. From the **Overview** pane, select **Diagram** to view the flowchart.
5. From the flowchart tab, click **Export** to save as a Visio file.

Unconnected flowcharts

Unconnected flowcharts, such as a Visio diagram, can be very helpful for describing high-level business processes that are performed outside of the Finance and Operations apps.

Upload an unconnected flowchart

1. In the **Project libraries** section, select a library to display it.
2. Expand the library and then click any library node that has a flowchart icon associated with it.
3. From the **Overview** pane, select **Diagram**.

4. From the **Visio** tab, click **Upload** to upload a Visio file.

> **NOTE**
>
> If you have uploaded the wrong file, you can delete the existing one, and upload a new one to replace it.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Upload custom business processes to Business process modeler (BPM)

2/18/2021 • 2 minutes to read • Edit Online

In Microsoft Dynamics Lifecycle Services, you can record information about custom business processes by using an updated version of Task recorder. You can then upload the files that you record to Business process modeler.

This topic explains where to find the updated version of Task recorder and how to upload the custom business process files that you record. The updated version of Task recorder is available as a hotfix. You can download the hotfix from the following sites:

- Microsoft Dynamics AX 2012 and Microsoft Dynamics AX 2012 Feature Pack – Knowledgebase article 2863182
- Microsoft Dynamics AX 2012 R2 – Knowledgebase article 2863182

For more information about how to work with the updated Task recorder, see Task recorder update for Microsoft Dynamics AX 2012.

Upload custom recorded business processes

You can upload business process artifacts (*.axbpm files) to the business process library. These files are generated from Task recorder. After they are uploaded, you can view and modify the recorded processes in Business process modeler. To upload custom business processes that you recorded, follow these steps:

1. On the **Project** home page, click the **Business process modeler** tile.
2. On the **Business process library** page, click **Upload** in the **My libraries** section or the **Corporate libraries** section.
3. On the **Upload** page, select the industry and enter a name and description for the file that you are uploading. Click **Upload**, select the .axbpm file, and then click **OK**. The upload process can take some time. You can view the status of the upload on the **Administration** page.
4. After the business process file has been uploaded, you can view the business process framework from the **Business process library** page.

Additional resources

Business process modeler (BPM) in Lifecycle Services (LCS)

Business process libraries in Business process modeler (BPM)

Flowcharts in Business process modeler (BPM)

Cloud operations and servicing

2/18/2021 • 5 minutes to read • Edit Online

For customers, partners, and Microsoft to be successful in this endeavor, we must ensure that most of the actions are self-serve with the Microsoft Dynamics Service Engineering (DSE) team managing by exception. To attain this self-serve mode, the Microsoft Product team continues to add more automation around the various features needed to operate an environment.

The Finance and Operations apps are managed services. This means that Microsoft is responsible for managing and operating the production environments. Microsoft's Dynamics Service Engineering team is available 24 hours a day, 7 days a week, and 365 days a year to operate and manage our customers' production systems.

Monitor and troubleshoot the health of your environment

A key tenant for a successful onboarding experience to the cloud service is knowing the health of your environments at all times and being able to troubleshoot health issues when necessary. Lifecycle Services (LCS), which is the admin center for Finance and Operations, contains a collection of monitoring and diagnostics tools which can help ensure that you have an accurate view of the environments that you manage. For more information, see Monitoring and diagnostics tools in Lifecycle Services (LCS).

Update your environment

After go-live, the Production environment must be updated at regular intervals. Lifecycle Services (LCS) provides a self-serve experience to continuously update your environments.

Update types

For customers who are on **Dynamics 365 for Finance and Operations version 8.0 (April 2018) and earlier**, the following updates are available:

- **Platform updates** – A single cumulative binary update of all the platform fixes.
- **Application hotfixes** – Application hotfixes that are released as granular X++ updates.
- **Application release** – A new major release of the application. This type of update typically requires an upgrade.
- **Application customizations** – Customizations that are built on top of the application. The best practice is to apply a single deployable package that consists of all your independent software vendor (ISV) solutions and customizations.

For customers who are on **Dynamics 365 for Finance and Operations version 8.1 (October 2018) and later**, the following updates are available:

- **Application updates** – A single cumulative binary update of the application and the platform fixes. You can update for yourself by using the regular update flows. Otherwise, you will be automatically updated by Microsoft.
- **Application customizations** – Customizations that are built on top of the application. The best practice is to apply a single deployable package that consists of all your ISV solutions and customizations.

Cloud infrastructure

Microsoft is responsible for managing the infrastructure for your environments. Therefore, some updates, such as operating system updates, must be done on a monthly basis in a planned maintenance window. Other kinds of updates might include changes to the infrastructure components.

Update policy

Currently, service updates require production tenant downtime and are applied in two kinds of maintenance windows.

- **Microsoft planned maintenance window** - Microsoft will provide customers with no less than five business days' notice regarding upcoming planned maintenance downtime. The default downtime window is defined per region and is scheduled to occur over a weekend to minimize impact to the business. Cloud infrastructure updates can be done in this window. For more information about planned maintenance, see the Planned maintenance window FAQ.
- **Customer initiated maintenance window** - A customer selects the maintenance window through LCS as a part of the package application flow. Updates are done in this maintenance window.

Search for and apply an update in Lifecycle Services

Updates are applied as deployable package on an environment. A deployable package is a format that is used to apply updates to all the environments in a project. When you encounter an issue in the production environment, you can quickly find and apply a hotfix on all of the environments (Dev/Sandbox and Prod).

- **Search for and download an update** In LCS, you can search for an update using Issue search in Lifecycle Services (LCS) or the Download updates from Lifecycle Services (LCS). Because the steps to prepare an update differ based on the update type, after the update is downloaded, use the following list to determine how to proceed with preparation.
  - Platform update: Platform updates are cumulative and binary. This means that they can be applied directly to an environment. After the update is downloaded, it can be automatically applied to an environment by uploading it to the Asset Library.
  - Application hotfixes: Application hotfixes are code changes. After the application hotfix is downloaded, it must be applied on a dev environment to generate a deployable package. For more information, see Create deployable packages of models and Install metadata hotfixes in development environments.
  - Application customizations: These are customizations that ISV or partners create. These are deployable packages that are uploaded to the Asset Library and can be applied from there.
- **Apply an update** Use the information in the topic, Apply updates to cloud environments, to walk through the steps for applying a deployable package. The update package can be a binary hotfix for Application Object Server (AOS) or a deployable package that was created in your development environment.
- **Validate an update** After an update is applied, you should validate the application to:
  - Ensure that the update addressed the issue that it was applied for.
  - Verify that no regressions occurred from applying the update.
  - Verify that the build information was updated to reflect an update to the binaries.
    - For Platform updates, verify that the version of the AOS Service Model under Microsoft is updated.
    - For Application updates, check the version of the model that included the fix. For example, if the fix was in Application suite, then the version of the Application suite is updated.

Upgrade your environment

For information about how to upgrade to the latest version, see Process for moving to the latest update of Finance and Operations and What's new or changed in Finance and Operations home page.

Environment data management

These are the options for managing databases, including the ability to copy a database from one environment to another or restore a database to a previous state. For more information, see Database movement operations home page.

Sign up for cloud operations notifications

When the status of the package application is changed, LCS sends a notification to all of the users in a project. Any additional stakeholders who should be notified must be specified in the notification list.

1. To add additional stakeholders, in LCS, in the Environment details view, click Notification list.
2. Add the email address of each user who must be notified, and then click Save.

Submit service requests to the Dynamics Service Engineering team

2/18/2021 • 7 minutes to read • Edit Online

A service request is a ticket that you use to request that the Dynamics Service Engineering (DSE) team perform a predefined set of tasks on your environments.

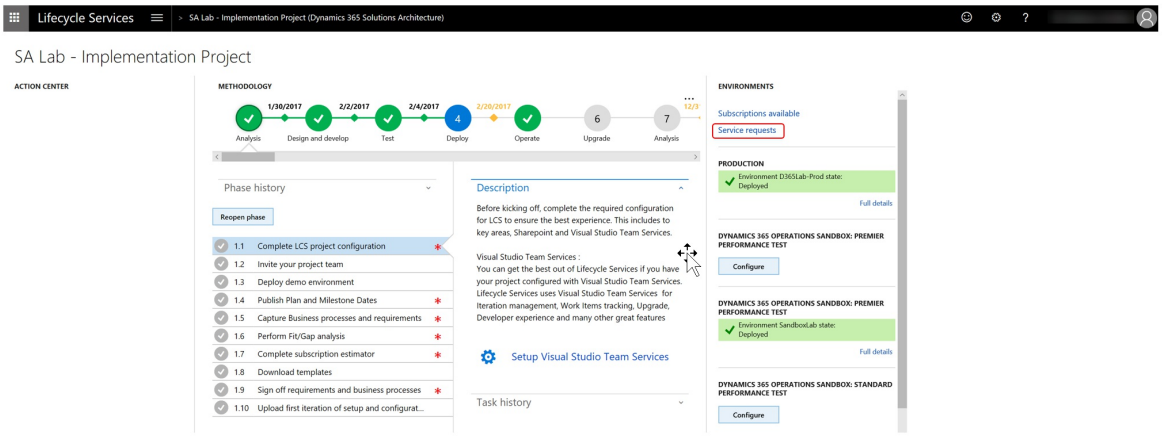
> **NOTE**
>
> Don't use service requests for product issues. If you encounter a situation that doesn't fit into any of the tasks that are described in this topic, submit a support ticket instead. For more information about support tickets, see Get support for Finance and Operations apps or Lifecycle Services (LCS).

You can use Microsoft Dynamics Lifecycle Services (LCS) to submit service requests directly to the DSE team. You can also view which requests have been submitted, executed, and canceled for your environments.

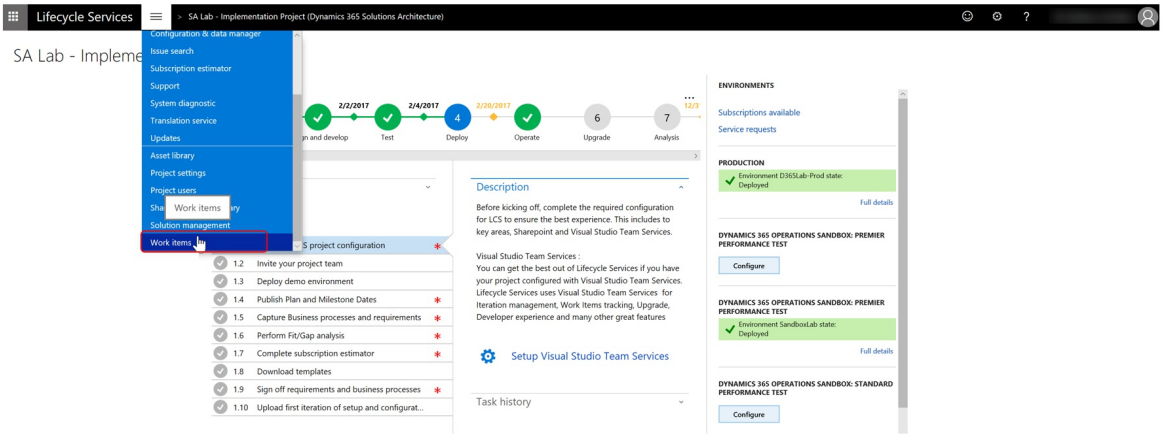
View service requests

There are two ways to view service requests:

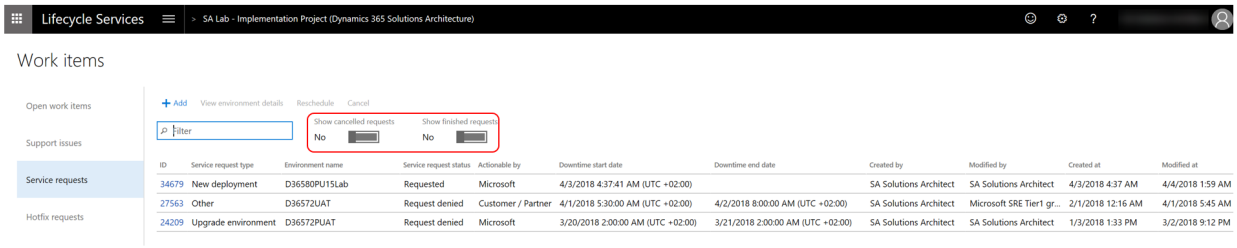
- On the project dashboard, in the **Environments** section, select **Service requests**.



- Select the **Menu** button and select **Work items**. On the **Work items** page select the **Service requests** tab.



By default, the **Service requests** tab on the **Work items** page lists all requests that are currently active and requests that have been denied. However, you can use the filter options to show canceled and finished requests too.



After you submit a request, it has a status of **Requested**. Before the DSE team acts on the request, it might ask for clarification by entering a comment in the **Comment** field. For example, you might receive a comment from the DSE team if you request deployment of a production environment, but the data center differs from the data center where your sandbox environments are deployed. Carefully review the comments, and provide any required clarification in your own comment. To view the details of a specific request, or to submit comments for a service request, select the request ID.

If you signed up for LCS notifications, you receive an email when the status of a service request changes or a comment is entered.

If you submit a service request to the DSE team, and the action is outside the team's scope, the service request will be denied. In this case, the reason for the denial and suggestions for further action are provided. For some typical examples of service requests that the DSE team will deny, see the "Denied service requests" section later in this topic.

Create service requests

There are two ways to create a service request: automatically and on demand.

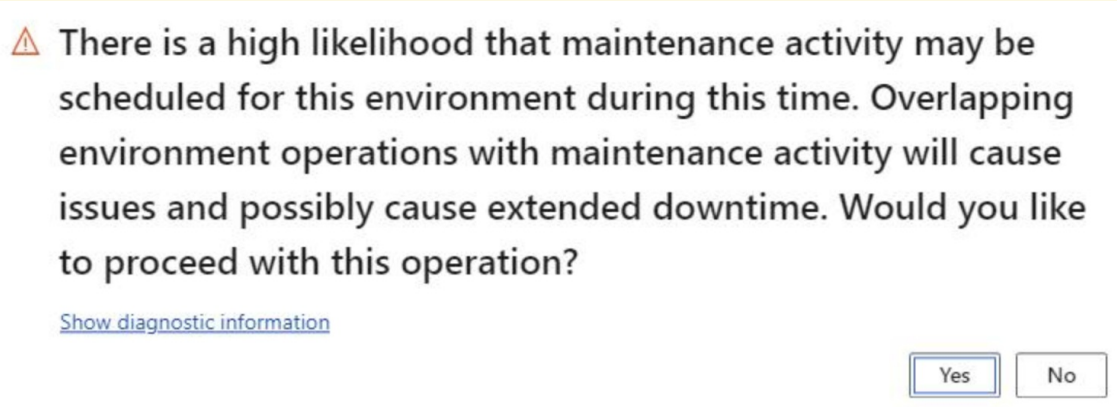
- **Automatically** – A service request is automatically created when you request deployment of an environment, or an application of a package.
- **On demand** – A service request is manually created when you enter a request for a database point-in-time restore, and some other services.

Automatically create a service request

- **Environment deployment** – To set up deployment options and submit a request to the DSE team to deploy a new environment, in the **Environments** section, select **Configure**.
- **Package application** – To apply a package to the production environment, on the **Environment details** page, select **Maintain**, select the package to apply, and then select **Schedule**. For more information, see Apply updates to cloud environments.

IMPORTANT

If your scheduled time overlaps with a planned maintenance window, you will receive the following warning message.



If you choose to continue deploying the package, the package deployment operation will be rolled-back in the event of conflict, as planned maintenance takes priority.

This restriction is appliable to **Microsoft-managed IAAS environments** only.

Create a service request on demand

Service requests that are created on demand aren't explicitly accepted by the DSE team. They will be addressed during the specified downtime window unless the DSE team has entered a comment in the request or has had to deny the request. For details, review the comments in the service request.

Microsoft frequently reviews all incoming service requests. By selecting the correct type of service request for your scenario, you help the DSE team handle the request in a timely manner.

1. On the **Work items** page, on the **Service requests** tab, select **Add**.
2. In the **Create request** dialog box, select the type of service request to create. The options on the page then reflect the specific type of request that you selected.

   - **Sandbox point-in-time restore request** – Select this request type to restore a *non-production* database to a specific point in time. For more information, see Database movement operations home page.

     NOTE

     If you need to restore a *production* database to a previous point-in-time during the cutover phase, select the **Production point-in-time restore request** type. If you need to restore a production database when you're already live in operations, submit a support ticket through LCS.

   - **Database refresh request** – Select this request type to refresh a database from a production environment to a sandbox environment, or from one sandbox environment to another. For more information, see Refresh database. *This request type is being retired on January 31, 2019*.

     NOTE

     If you need to refresh a database from a sandbox environment to a production environment during the cutover phase, select the **Sandbox to Production** type.

   - **Sandbox to Production** - Perform a database refresh of your configuration data to a production

environment during the cutover phase. For more information, see Database movement operations home page.

- **Other request** – You need to use the **Other request** type exactly as described here. If you word a request in a way that isn't clear to the DSE team, the team will enter a comment to ask for clarification, and your request will be delayed. If you use the **Other request** type for any request that isn't listed below, the request will be denied. Select this request type to request that the DSE team perform one of the following actions:
  - Turn on maintenance mode in a production environment. For more information, see Maintenance mode.
  - Tenant move of a live Production environment. Request the Microsoft Service Engineering team to move the Production database and Azure Blob Storage from the old tenant to the new tenant if you are moving tenant on a live Production environment. Make sure that you only request this service when you are ready with all prerequisites. For more details, see Move LCS implementation projects to different Azure AD tenants.
  - Define explicit Internet Protocol (IP) safe list rules in a production environment.

    > **NOTE**
    > Support for explicit safe list rules is deprecated for self-service environments. For more information, see Removed or deprecated platform features.

  - Request that Microsoft Power BI Embedded be activated in a sandbox environment, Standard Acceptance Test environment, or production environment if you receive the following message: "Power BI embedded isn't enabled. Please contact your system administrator."

Commonly denied service requests

Here are some typical examples of service requests that will be denied:

- You submit a request of the **Other request** type for one of the following actions, but you should have submitted a support ticket instead:
  - You want to activate a new subscription estimate after you're live in production or after you've requested a production environment.
  - You want to reset the Financial reporting data mart in a release that is earlier than Microsoft Dynamics 365 for Finance and Operations Financial reporting release 7.2.6.0.
  - You want to restore a production database after go-live.
  - You encountered an issue after the DSE team did an application upgrade.
- You submit a request of the **Other request** type for an action that you should have requested through a different request type. Examples include a database refresh in a non-production environment.
- You submit a request of the **Other request** type for an action that you should perform yourself. Examples include a database upgrade in a development environment.

Service request types and SLAs

SERVICE REQUEST TYPE	APPLICABLE ENVIRONMENTS	REQUESTED SERVICE	LEAD TIME	DOWNTIME

SERVICE REQUEST TYPE	APPLICABLE ENVIRONMENTS	REQUESTED SERVICE	LEAD TIME	DOWNTIME
Environment deployment	Any	Environment deployment	Service level agreement (SLA): within two business days	
Package application	Production	Deployable package application	Five hours	Five hours
Sandbox point-in-time restore	Any Tier 2 or higher sandbox	Database point-in-time restore	Five hours	Four hours
Production point-in-time restore	Production	Database point-in-time restore	Based on data volume	Based on data volume
Sandbox to Production	Tier 2 or higher sandbox to Production	Sandbox to Production	Five hours	Four hours
Other	Production	Maintenance mode	Five hours	Not applicable, because the customer indicates in the service request when the environment should be taken out of maintenance mode again
	Production	IP safe list rules	Five hours	Two hours
	Production	Power BI Embedded	Five hours	Two hours

Planned maintenance window FAQ

2/18/2021 • 5 minutes to read • Edit Online

What is a planned maintenance window?

A planned maintenance window is the timeframe that Microsoft has scheduled to apply infrastructure or service updates to your cloud service.

How does a planned maintenance window work?

For planned maintenance scheduled on your Tier 2 through Tier 5 sandbox environments and production environments, Microsoft will send a notification to all stakeholders **five business days** before the start of the patching window. The patching window is the period when the environment is patched. It's defined by geographic region. Details about the maintenance activity will be included in the notification that is sent to stakeholders. For Microsoft-managed Tier 1 environments, we will not send any notifications before the update.

When is this planned maintenance window taken?

To limit the impact on users, the maintenance window is planned according to the region where environments are deployed. The following list shows the maintenance window for each region. All environments fall into one of these three regions. The times are shown in Coordinated Universal Time (UTC, which is also known as Greenwich Mean Time).

- **NAM:** 2 AM to 10 AM
- **EMEA:** 10 PM to 6 AM
- **APAC:** 12 PM to 9 PM

Will the maintenance from Microsoft require any uptake?

Most of the maintenance operations require no action on your end. If there is a critical security update that requires uptake, you will be notified.

Who will be notified about the upcoming planned maintenance?

The following stakeholders will be notified about the upcoming maintenance:

- Project owners
- Organization admins
- Environment admins
- Other people who are specified in the list during deployment or through the **Notify** button on the environment details page in Microsoft Dynamics Lifecycle Services (LCS)

How do I sign up to be notified about the maintenance window?

Any partner, independent software vendor (ISV), and other interested party who wants to be notified about upcoming updates can request to be added to the LCS project as a relevant stakeholder (project owner, environment admin, or additional stakeholder).

Why can't these updates be applied in zero downtime?

Microsoft is continually working to reduce the necessity of downtime for the service, and many regular maintenance tasks don't incur downtime. However, to help guarantee the most predictability, Microsoft can't yet do all patching in zero downtime.

Microsoft service updates

A separate set of frequently asked questions (FAQ) provides details about service updates that are done by

Microsoft. See One Version service updates FAQ.

Infrastructure updates

How long is the maintenance window?

Most operating system–level updates are completed in approximately one hour. However, Microsoft asks for a three-hour window, so that there is time to handle any failures and to bring the system back to a healthy state.

The exact downtime for all updates will be included in the maintenance window notification email that is sent to you before the start of the update.

How frequent are the updates?

Operating system–level updates are applied monthly to your Microsoft-managed Tier 2 through Tier 5 sandbox environments and to the production environments. However, Microsoft-managed Tier 1 environments are updated weekly in the maintenance windows defined per region.

Where can I learn more about what is applied?

For more information about the updates that will be applied, see Microsoft Security Bulletins.

Where can I track progress of the update?

During operating system–level updates, LCS doesn't currently indicate that any patching is in progress. However, Microsoft plans to add this functionality at some point.

What environments are updated?

Operating system–level updates are applied to all Microsoft-managed environments that are included as part of the Microsoft base offer. This includes your Tier 1, Tier 2 through Tier 5, and Production environments. They are also applied to add-ons that have been purchased. However, other environments, such as environments hosted in your subscription (known as Cloud hosted environments), are the responsibility of the customer or partner.

What notifications will I receive about upcoming planned maintenance?

You will receive a notification email for the scheduled update on your Tier 2 to Tier 5 sandbox environment and production environment, five days before the update is scheduled to occur. This email will include information about the environments that will be updated, the update type, the estimated amount of time that the update will take, and any action that you might have to take. We do not send notifications for Microsoft-managed Tier 1 environments.

Will I be notified when the update is completed?

If your update is completed within the defined maintenance window, you won't receive any notification when the update is completed.

How do I report an issue that is identified during validation of the updates that were applied to the environment?

To report an issue that is identified during update validation, file a support ticket with Microsoft and append the title with 'Planned Maintenance Window'.

What happens if the patching fails?

If the patching fails during an operating system–level update, the specific patch is skipped and will be applied in the next update cycle.

Will I be compensated if the update takes longer than the scheduled maintenance window?

If the update takes longer than the scheduled maintenance window, the extra time is considered unplanned downtime and is subject to the general service level agreement (SLA).

How do I reschedule security maintenance activities?

Security maintenance helps ensure a secure environment, and not doing the maintenance could potentially

introduce an avoidable security risk.

If there is an absolute business need and you are unable to move forward with this maintenance during the timeframe listed above, you can request to reschedule the current maintenance activity for Tier 2 through Tier 5 sandbox environments and production environments by filing a support ticket with Microsoft. The deadline for filing the support ticket to request a reschedule will be included in your notification email. Any request submitted after that deadline will not be honored.

We do not offer rescheduling of security maintenance activities on Microsoft-managed Tier 1 environments.

Monitoring and diagnostics tools in Lifecycle Services (LCS)

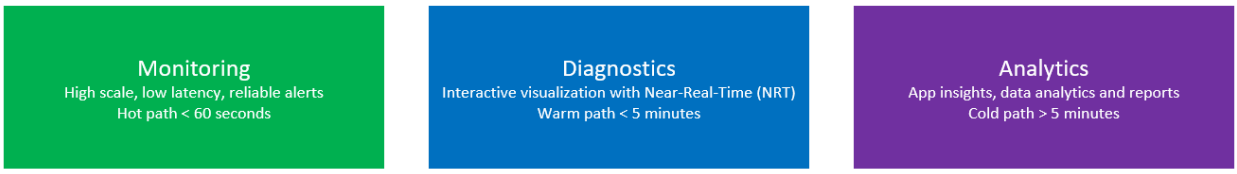
2/18/2021 • 5 minutes to read • Edit Online

This topic describes the various tools that Microsoft Dynamics Lifecycle Services (LCS) provides to help you monitor, diagnose, and analyze the health of the Finance and Operations environments that you manage.

To have a successful onboarding experience to the cloud service, you must know the health of your environments at all times. You must also be able to troubleshoot any health issues that occur. Microsoft Dynamics Lifecycle Services (LCS), which is the administration center, contains a collection of monitoring and diagnostics tools that can help to ensure that you have an accurate view of the environments that you manage.

Telemetry data

The telemetry data that is the basis of the Monitoring and diagnostics portal in LCS has three primary use cases: monitoring, diagnostics, and analytics.



Monitoring

In business operations software, you should always know whether your environment is up and running, so that it can perform business operations. You should also be able to easily view the health of the environment through LCS. Microsoft supports two types of monitoring capabilities:

- **Availability monitoring** – This type of monitoring performs a check against the environment to make sure that it's available at all times. If the check fails, the Microsoft Service Engineering team is immediately notified.
- **Health monitoring** – In addition to availability checks, some basic health checks must be performed. These health checks span various components, such as Application Object Server (AOS), Batch Framework, Data Management Framework, Microsoft Azure SQL, and Management Reporter. These checks are done based on multiple data sources, such as the telemetry that is collected from the environments, checks that are done by a watchdog service that continuously monitors the environment, and CPU counters and other system-level counters that the environment emits. Some health checks are self-healing and are mitigated immediately. However, other health checks are reported to the Microsoft Service Engineering team for investigation.

Diagnostics

When a user reports an issue, you can use various tools in LCS for troubleshooting. The rich set of telemetry data helps you build a storyboard view that shows what that user and other users were doing when the issue was reported. In addition to user activity tracking, a rich set of SQL data is available for performance troubleshooting.

Analytics

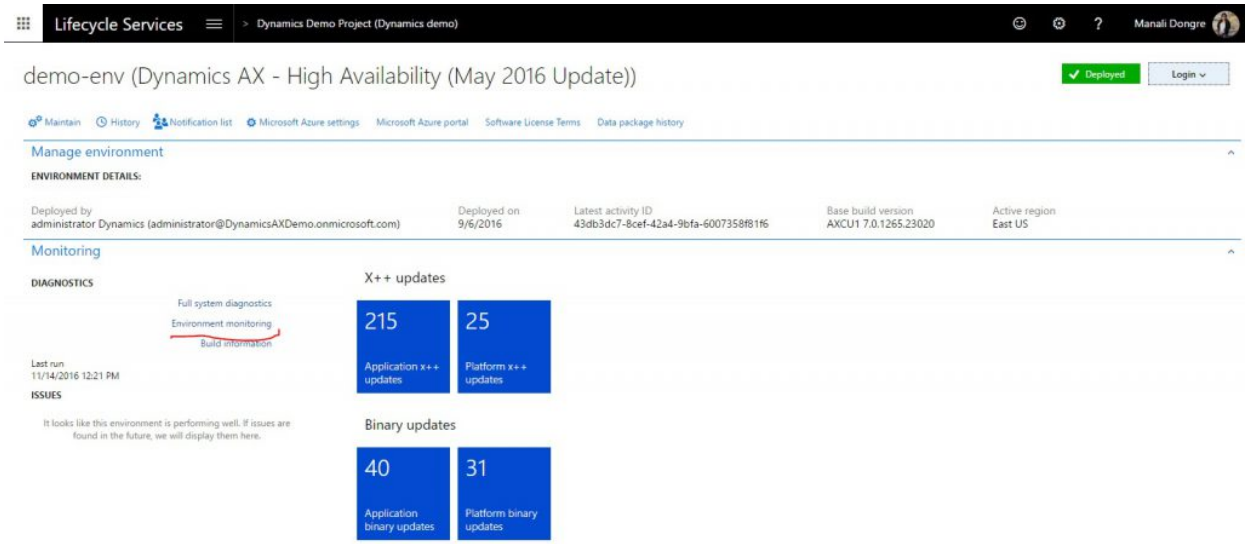
Analytics is another critical use case for the telemetry data that is collected. Currently, only Microsoft can perform analytics, so that it can gauge and understand feature usage and performance through Microsoft Power BI.

Responsibilities

For a managed cloud service such as Finance and Operations, Microsoft is responsible for actively monitoring the health of production environments at all times. If a customer's environment is affected by an issue, the Microsoft Service Engineering team is immediately alerted. The team will start to investigate the issue and will work with you to find a resolution. However, you're responsible for proactively or reactively monitoring and troubleshooting the health of non-production environments.

Access the Monitoring and diagnostics portal

1. Open LCS, and navigate to the appropriate project.
2. In the **Environments** section, select the environment to view, and then select **Full details**.
3. On the environment details page, select **Environment monitoring** to open the Monitoring and diagnostics portal.



Tools

Several tools and resources are available in the Monitoring and diagnostics portal.

> **NOTE**
> Not all environments contain all the tools. The following table shows the tools that are available for each type of environment.

ENVIRONMENT TYPE	TOOLS
Production systems	• Activity monitoring • Environment monitoring • SQL insights • System diagnostics
User acceptance testing (UAT)/sandbox	• Activity monitoring • SQL insights • System diagnostics

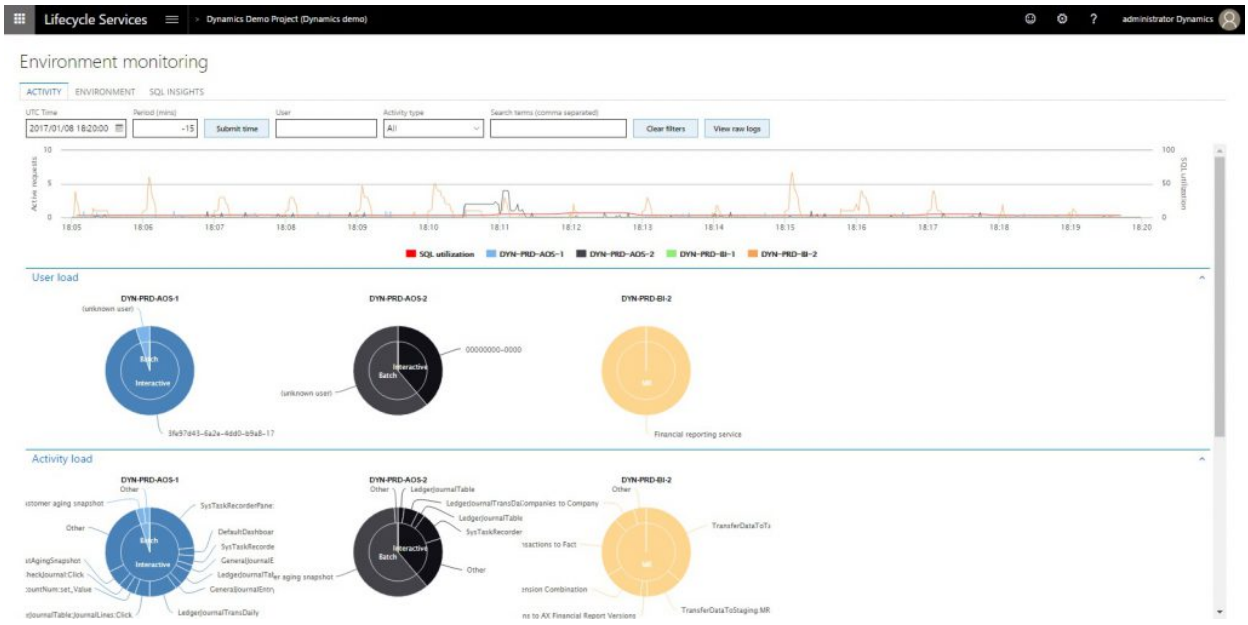
ENVIRONMENT TYPE	TOOLS
Demo/build	• Activity monitoring • System diagnostics
Environments deployed in customer/partner subscriptions	• System diagnostics

Monitoring dashboard

On the **Environment monitoring** page, select the **Health metrics** tab to view the **Monitoring** dashboard. Health metrics are collected for every machine and component. These health metrics include CPU usage, available memory, errors logged per second, and batch heartbeat. You're alerted about any abnormalities in the metrics. Although some alerts are self-healing, the Microsoft Service Engineering team will investigate the cause of other alerts and then take action to mitigate them. You can view the health monitors for a specific area to see what is occurring.

Activity monitoring

On the **Environment monitoring** page, select the **Activity** tab to use the Activity monitoring tool. This tool provides a storyboard view that shows what you or another user was doing during a specific period.



- The **User interaction** chart shows a user's activities on various machines in the environment and the SQL utilization trend.
- The **User load** section shows all the system users. Each chart shows the time that the user spent on a specific machine.
- The **Activity load** section shows the activities that were performed on each machine. If you hover over an activity, you see the Form:Control:Action as a tuple. For example, if you look at LedgerJournal:New:Click in this section, you can see that user A opened the **LedgerJournals** page and selected the **New** button to create a new journal entry.
- The **User activity** grid shows the various activities that users performed, based on their session timestamp.

You can use the filters on this page to narrow the information logs. Here are some of the filters that are available:

- **Time duration** – Go back 60 minutes from the selected date and time.
- **User** – View a specific user's activities.
- **Search terms** – Create a search that is based on the issue that is being investigated.

NOTE

The page doesn't load data by default. To load the data that is required in order to show the page, you must select the time duration and then select **Submit time**.

IMPORTANT

The Activity monitoring tool retains data for only 30 days.

Raw information logs

For advanced troubleshooting, you can view raw information logs. You can use a set of predefined queries to get raw logs for an issue. You can then export the logs to do more advanced analysis. The following types of queries are available:

- Slow queries
- Deadlocks
- Crashes
- Financial reporting issues

SQL insights

The Monitoring and diagnostics portal also includes advanced SQL troubleshooting tools to enable performance analysis. Some of these tools are similar to the DynPerf tool that was used for SQL troubleshooting in Microsoft Dynamics AX 2012. For more details, see Performance troubleshooting using tools in Lifecycle Services (LCS).

Impact analysis report in Lifecycle Services (LCS)

2/18/2021 • 3 minutes to read • Edit Online

IMPORTANT

Functionality noted in this topic is available to targeted users as part of a private preview release. The content and the functionality are subject to change.

The **Impact analysis** report can help you focus your testing efforts for a new release. It provides insight into the Microsoft code that was changed in the release, together with usage information for the production environments that are associated with the project in Microsoft Dynamics Lifecycle Services (LCS). This content provides an overview of the capabilities and also some important considerations. However, the report doesn't provide comprehensive information. Therefore, you should use it in combination with other tools when you're planning test activities for a release.

After accessing a project, the report can be accessed via the hamburger in Lifecycle Services (LCS).

Code churn

One key measure in the report is the amount of code change, or churn, that Microsoft made for selected models in recent releases. Code change measures the number of lines of code that were changed, added, or deleted. The report has a code change measure for each file check-in that contributed to a release.

Code changes can be aggregated to the level of a specific module by using a standardized naming convention that is part of the Microsoft engineering system. In this way, you can determine the relative amount of change in different parts of the product for one or more releases.

In addition to filtering code changes by the release and module, you can filter them by the phase of the engineering cycle. The Engineering branch is where changes are made before the release is made available as a preview. This branch has most of the changes. The Stabilization branch includes changes that are made between the preview and general availability of the release. The ability to filter for changes that were made during stabilization can be useful for organizations that start their release review and testing when the release is made available as a preview.

Usage

The second key measure in the report is the form-based usage of the product in the production environments that are associated with the active LCS project. This usage is measured by summing the user interactions for forms. A user interaction is an event that is logged whenever interaction is triggered between the web client and the service. The most frequent type of user interaction is mouse or keyboard input by a user. However, system-generated interactions can also be counted.

By using the same naming convention that is used for code churn, you can have the usage information summed to the module level. The **Custom** module represents forms that weren't provided by Microsoft.

The usage information isn't tied to the release filters. Instead, it's time-based and has two levels of detail. For module-based visualizations, the data is from the last 95 days of usage in the production environments that are associated with the active LCS project. For form-based visualizations, the data is from the last 35 days of usage in the production environments that are associated with the active LCS project. These time frames were chosen because they offer a compromise between volume of data and sufficient history to help guarantee coverage of

activities that occur only monthly or quarterly.

Important considerations

There are a few important considerations when you use the report:

- The tool uses the number of lines of code that were changed (added, updated, or deleted) as an approximation of change impact for the release. Although there is reasonable correlation between the number of lines of code that were changed and the impact, be aware that any single line of code change can cause impact.
- Not all Microsoft models are included in the code change analysis. Platform changes aren't considered. Additionally, changes in other lower-level models, such as Ledger, CaseManagement, and ApplicationCommon, aren't considered. The models that are included are identified in the code change details in the report.
- Use the report together with the documentation for the release to gain a functional view of the changes.
- The report shows all code changes, regardless of whether the related features are enabled. If a feature is disabled through feature control capabilities, the code changes should have limited impact.
- The production usage is based on user interactions with the web client. Other uses of the system, such as batch processing or integration scenarios, are excluded.
- Code changes from solutions from independent software vendors (ISVs), and code changes from implementation-specific extensions, are excluded.

In summary, the report provides an approximation of code change and usage. In combination with other activities and tools, it can be useful for assessing risk in a release.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Restart environment services

2/18/2021 • 2 minutes to read • Edit Online

You can use the Restart services functionality in Microsoft Dynamics Lifecycle Services (LCS) to restart individual services that are associated with a Tier 2, Tier 3, Tier 4, or Tier 5 standard acceptance test (sandbox) environment that is deployed in a Microsoft subscription. You can use this functionality to restart the following services:

- IIS
- DIXF
- LCS Diagnostics
- Batch
- Reporting Server

Any user who has been added as a project owner, organization admin, or environment manager in an LCS project has permissions to use this functionality.

Restart a specific service

To restart a specific service in a deployed environment, follow these steps.

1. In LCS, open the appropriate project, and select the environment to restart the service for.
2. On the **Environment details** page, select **Maintain** > **Restart services**.
3. In the **Restart a service** dialog box, select the service to restart, and then select **OK**.

   The **Environment state** value is updated when the service is restarted.
4. To view the updated status, refresh the page.

> **NOTE**
>
> Because restart of a service might require only a few seconds, the **Environment state** value might already have been reset to **Deployed**. When the restart is completed, an entry is added to the **History** page.

Stop and start all services

To stop and start **all** services, use the **Stop** menu option followed by the **Start** option on the environment details page.

> **NOTE**
>
> This functionality is only available in Tier-2+ Sandbox environments and not in the production environment.

Report a production outage

2/18/2021 • 2 minutes to read • Edit Online

Lifecycle Services (LCS) has a feature called **Report production outage**. This feature is available to all customers who have purchased one or more Dynamics 365 Finance and Operations apps and have implementation projects with a production environment deployed in LCS. This feature provides a quick and effective channel to escalate issues to Microsoft Support in the event that the services in a production environment are degraded or become unavailable.

Following mutually inclusive conditions, a production outage can be defined as one or more system-wide issues on a live production environment that impact multiple users and prevent your business from performing daily operations.

Reporting flow

The following list shows the order in which an issue should be handled:

1. In a live production environment, a customer experiences an outage or other situation with prevents business from continuing.
2. The customer reports a production outage issue by using the LCS Support portal.
3. The customer selects a production outage issue and provides additional information.
4. A Microsoft support engineer acknowledges the production outage ticket within 30 minutes of submission and begins to immediately collaborate with stakeholders to investigate and resolve the issue.
5. A support engineer contacts the customer to provide a status update.

Access and availability

All users who have been added to a customer's implementation project have access to this feature. This includes project owners, organization admins, team members, and environment managers.

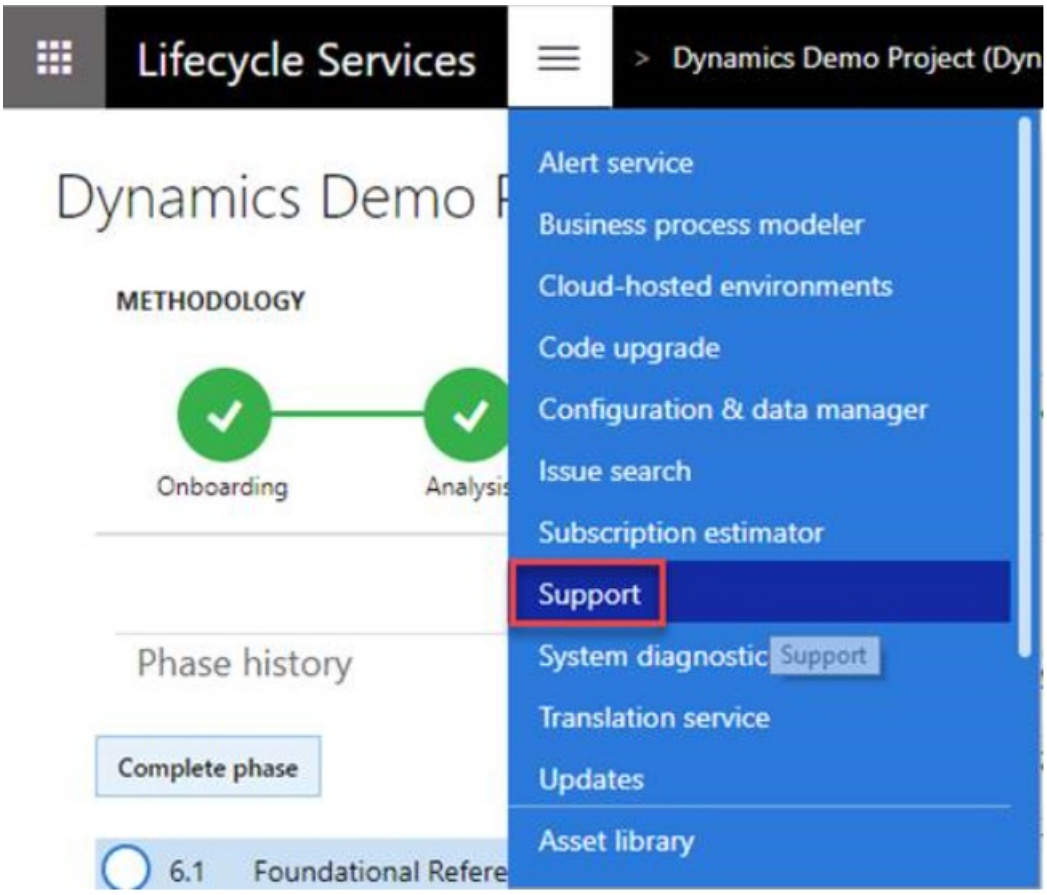
This feature is available for:

- Dynamics 365 Finance
- Dynamics 365 Supply Chain Management
- Environments that are managed by Microsoft
- A production environment in the LCS project
- All support plans

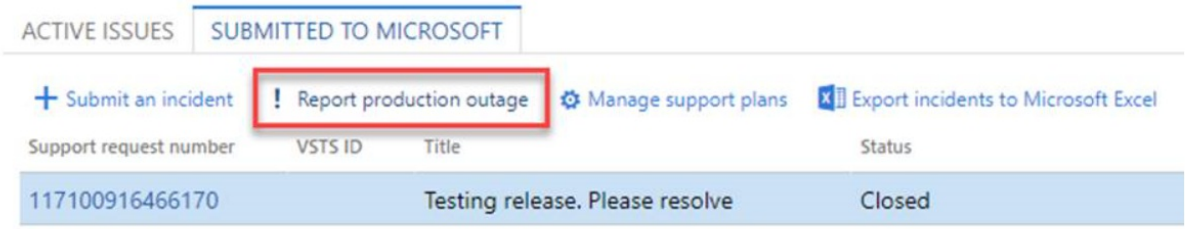
Report a production outage

To report a production outage, follow these steps:

1. Log in to your LCS project.
2. From the hamburger menu, click **Support**.



3. On the **Submitted To Microsoft** tab, click **Report production outage**.



4. Confirm the production outage, select the outage scenario from the drop-down list, and then click **Continue**.
5. Add a title and details about the outage, and then click **Next**.
6. Provide contact information, and then click **Next**.
7. Click **Done**.

If you're unable to report a production outage in LCS, phone support is available.

> **NOTE**
>
> If you don't see your situation listed in the outage scenarios, enter a support incident through LCS. During the initial investigation by a Microsoft support engineer, if it is found that the situation does not meet the current list of production outage scenarios, the support incident will be transferred to the correct support team and service-level agreement (SLA) based on your current support plan.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Track user sign-ins

2/18/2021 • 2 minutes to read • Edit Online

Many organizations are required to maintain an audit trail of users who have used the system. This requirement can be in place for compliance reasons, or to enable trackbacks in the event of incorrect use.

In Microsoft Dynamics AX 2012, the **Audit log** form recorded which users accessed the Microsoft Dynamics AX environment. In Microsoft Dynamics 365 Finance and Operations apps, this information is captured in telemetry. IT administrators can download this information by using Microsoft Dynamics Lifecycle Services (LCS) and then move it to offline storage to maintain the audit trail of users who have signed in.

To generate an audit log of users who have used the system, follow these steps.

1. Sign in to LCS, and open the project that is associated with your implementation.
2. Navigate to the production environment, and open the **Environment details** page.
3. On the **Monitoring** tab, select the **Environment monitoring** link to open the monitoring dashboard.
4. On the **Activity** tab, select **View raw logs**.
5. In the **Query** field, select **User Login Events**. You see a time duration that has a start date that is set to **End date - 7 days**.
6. Set the end date, and then select **Search**. The search results that are returned include all users who signed in to the system during the seven days before the selected end date.
7. The search results show the **AADUserID** value and the sign-in start and end times of the user's session. To map the **AADUserID** value to the user's user name and email address, use the **Users** page (**System administration** > **Users**).
8. To export the records and keep them for a longer period, select **Export grid**.

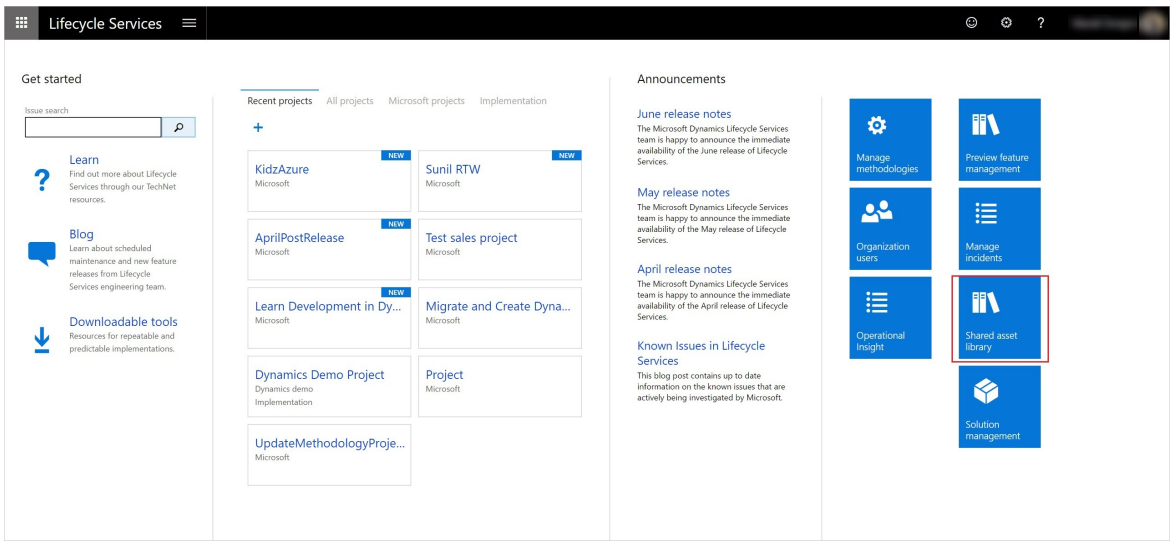
To help guarantee a complete audit trail, an IT administrator must complete this procedure every seven days.

Asset library in Lifecycle Services (LCS)

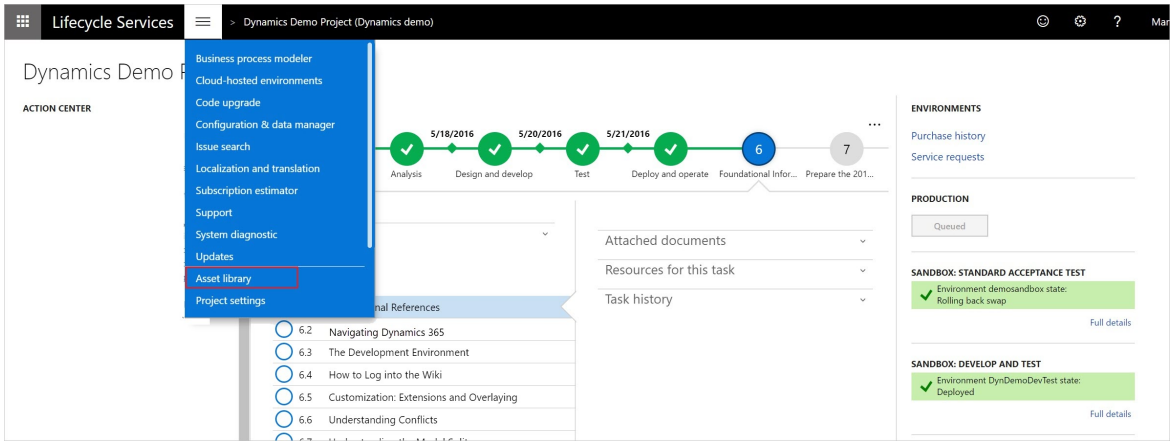
2/18/2021 • 4 minutes to read • Edit Online

The Asset library is a storage location for the various assets that are associated with a tenant in Microsoft Dynamics Lifecycle Services (LCS). Two types of Asset library are available in LCS: the Shared asset library and the project-level Asset library.

- **Shared asset library** – The Shared asset library is used by Microsoft and Partners to share assets across multiple tenants, projects, and environments in LCS. This library can be accessed by any user who signs in to LCS. To access the Shared asset library, sign in to LCS, and then click the **Shared asset library** tile.



- **Project-level Asset library** – The project-level Asset library is used to share assets across environments within a project in LCS. This library can be accessed by all users within a project. To access the project-level Asset library, sign in to LCS, and open a project. Then, on the hamburger menu, click **Asset library**.



> **NOTE**
>
> Uploading versions for the same asset in the project asset library is not supported.

Asset library support

The Asset library supports multiple types of assets. Here are some asset types that are frequently used:

- **Software deployable package** – This asset type represents all the packages that are used to update an environment with the latest set of updates.
- **Data package** – This asset type stores assets that are used for configuration and data management.
- **GER Configuration** – This asset type stores all localization and translation assets that are applied to the client.
- **Retail SDK** – This asset type stores all the latest scripts for the Retail software development kit (SDK).
- **Database backups** - This asset type is used for import and export of databases from Sandbox Tiers 2 - 5 environments.

Asset scopes

Every asset that the Asset library supports has multiple scopes. Here are some of the supported asset scopes:

- **Me** – When an asset is uploaded, it's set to the **Me** scope. An asset that has the **Me** scope is visible only to the person who uploaded the asset.
- **Project** – When an asset is imported from the **Global** scope to another project, it's set to the **Project** scope.
- **Organization** – When an asset must be shared with multiple users within a tenant, the tenant admin can promote the asset to the **Organization** scope.
- **Global** – Only Microsoft can upload assets to the **Global** scope. These assets are assets that Microsoft wants to be made publicly available to all LCS projects and users.

Asset status

Every asset has one of two statuses: **Draft** or **Published**.

- **Draft** – The asset can still be edited.
- **Published** – The asset is published at an **Organization** or **Global** scope, and edits are completed.

Actions in the Asset library

You can perform various actions in the Asset library as you require.

Upload an asset to the Asset library

1. Select the tab to upload the asset to.
2. Click the plus sign (**+**).
3. Enter a name and description for the asset.
4. Upload the file for the asset, and then click **Confirm**.

Upload a new version for a specific asset (Shared asset library only)

1. Select the asset in the Asset library.
2. On the toolbar, click the **Upload new version** button.
3. Repeat the steps in the previous procedure, "Upload an asset to the asset library."
4. On the toolbar, click **Versions** to view multiple versions for a single asset.
5. Individual versions can then be imported in to a specific project asset library as required.

Move assets from the Shared asset library to the project-level Asset library

There are two ways to move an asset from the Shared asset library to the project-level asset library: you can import the asset or copy it.

Import from the Shared asset library

Follow these steps to import an asset from the Shared asset library to the project-level Asset library so that it can be applied across environments.

1. In the project-level Asset library, select the tab for the asset type to import.

2. Click **Import**.
3. In the list of assets in the Shared asset library, select the asset to import, and then click **Pick**.

The selected asset is imported and put into the project-level Asset library. The status of the asset in the project-level Asset library is set to **Published**. This method is for packages that you don't plan to edit. If you want to edit an imported package, create a copy by using the following procedure. The status of the package will then be **Draft**.

Copy from the Shared asset library

Follow these steps to create a copy of an asset so that it can be edited.

1. In the project-level Asset library, select the tab for the asset type to copy.
2. Select the asset to copy, and then, on the toolbar, click **Copy**.

A copy of the published asset is created, and the status is set to **Draft**.

Save to my library

After you've edited an asset, follow these steps to move the edited asset back to the Shared asset library so that it can be promoted to the **Organization** scope and shared with multiple customers.

1. In the project-level Asset library, select the tab for the asset type to import.
2. Select the asset to save, and then click **Save to my library**.

The asset is saved from the project-level Asset library back to the Shared asset library, and the scope is set to **Me**.

Performance troubleshooting using tools in Lifecycle Services (LCS)

2/18/2021 • 5 minutes to read • Edit Online

This topic describes how you can troubleshoot and mitigate performance issues using the tools available in Microsoft Dynamics Lifecycle Services (LCS).

Overview

Common feedback from customers and partners has been that they are unable to successfully diagnose performance issues using the tools in LCS. We have addressed this feedback by creating a more reliable way to collect performance metrics on demand. This enables customers and partners to execute a predefined set of actions that can be used to mitigate issues in a sandbox or production environment. This feature queries SQL Server directly, so you get query store metrics in near real-time. We have also added an audit trail on the action performed so that you can easily determine who performed the action and when it was performed.

Details

All SQL performance tools in LCS are available under the **SQL Insights** tab on the **Environment Monitoring** page for a specific environment. The following tabs are available:

- **Live View** – Shows current DTU, executing statements, and blocking statements. The current **SQL Now** page that shows performance issues will be replaced with **Live View**.
- **Queries** – Shows a list of predefined queries that can be used to retrieve metrics on demand. Examples of queries include a current blocking tree, a list of active plan guides, and a list of most expensive queries.

  > **IMPORTANT**
  > To help guarantee that the query results are returned instantaneously, most of the queries are run synchronously. However, if there is an ongoing performance issue, synchronous query execution might cause a time-out error. To address this issue, a new **Use Fast Query** option has been added. By default, this option is turned on for most queries. If you receive a time-out error after you run a query, turn the **Use Fast Query** option off, and then try to run the query again. The query will now run asynchronously.

- **Actions** – Shows a list of predefined actions that should be taken to mitigate issues in the sandbox and production environments. Examples of actions include adding/dropping an index, updating stats on a table, rebuilding indexes, and terminating a blocking statement. Any time that an action is performed, the environment history for an environment will show a record for the action performed. A history record is created only for actions and not when queries are executed.
- **Performance Metrics** – Shows the most expensive queries that were run in the system during the selected period, based on logical I/O, execution count, duration, CPU time, and wait count. This data is queried from the SQL query store. The data is retained for 30 days, and the tool runs its data collection every day at a random time between midnight and 4 AM in the time zone in which your environment is hosted. The last run date and time is visible from your environment details page in Lifecycle Services, under the **Monitoring** tab in the **Last run** field. To use the tool, select a period during the last 30 days. When the query results appear, select the bar in the duration chart to highlight where the query falls based on other metrics. On the **Statement** tab, you can either view the query or download the query

execution plan.

- **Index Analysis** – Shows aggregated index and table information, based on user scans, user seeks, user updates, and row count. Like performance metrics, this tool shows the trend for the selected index along with additional table metrics.
- **Queries** tab and **Actions** tab – For details about the queries that are shown on the **Queries** and **Actions** tabs, see the Query cookbook.

How do I use this feature?

1. Go to your project in LCS and open the environment details page. Select the **Environment Monitoring** link in the **Monitoring** section. Select the **SQL Insights** tab to access this feature.
2. You can navigate to each of the tabs (**Live View**, **Queries**, **Actions**, **Performance Metrics**, **Index Analysis**) to view or query for more information.
3. You have the option to search or export to Excel any of results from the query execution.
4. After you have narrowed down the reason for the performance issue, you can use a predefined action to mitigate the issue.
5. After an action is performed, an entry is made on the **Environment History** page, which shows the details of the action, the parameters that were passed in, a timestamp, and who triggered the action.

Sample flow

Scenario: Users report slow performance when using the system. One issue could be a blocking statement. Blocking by itself is typical in a healthy system and is only a problem when it becomes excessive or starts degrading business activities.

1. Go to the **Live View** tab and check if there are any blocking statements. If there is a blocking statement, copy the blocking query ID.
2. Open the **Queries** tab and select the **Current Blocking Tree** query. This will return the root blocker that is blocking the SQL operation.
3. To resolve the issue, you can either let it run and clear naturally, or end the process for the lead blocker, which will roll work back. Typically, you should only end the lead blocker process if you think that it will not clear naturally (such as a bad query plan), or in situations where a critical process is unable to run and needs to complete immediately.
4. Confirm that it's okay to terminate the statements that are currently being executed.
5. Open the **Actions** tab and select the **End SQL Process** action and pass in the root blocker query ID. This will execute a query against the SQL database to terminate the blocking statement.
6. Go to the **Queries** tab and run **Current blocking query** to verify if the blocking statement was terminated.
7. You can also check the **Environment History** page to see details on what process was terminated.
8. To avoid this issue in the future, you should use indexes or plan guides, or turn off lock escalation, or use page locks if processes are blocking each other while operating on different records. If processes are operating on the same records, the only way to avoid blocking is by refactoring or rescheduling the processes to not operate on the same records at the same time.

Query cookbook

2/18/2021 • 22 minutes to read • Edit Online

This topic provides details on each query under the **SQL Insights** tab on the **Environment Monitoring** page in Lifecycle Services (LCS) and how they should be used when troubleshooting performance issues. For details about this feature, see Performance troubleshooting using tools in Lifecycle Services (LCS).

Current blocking

Description

Lists any currently blocked queries, and also the SPID that is blocking them, how long they have been blocked, and what resource they are waiting on. This can be used in conjunction with the query to see the blocking tree, which provides a graphical overview of some of the same information. Blocking by itself is normal in a healthy system and is only a problem when it becomes excessive or starts degrading business activities.

Next steps

- Determine which process is blocked, and which process is blocking it and why.
- To resolve blocking, the only two options are to let it run and clear naturally, or to end the lead blocker process, which will roll back work. Generally, the lead blocker should only end in situations where it is not believed that it will clear naturally (such as a bad query plan), or in situations where a critical process is unable to run and needs to be completed immediately.
- To avoid the same blocking in the future, you can use indexes or plan guides, or disable lock escalation and page locks if processes are blocking each other while operating on different records. If processes are operating on the same records, the only way to avoid blocking is by refactoring or rescheduling the processes so that they do not operate on the same records at the same time.

Current blocking tree

Description

Provides a graphical view of the SPIDs and statements that are currently causing blocking or being blocked. This can be used in conjunction with the current blocking query to see more detailed information. Blocking by itself is normal in a healthy system and is only a problem when it becomes excessive or starts degrading business activities.

Next steps

- Determine which process is blocked, and which process is blocking it and why.
- To resolve blocking, the only two options are to let it run and clear naturally, or to end the lead blocker process, which will roll back work. Generally, the lead blocker should only end in situations where it is believed that it will not clear naturally (such as a bad query plan), or in situations where a critical process is unable to run and needs to be completed immediately.
- To avoid the same blocking in the future, you can use indexes or plan guides, or disable lock escalation and page locks, if processes processes are blocking each other while operating on different records. If processes are operating on the same records, the only way to avoid blocking is by refactoring or rescheduling the processes so that they do not operate on the same records at the same time.

Current DTU

Description

Provides a detailed breakdown of the current DTU by component. DTU is a percentage measure of total SQL load and is reported as the maximum value of any of the subcomponents. Use this query to determine which subcomponent is causing high DTU. High CPU is often related to bad query plans. High Data I/O is often related to large imports or exports. A high Log Write percentage is often related to bulk processes, such as reporting, which make changes to large numbers of intermediate records.

Next steps

- If high CPU is the cause of high DTU, view the currently running queries or most expensive queries for the last hour. To address the issue, get better query plans by adding indexes, changing the query, or, as a last resort, adding plan guides.
- If high Data I/O is the cause of high DTU, look at currently running queries and also current batch jobs, and try to determine what might be writing large volumes of data. Consider scheduling those processes to run outside of business hours.
- If high Log I/O is the cause of high DTU, look at currently running queries and also current batch jobs to determine what might be doing large amounts of intermediate processing. Often, high Log I/O is caused by the use of real fully logged tables as pseudo-temp tables during processing, such as in reporting. Address this by running those processes outside of business hours or by changing the processes.

Currently running queries

Description

Provides a list of all queries that are currently in a state of being executed or blocked on this database, and also the total execution and wait times of each query. Queries that have high execution time and low wait time are often indicative of bad query plans. Queries with high wait time and low execution time are indicative of blocking. If relatively fast operations are being run many times, sometimes they can be found by running this query multiple times in a row and looking for commonly occurring queries with fast execution time.

Next steps

- If high CPU time is seen, get the query plan for the query, and also see whether other query plans that have been used for this query are more efficient. Consider addressing the issues with a new index, with a change to the query, or, as a last resort, by adding a plan guide.
- If high wait time is seen, view the current blocking and current blocking tree to determine why the query is blocked. This is occasionally addressed by disabling lock escalation or page locks if that is the cause of the blocking. More often, it is addressed by segmenting the work that is being performed to ensure that the same record is not processed by two queries at the same time.

Get fragmentation details

Description

Fragmentation is when the records are not stored contiguously inside of the page. When there is unused space between records on a page, which occurs through data modification that is made against the table and to the indexes defined on the table, this is SQL index fragmentation. Dynamics 365 Finance and Operations apps use SQL Azure DB premium SKUs which are backed by SSDs. This means that the fragmentation does not cause the same level of issues as on a database that is backed by a SCSI drive, but it still it could cause slower performance. With higher fragmentation, there is a higher chance that it could affect the performance of queries that use this fragmented table/index.

Next steps

- This query shows the list of tables, indexes with their fragmentation percentage, and a recommendation to either REBUILD the index or REORG the index. When you rebuild or reorg an index it will pack the records next to each other and avoid gaps in between, thus reducing the fragmentation.

Get index details

Description

Provides details about all indexes on a given table. This query is usually used before adding new indexes, to ensure there are no existing indexes that should be altered instead.

Next steps

- If you are adding a new index, verify that there is not an existing index that should just be adjusted instead. Each new index adds overhead on all write operations. Therefore, new indexes should be added sparingly.
- Existing indexes can occasionally cause blocking if they support page locks, and large processes running at the same time are locking pages that contain records for each process. Disabling page locks should also be done sparingly, because it will degrade write operations that would have benefitted from page locks.

Get lock details

Description

Provides a list of all locks held by a given SPID. This is useful when a blocking tree shows that a SPID is part of a blocking chain. Sometimes, this query will reveal that page locks or lock escalation are the cause of the blocking. Other times this query is useful for seeing how many keylocks in given tables are being held.

Next steps

- If page locks or lock escalation are causing issues, they can be disabled. However, this should be done cautiously, because there are valid cases where lock escalation and page locks are needed for good performance.
- If a SPID is just holding large numbers of locks, reducing the amount of data processed within one transactional scope can reduce the number of locks and therefore the chances of experiencing blocking and deadlocks.

Parameters

The SPID can be obtained either from the list of currently running queries or through the list of current blocking queries. No historical information about SPIDs is available because the SPID ID is only valid while the SPID is currently being executed.

List of current plan guides

Description

Provides a list of all plan guides currently installed in the database.

Next steps

- You can determine whether a query plan was created based on a plan guide by looking in the query plan XML. If the plan was generated based on a plan guide, the plan guide name will be specified in the XML.
- If a plan guide is installed but is not being used, verify that the SQL statement text and parameter list for the plan guide are identical to the actual query. Any differences, even in whitespace, will stop a plan guide from being used.

Get list of query ID's

Description

If query text is known to be causing a problem, this query can be used to find the query ID for that query. This can occur if the SQL statement is showing up in an error message, in tracing, or in telemetry. After the query ID for the statement is found, it can be used for other operations, such as finding query plans or applying plan guides.

Next steps

- After the query ID for a given statement is found, it can be used to find previous plans for that query so that it can be optimized.

Parameters

- Provide a substring of the query statement, such as "%FROM GENERALJOURNALENTRY T1%".

Get XML for a given plan ID

Description

This provides the SQL plan in XML format for a given plan ID.

Next steps

- Save the XML as a .sqlplan file, and open it in SQL Server Management Studio or other plan reading tools. From the query plan, you can often find optimizations, such as index additions, that will be beneficial.
- The plan might have an index recommendation listed, but be cautious when applying those indexes. SQL will often recommend creating indexes with excessive numbers of columns to optimize a single query, which can have a negative impact on other queries or the system as a whole. View these recommended indexes as hints to begin analysis, not as something to be directly applied.

Parameters

- The plan ID can be obtained from a variety of sources, such as currently running queries. If you only have a query ID, you can use "get a list of query plans for a query" to figure out which plan IDs have been used for a query.

Get query plan and execution stats

Description

Provides a list of plans and execution statistics for a given query. This is useful for determining whether a query is obtaining multiple query plans with different performance characteristics over time.

Next steps

- If a query is getting multiple query plans with vastly different performance characteristics, start by retrieving and analyzing the plans to determine what is different and why one is far better than others.
- In order for SQL to use a better query plan, the best solution is to make index changes or add query hints directly in code.
- If indexes and query hints are not enough, as a last resort, a plan guide can be added to force the better plans always to be used. Be careful when using a plan guide, because sometimes a query plan is fast for some parameter values but overly slow for other parameter values.

Parameters

- The **query ID** can be obtained from multiple other operations, such as currently running queries, current blocking queries, most expensive queries, and query by statement text.
- The **time interval** in terms of hours will often be specified as "168" to show all plans for the last week or "24" to show all plans for the last day. Often, plans will change multiple times per week but might be stable for a few days. Therefore, selecting larger intervals here is often advantageous.

Get wait stats

Description

Provides information about what a given query plan has waited on in a given time range. This is useful when a query regularly takes longer than it should, and you have been unable to determine whether it is CPU time or

blocking. Similarly, if you see blocking but need to know what kind it is, this will provide that information.

Next steps

- If a plan is primarily waiting on CPU, the plan is just inefficient and needs to be fixed through the addition of indexes or hints, or by restructuring the query.
- If it is waiting on lock, the query is experiencing blocking. This is more difficult to determine unless you can see the blocking happening actively through the blocking tree query. In this case, the next step is to evaluate the records that are being edited and try to determine what other processes could be editing the same records at the same time. Often, the same query running in multiple threads at the same time will block itself.
- If it is waiting on I/O or latch, there are likely either large volumes of data being transferred or large numbers of small transactions. If possible, perform writes in small batches instead of just one record per transaction.

Parameters

- The plan ID can be obtained from a variety of sources, including currently running queries and the most expensive queries. If you have a query ID, you can get plan IDs by using the query to get a list of plans for that query.
- Ideally, the time range should target a specific time when an issue was occurring. For instance, if DTU was high for an hour, you should target just that hour of execution.

List most expensive queries

Description

Provides a list of the most expensive queries during the specified period. The decision about which attribute should be considered when computing most expensive queries can also be provided. This is often used when high DTU is seen for a period, to determine which query is causing the high DTU. In order to use this query effectively, it is important to determine which DTU component is being addressed.

Next steps

- View the most expensive queries by total CPU to find queries that are causing high DTU utilization. These queries are usually fixed through indexes, query changes, or, as a last resort, plan guides.
- View the most expensive queries by total duration to find queries that might be slowing down the user experience. If these queries also have high total CPU, fix the query plans. If they have low CPU, it is likely the result of blocking.

Parameters

- For the number of hours, specify "0" if you are looking at currently occurring issues, such as currently high DTU. This will provide results for the current hour. If you are looking for generally high DTU over time, specify either "24" for the last day or "168" for the last week.
- For the ordering, select "total cpu" to find high DTU impact for recurring queries. Select "average duration" to find queries that are not run often but that might be slow. Select "total duration" to find recurring queries that might be slowing down users.

Create non-unique index on table

Background

Indexes should generally be created through the standard metadata package and database synchronization. This ensures that Sandbox and PROD, and any other environments, always share a common definition. Occasionally, it is necessary to install a new non-unique index at runtime to mitigate an ongoing outage or issue where a regular deployment would take too long. This action can be used to install that non-unique index.

Next steps

- After specifying the index to create, the next step should be to migrate that same index definition to the next

available official metadata release.

- Duplicate indexes generally do not cause any noticeable issues. Therefore, leave the manually created index in the system until after the official index has been completely rolled out.

Parameters

- The index name is any unique string not currently in use by an existing index on this table. For easy identification, it is recommended but not required that the index name follow the format PERF_IDX_<tablename>_<number>. For instance, an index might be named PERF_IDX_LEDGERJOURNALTRANS_1.
- The columns parameter is a comma-separated list of physical column names in the order that they should appear in.
- The included columns parameter is a comma-separated list of physical column names in the order that they should appear in. If no included columns must be specified, leave this parameter blank.

Create a plan guide to force Plan ID

Description

SQL will recompute a new plan each time the plan cache is flushed, such as when an update of statistics runs. The plan that is chosen is based on "sniffing" the parameters of the first execution of that query. After that, the same plan will be used, regardless of parameters. For this reason, the same query might sometimes get multiple plans, some of which are far worse than others for given data distributions. This action can be used to force SQL to use a specific plan, regardless of which parameters are sniffed. Note that this action applies only to the database that it is executed on. Therefore, it should be used as a last resort, after the addition of indexes, hints, or code changes that will flow between development and production environments.

Next steps

- After the list of plan IDs and execution statistics for a specific query ID are found, the SQL plan for each ID should be downloaded and analyzed. After the best plan is determined, this script can be used to force that plan to be used, regardless of parameters.

Parameters

- The force_plan_id parameter is the ID of the plan that should be used going forward. This can be found from the query to get all plan IDs for a given query. Be sure to validate that the XML of the plan is a good plan before forcing, because forcing a bad plan can have negative effects.
- The name parameter is any unique string not currently used by an existing plan guide.
- The statement parameter is the SQL statement that should be forced. This can be obtained from many other queries, including currently running queries and most expensive queries. Ensure that this is copied precisely, including whitespace, because differences in whitespace will cause the plan guide not to work. Also, do not include the parameters for the query in this text, because they are specified separately. For instance, on a select query, the first characters of the statement should always be "SELECT".
- The params parameter is the set of @P1, @P2, and so on, parameters for the query. These can be obtained from other queries, including currently running queries and most expensive queries. When pasting in the parameters, remove the outer parenthesis. For instance, paste in "@P1 int,@P2 nvarchar(21)". If there are no parameters for the query, leave this value blank.

Create a plan guide to add table hints

Description

SQL will recompute a new plan each time the plan cache is flushed, such as when an update of statistics runs. The plan that is chosen is based on "sniffing" the parameters of the first execution of that query. After that, the same plan will be used, regardless of parameters. For this reason, the same query might sometimes get multiple plans, some of which are far worse than others for given data distributions. This action can be used to add

specific hints to the query. This is different than forcing a plan guide, because SQL will still perform parameter sniffing and occasionally select different plans. However, the query hints will be applied in those plan computations. Note that this action applies only to the database that it is executed on. Therefore, it should be used as a last resort, after the addition of indexes, hints, or code changes that will flow between development and production environments.

Next steps

- Usually, query hints are determined after looking through multiple different query plans for a given query. For example, if an index seek on a table always outperforms a scan, it might be beneficial to add a FORCESEEK hint.
- After hints have been determined and tested locally, use this action to install a plan guide that will add those query hints to future executions of the query.

Parameters

- The name parameter is any unique string not currently used by an existing plan guide.
- The statement parameter is the SQL statement that should be forced. This can be obtained from many other queries, including currently running queries and most expensive queries. Ensure that this is copied precisely, including whitespace, because differences in whitespace will cause the plan guide not to work. Also, do not include the parameters for the query in this text, because they are specified separately. For instance, on a select query the first characters of the statement should always be "SELECT".
- The params parameter is the set of @P1, @P2, and so on, parameters for the query. These can be obtained from other queries, including currently running queries and most expensive queries. When pasting in the parameters, remove the outer parenthesis. For instance, paste in "@P1 int,@P2 nvarchar(21)". If there are no parameters for the query, leave this value blank.
- The hints to apply can be tested locally by just manually adding them to the end of a query. If table aliases are used, they should match the table aliases in the statement text.

Drop index

Background

Usually, extra indexes on a table only add minor incremental costs. But in some situations, such as high-throughput tables, the existence of an extra index can cause large performance degradation. The correct way to remove indexes is to remove them from metadata. But if they need to be removed more aggressively to mitigate an ongoing issue, this action can be used.

Next steps

- From the query to list all indexes for a table, determine the index name to drop.
- Verify that no other workloads will be negatively affected by removing this index. This is a dangerous operation to perform and might cause other workloads to regress.
- Provide the table and the index name to drop.
- Apply the same change in other environments, such as DEV and Sandbox, ideally by removing the index in metadata to keep all environments in sync.

End SQL process

Background

If a SPID is consuming too many resources and degrading the operation of other processes, it might be beneficial to end the SPID process. This will cause the open transaction to roll back, meaning that data should not be lost, but the process might need to be manually restarted. Note that rollback can also take a long time and consume a lot of resources if the transaction has already performed a lot of work. Therefore, this action should be used with caution.

Next steps

- From the blocking tree and other queries, determine which SPID should end.
- Verify that the processing that is being performed by the SPID can end without causing harm to ongoing business operations.
- Provide the SPID number to end, and roll back that operation.

Disable/enable page locks and lock escalation

Background

In many situations, SQL will either take page locks (locking a full page at a time) or escalate to entire table locks (locking the whole table). This enables SQL to perform actions that edit a high volume of rows more efficiently. However, this can cause problems and unnecessary contention in certain workloads. Consider two large posting jobs, each running for 10 minutes and editing different rows of data that happen to reside on the same physical data page. This might cause the two postings to serialize and to take 20 minutes because of the contention. This query can be used to disable lock escalation and page locks for a given table. It should be used cautiously, however, because there are valid reasons for SQL to use page locks and lock escalation.

Next steps

- After determining that unnecessary contention is being experienced on a given table because of lock escalation or page locks, use this query to disable those options.

Parameters

- The table name is the physical name of the table.
- The value of the lock escalation and page locks is either "ON" or "OFF".

Rebuild index

Background

Over time, as processes edit existing records and write new records into tables, indexes might become fragmented. This increases the amount of I/O necessary to process the same number of records, and might cause performance degradation. This operation will rebuild indexes to reduce page fragmentation. This is an expensive operation and has been known to cause contention while running. Therefore, the environment should be monitored while this operation is running.

Next steps

- Monitor blocking and DTU in the system while this operation is running. If the system degrades unacceptably, consider ending this operation.

Remove plan guide

Background

If plan guides have been installed and are having a negative impact, or if they just are not working, it might be necessary to remove a plan guide. This query enables removal of an existing plan guide. Note that removing a plan guide will not affect any currently running queries until the next time they are executed.

Next steps

- After determining which plan guide should be removed, specify the name of the plan.

Parameters

- The list of existing plan guides and associated names can be found from the list of current plan guides query.

Update statistics on table

Background

Updates statistics on the specified table. Occasionally, statistics can be found to be out of date and to be causing query plan issues, such as after an import process that has skewed the statistics of the table. Note that this action will have a significant performance overhead while running and has also been known to cause blocking on queries that use the same table. Therefore, it should be used sparingly and monitored while running to ensure that the system does not degrade.

Next steps

- While running this query, monitor both current blocking and current DTU to ensure that system performance does not degrade unacceptably. If it does, consider ending the update statistics command.

Parameters

- The table name parameter is the physical name of the table to update statistics for.

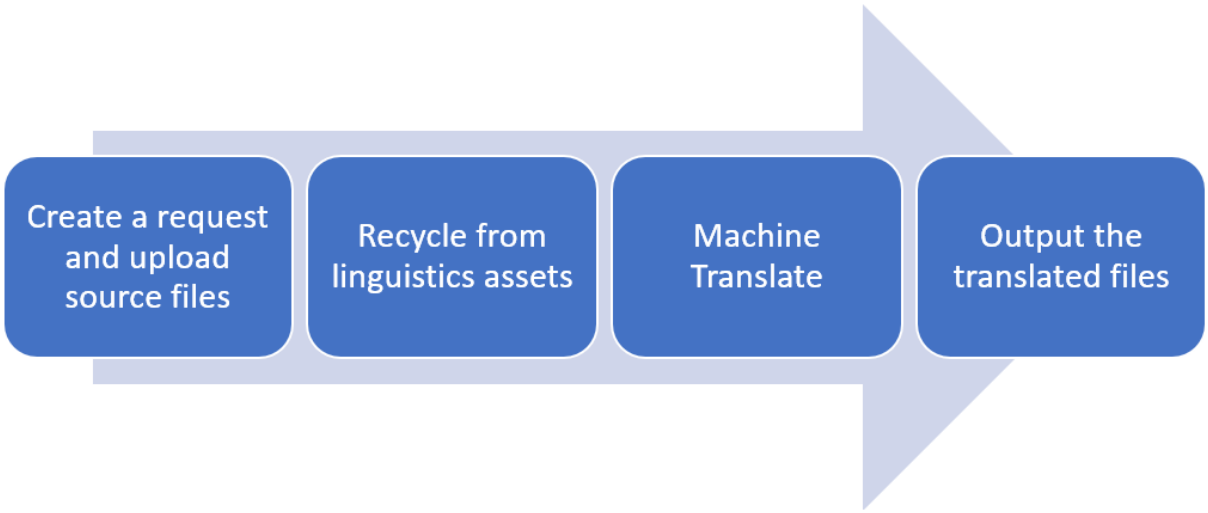
Dynamics 365 Translation Service overview

2/18/2021 • 4 minutes to read • Edit Online

The Microsoft Dynamics 365 Translation Service (DTS) is hosted in Microsoft Dynamics Lifecycle Services (LCS). It's designed to enhance the experience for partners and independent software vendors (ISVs) when they translate their solutions or add a new language for supported Dynamics products.

DTS uses a machine translation (MT) system that is custom-trained for Microsoft General Availability (GA) languages to maximize the quality of the translation output. DTS also supports translation recycling from the linguistic assets of Microsoft Dynamics and partners/ISVs. Therefore, identical strings are translated one time and then consistently reused.

The following illustration shows, at a high level, how the service works.



Recycling existing translations

Existing linguistic assets can be recycled only when the assets are uploaded in a zip file that contains translation memory (TM) files that use Localization Interchange File Format (XLIFF). For more information, see Translation memory files.

Custom-trained MT system

DTS uses a Microsoft Translator service and a custom translator to customize Microsoft Translator's advanced neural machine translation for Microsoft Dynamics products. The custom-trained MT system can only be used for GA languages, unless partners upload XLIFF TM files that contain more than 10,000 translation units (TUs). (A TU typically contains a source string, translation, state, state qualifier, and note.) In those cases, DTS creates a custom-trained MT system that is specific to the translation request that the XLIFF TM files are submitted for.

> **NOTE**
>
> Microsoft Translator supports the text translation through the Microsoft Translator Text API. DTS uses V3 Translator API because V2 will be discontinued on April 30, 2019 with the retirement ofMicrosoft Translator Hub. For information about the supported languages for V3, see Language and region support for the Translator Text API.

Supported products

DTS currently supports the following product versions.

PRODUCT NAME	VERSIONS	SUPPORTED FORMAT FOR USER INTERFACE FILES	SUPPORTED FORMAT FOR DOCUMENTATION FILES	NOTES
Microsoft Dynamics AX 2012	All versions	.ktd, .ald	.docx	
Dynamics 365 Finance and Operations apps	All versions	.label.txt	.docx, .html	.txt is the specific label format and .html is the custom help solution format.
Microsoft Dynamics 365 Commerce	All versions	.label.txt	.docx	
Microsoft Dynamics CRM	All versions	.resx	.docx	
Microsoft Dynamics NAV	All versions	.etx, .stx, .resx, .txt, .xml, .xlf	.docx	.txt and .xml are the NAV specific format, and .xlf is the Business Central extension resource format.

Accessing DTS

You can access DTS in two places in LCS:

- From the LCS home page
- From within an LCS project

Accessing DTS from the LCS home page

Sign in to LCS, and scroll to the right side of the page. Expand the tiles waffle, and then select the **Translation service** tile to open the dashboard view for DTS.

Accessing DTS from within an LCS project

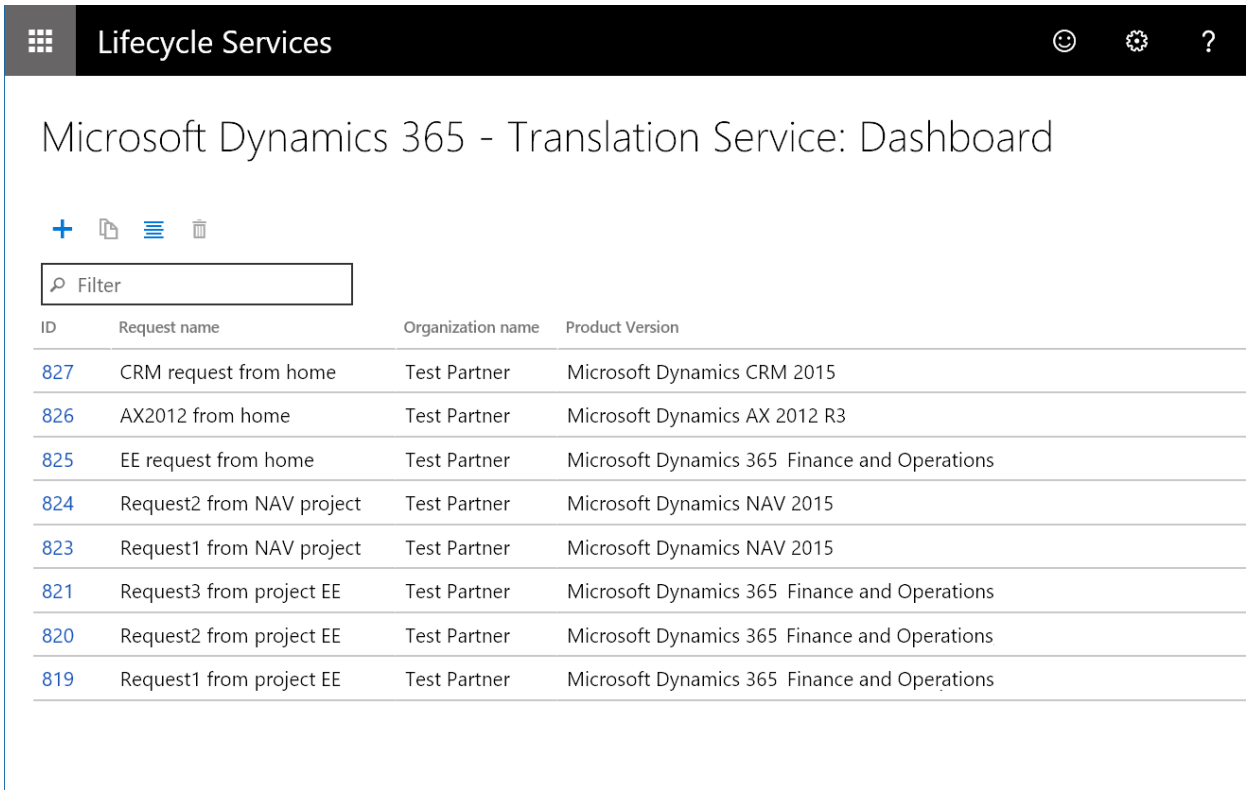
Create a new project, or open an existing project. On the project dashboard, in the **More tools** section, select the **Translation service** tile. Alternatively, on the project dashboard, select the **Menu** button, and then select **Translation service**.

Accessing DTS from the LCS home page vs. accessing it from within an LCS project

When you access DTS from the LCS home page and create a translation request, you can select the product that is used for the request. To add more requests that use different products, you can just change the product selection. You don't have to close the service and open a different translation project.

This option is convenient when you work on multiple product translation projects. However, because you access the service outside an LCS project, no other users can view your requests on the DTS dashboard. Instead, this option gives you your own DTS dashboard that shows all the translation requests that you've made from within all LCS projects and from the LCS home page.

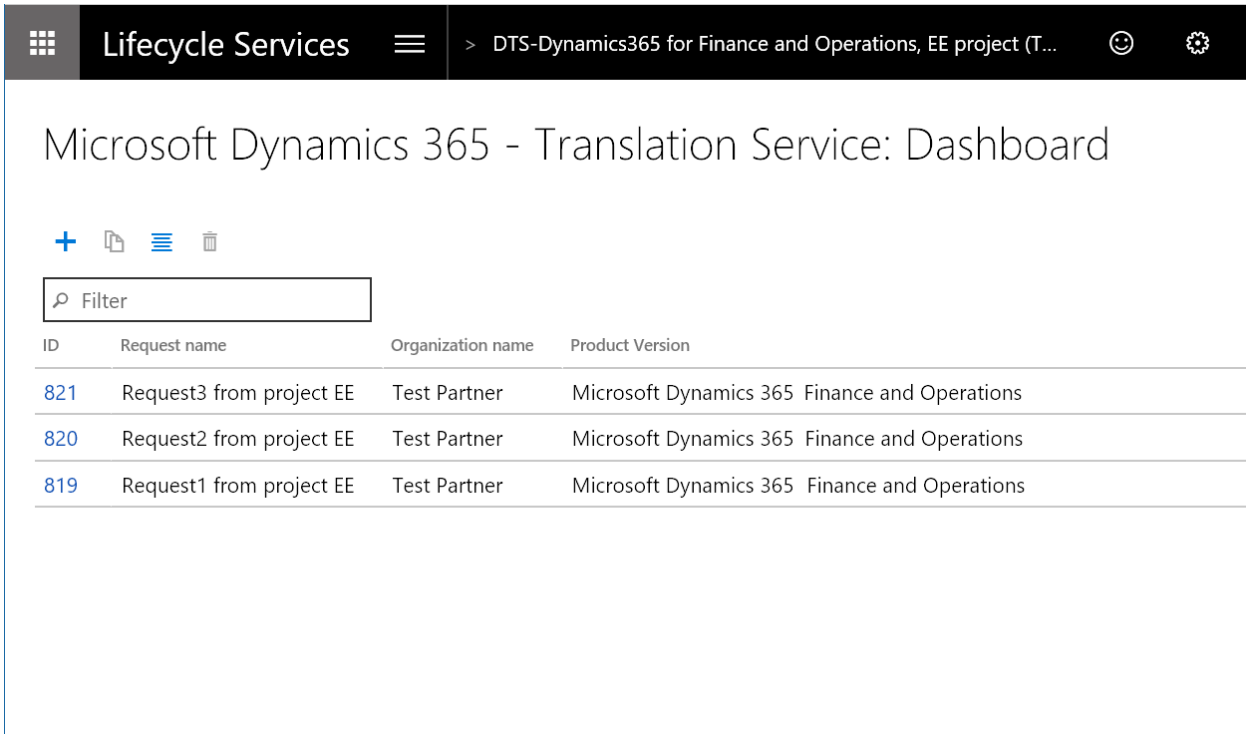
The following illustration shows an example of the DTS dashboard that you open from the LCS home page.



Because an LCS project is always tied to a product, any translation request that you submit from a project automatically carries the product type and version information from the project. You can't select a different product for the request.

In an LCS project, the project owner and the users will have permission to access the DTS dashboard and the translation requests that are submitted from within that project. Therefore, this option is useful when you work with a group of people on one product translation project in LCS.

The following illustration shows an example of the DTS dashboard that you open from within an LCS project.



Accessing LCS preview features

LCS offers some services or features only as preview features for various reasons. To view the list of preview features that are available, on the LCS home page, select the **Preview feature management** tile. To turn on a

feature, select the feature, and then set the **Preview feature enabled** option to **Yes**.

Two preview features are available for DTS:

- **Dynamics 365 Translation Service - Documentation Translation Support** – You must turn on this feature if you want to translate a product or solution document (for example, a Microsoft Word document).
- **NAV product availability** – You must turn on this feature if you want to create an LCS project for NAV products and access DTS from within the project.

Glossary

TERM	DESCRIPTION
XLIFF	XML Localization Interchange File Format. XLIFF is an XML-based format. It was created to standardize the way that localizable data is passed between tools during a localization process, and to serve as a common format for files that are used by computer-aided translation (CAT) tools.
Microsoft GA languages	General availability of the Microsoft-produced languages. The list varies, depending on the product.
TU	Translation unit. A TU typically contains a source string, translation, state, state qualifier, and note.

For more information about how to use DTS, see Translate user interface files and Translate documentation files.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Translate user interface files

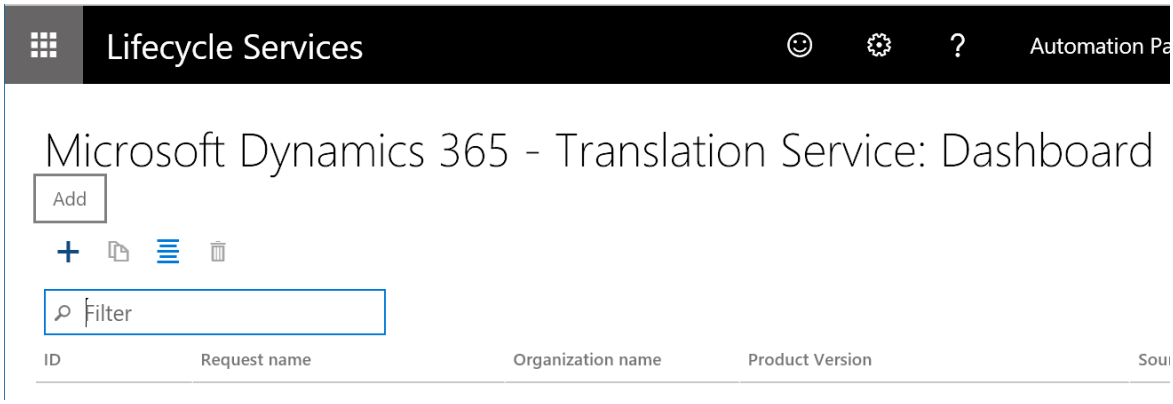
2/18/2021 • 6 minutes to read • Edit Online

This topic provides information about how to translate a user interface (UI) file for Microsoft Dynamics products or solutions.

For more information about the Microsoft Dynamics 365 Translation Service, see Dynamics 365 Translation Service overview. For information about how to translate a documentation file, see Translate documentation files.

Create a translation request

1. In Microsoft Dynamics Lifecycle Services (LCS), on the DTS dashboard, select **Add** to create a new translation request.

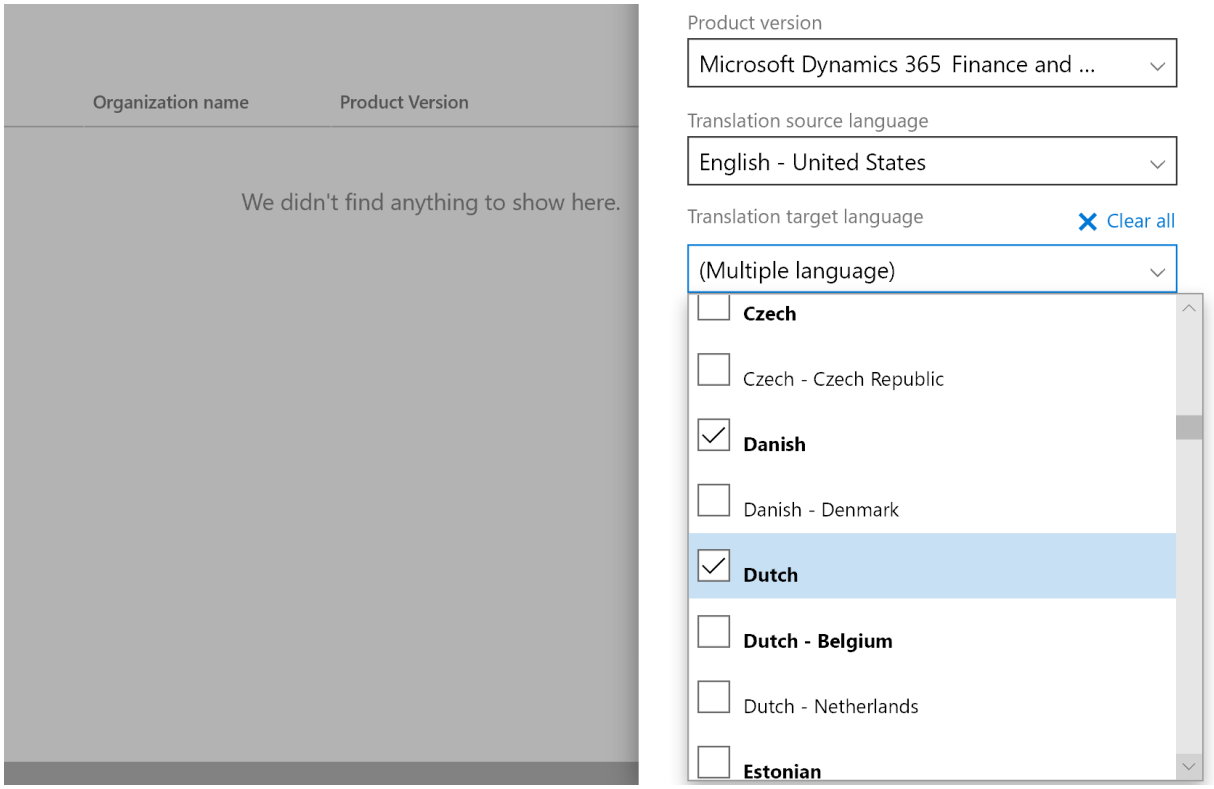


   You can open the DTS dashboard either from the LCS home page or from within a project. For more information, see Accessing DTS.

2. Enter the required information for the request.

FIELD	DESCRIPTION
Request name	Enter a name for the request.
File type	Select **User Interface**.
Product name	Select a product name. If you accessed DTS from within an LCS project, this field is automatically filled in and is read-only.
Product version	Select a product version. If you accessed DTS from within a LCS project, this field shows the default product version information from the project. However, you can select a different version.

FIELD	DESCRIPTION
Translation source language, Translation target language	Select the set of source and target languages to translate from and to. If your business requires that multiple target languages be translated for the same source language, you can select all of the target languages in one request. Select the target languages using the check box next to the language name. This saves time and allows you to track the status of all the target language translations in one request. The fields list all the languages that are supported for the selected product name and version. Language names that are shown in **bold** type are General Availability (GA) languages for Microsoft Dynamics products. This means that Microsoft-trained machine translation (MT) systems are available in those languages and the MT system is trained on the terminology for Microsoft Dynamics. For non-GA languages, the MT system uses the general domain training.



3. Select **Create**. Verify the request details were selected correctly and then click **Yes** to continue.

NOTE

To take advantage of the Microsoft-trained MT system for Microsoft Dynamics linguistic assets, you must select **English – United States** as either the source language or the target language. Here is an example.

TRANSLATION SOURCE LANGUAGE	TRANSLATION TARGET LANGUAGE	MT SYSTEM THAT IS USED
English – United States	Japanese	Microsoft-trained MT system
Japanese	English – United States	Microsoft-trained MT system
German	Japanese	Generic MT system, unless the user provides a translation memory (TM) that uses XML Localization Interchange File Format (XLIFF) and has more than 10,000 translation units (TUs)

Upload files

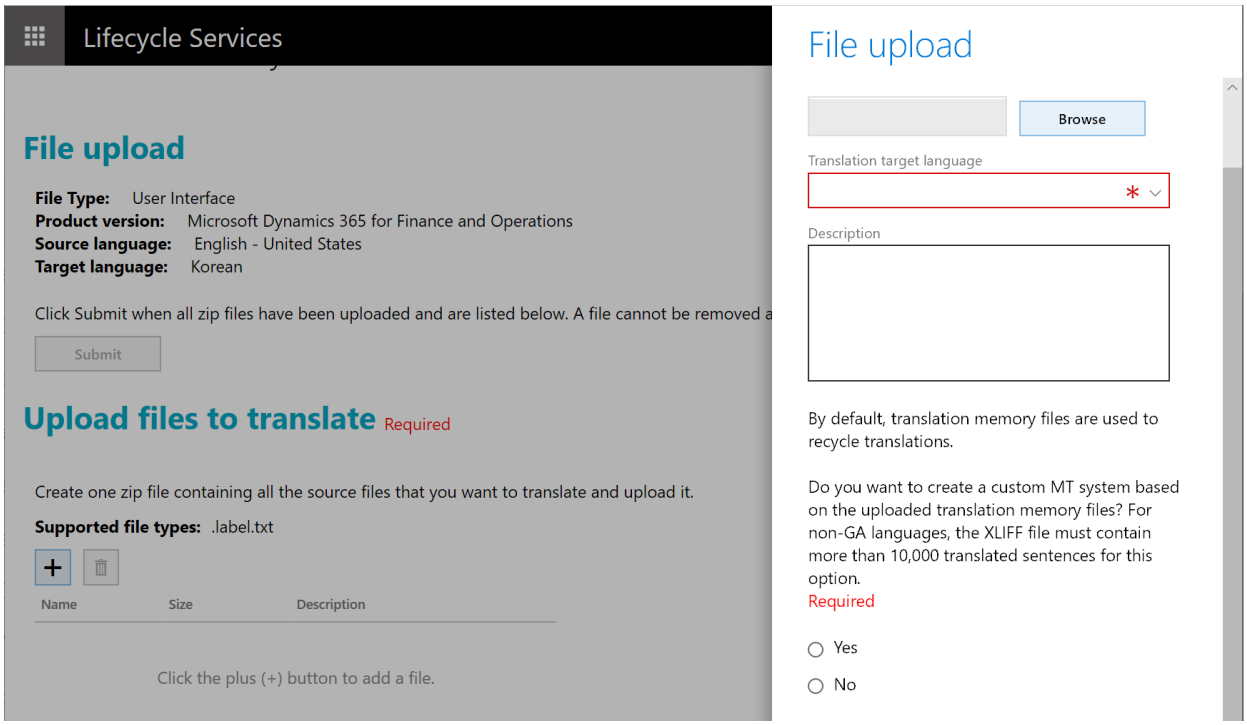
Select the plus sign (+) in each section to open the **File upload** page.

Upload the files to translate (Required)

Create one zip file that contains all the UI files in the source language that you want to translate from. The zip file can include different file types, provided that the file types are supported for the product. For more information about supported file types, see Supported products. Note that DTS doesn't change the source files that you upload. The source files are only used to create files in the corresponding target languages you requested.

Upload XLIFF translation memory files (Optional)

If you have XLIFF TM files from a previous UI translation request, or if you used the Align tool to create an XLIFF TM, create a zip file that contains all TM files before you upload them. Strings that match are then recycled to help guarantee consistency between product versions. For more information about XLIFF TMs, see Translation memory files.



If you created the translation request for multiple target languages, you must select which target language the TM file is for.

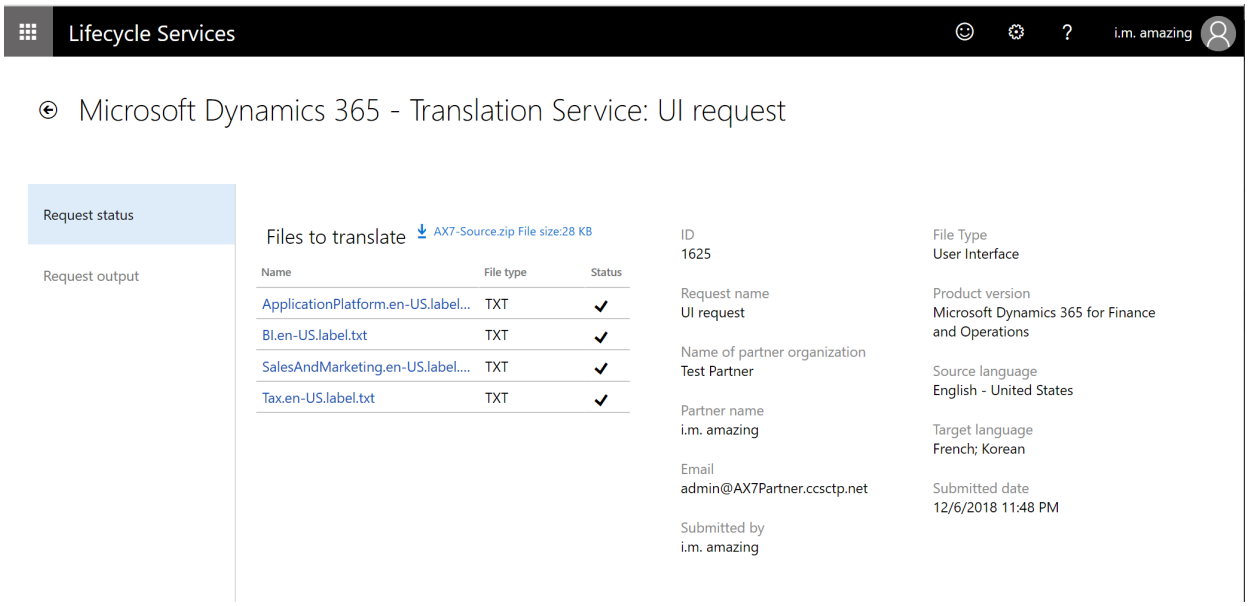
With the translation memory file you are providing, you have an option to decide whether you want to create a custom MT system trained with it. This option may take longer time to complete the request. You must choose Yes or No to be able to continue with the TM file upload.

However, if neither the source language nor the target language is a Microsoft GA language, and if the XLIFF TM contains fewer than 10,000 TUs, DTS uses a general domain MT system after recycling is completed. This behavior occurs because of requirements that are set by Microsoft Translator Hub (MT Hub).

After you've finished uploading files, select **Submit** to start the translation process.

After you submit the request, a new request ID is created on the DTS dashboard. If you submitted the request with multiple target languages, you will see each target language status displayed n a separate line with the same request ID. Selecting a line on the dashboard will extend the dashboard window to the right to show the request summary information.

To see the request status, click a request ID link on the dashboard. The **Request status** tab shows the source files list you uploaded with the summary of the request information.

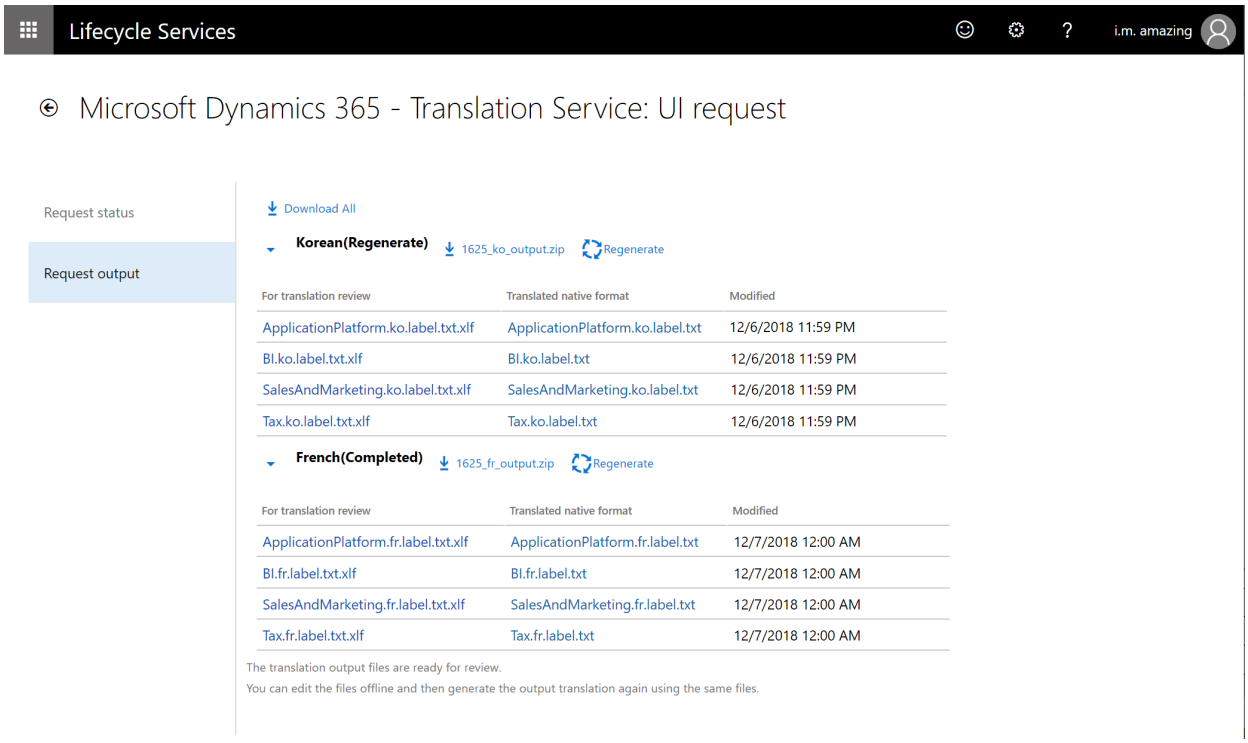


Note that the processing time depends on the number of requests that are in the DTS queue and the word count in the source files that you submit.

- UI translation requests that don't have an XLIFF TM can be completed in a few minutes, depending on the file size.
- If a UI translation request does have an XLIFF TM, the time that is required depends on the type of MT system:
  - Creation of a custom MT system requires two to three days.
  - If you're using a generic MT system, requests can be completed in a few minutes, depending on the file size.

After translation is completed

When processing of your translation request is completed, you receive an email notification from DTS. You can then view the result on the **Request output** tab of the request details page.



For UI translation requests, two types of output file are available after the translation process is completed.

- **For translation review** – Download the XLIFF file to review and, as required, edit the translations. The file shows the source and target languages side by side.
- **Translated native format** – Download this file if you don't intend to review or edit the translations. *Native format* means that the file is in the same format as the source file that you submitted.

Click an individual file link or the download links to download a single file, all files for one target language, or all files for all target languages in one zip for convenience.

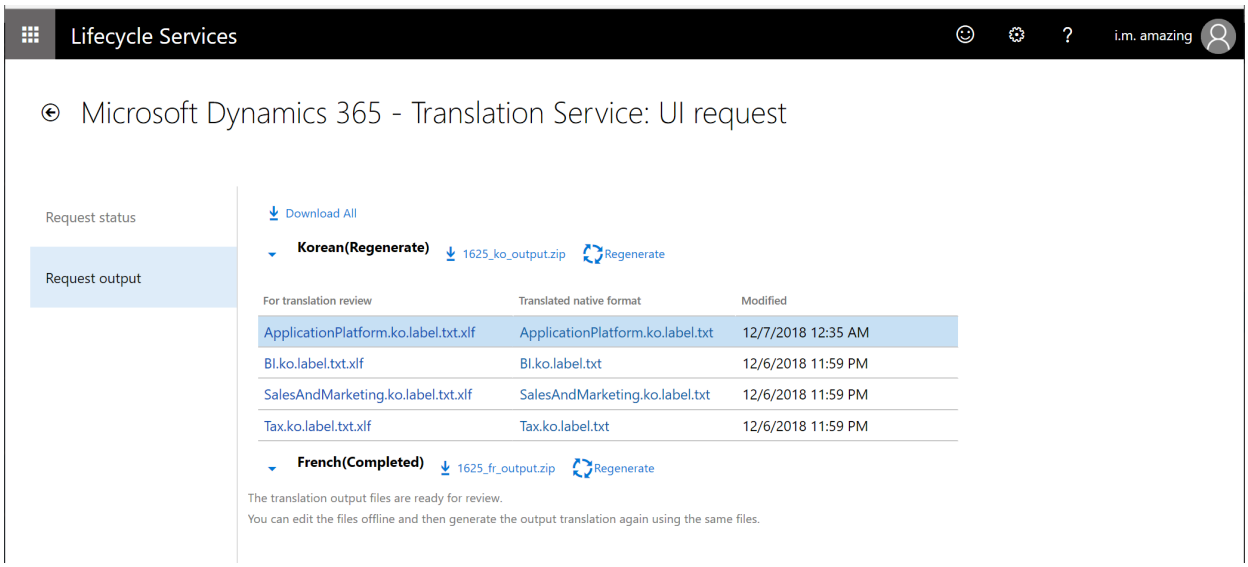
Review and edit the translations in the XLIFF file

We recommend that you review and edit the translations in the XLIFF file that DTS provides, to verify that the translation output meets your product's quality standards. For more information about how to edit the XLIFF file, see Translation memory files.

Regenerate output files

When you've finished reviewing and editing the translation files in XLIFF, you must regenerate the translated native format files next. You can then apply the latest translations (that is, your edited versions of the translations) to the UI files in the target language. You can regenerate any number of files from the output files set per target language.

1. Click the **Regenerate** icon next to the target language section. The **File upload** slider will open.
2. Zip the edited XLIFF files, and then click **Upload**. Don't change the XLIFF file name that DTS originally provided.
3. In the prompt, confirm the upload.
4. The **Request output** tab promptly refreshes the content. Expand the target language node you just regenerated to verify the **Modified** timestamp and then download the updated output files.



You can repeat the regeneration process as many times as you require.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

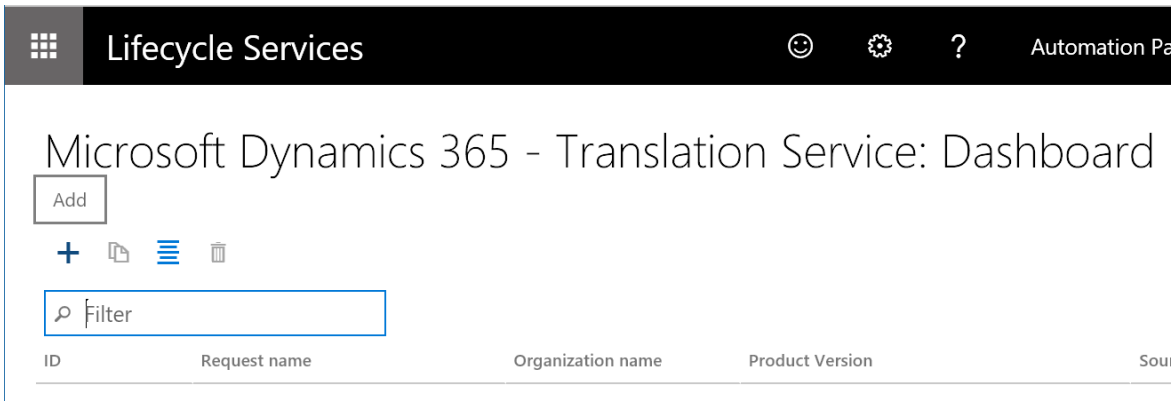
Translate documentation files

2/18/2021 • 6 minutes to read • Edit Online

This topic explains how to translate a documentation file for Microsoft Dynamics products and solutions.

Create a translation request

1. In Microsoft Dynamics Lifecycle Services (LCS), on the DTS dashboard, select **Add** to create a new translation request.

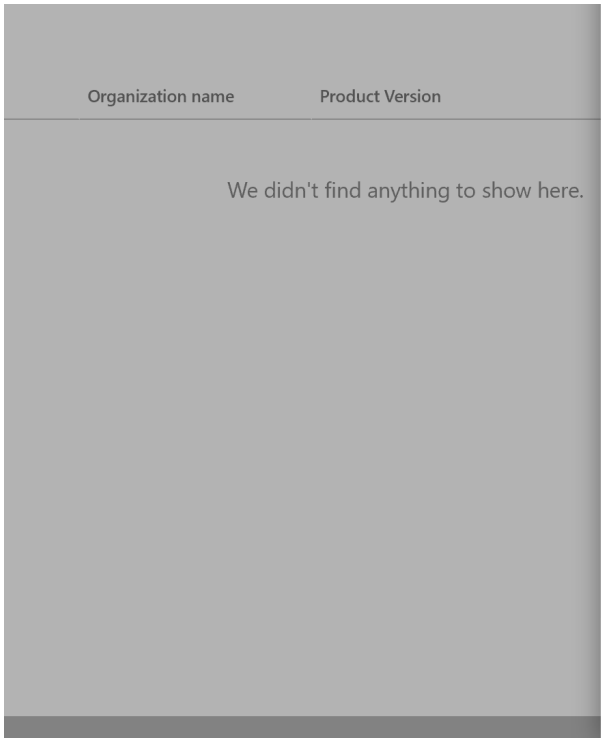


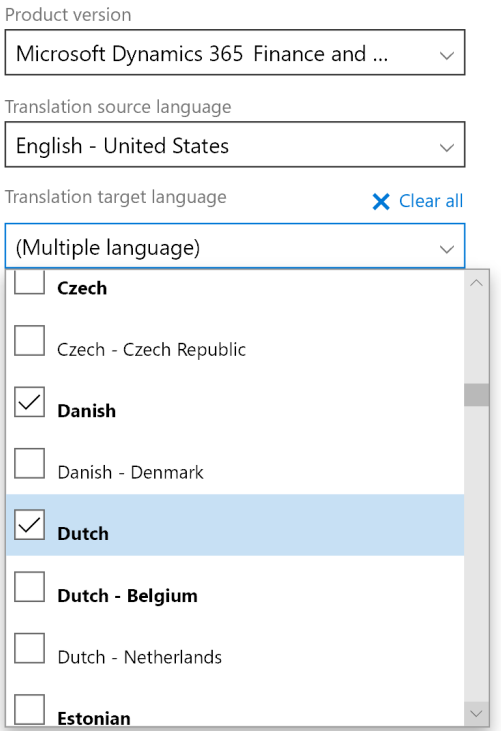
   You can open the DTS dashboard either from the LCS home page or from within a project. For more information, see Accessing DTS.

2. Enter the required information for the request.

FIELD	DESCRIPTION
Request name	Enter a name for the request.
File type	Select **Documentation**. This option is only available if you've turned on the **Dynamics 365 Translation Service - Documentation Translation Support** preview feature in LCS. For more information, see Accessing LCS preview features.
Product name	Select a product name. If you accessed DTS from within an LCS project, this field is automatically filled in and is read-only.
Product version	Select a product version. If you accessed DTS from within a LCS project, this field shows the default product version information from the project. However, you can select a different version.

FIELD	DESCRIPTION
Translation source language, Translation target language	Select the set of source and target languages to translate from and to. If your business needs multiple target languages translated for the same source language, you can select all target languages in one request. Make sure you actually select the target languages using the checkbox next to the language name. This saves your time to submit multiple translation requests individually and you can also track all the target languages translation status in one request. Language names that are shown in **bold** type are General Availability (GA) languages for Microsoft Dynamics products. Therefore, Microsoft-trained machine translation (MT) systems are available in those languages. In other words, the MT system is trained on the terminology for Microsoft Dynamics. For non-GA languages, the MT system uses the general domain training.





3. Select **Create**.

NOTE

To take advantage of the Microsoft-trained MT system for Microsoft Dynamics linguistic assets, you must select **English – United States** as either the source language or the target language. Here is an example.

TRANSLATION SOURCE LANGUAGE	TRANSLATION TARGET LANGUAGE	MT SYSTEM THAT IS USED
English – United States	Japanese	Microsoft-trained MT system
Japanese	English – United States	Microsoft-trained MT system
German	Japanese	Generic MT system, unless the user provides a translation memory (TM) that uses XML Localization Interchange File Format (XLIFF) and has more than 10,000 translation units (TUs)

Upload files

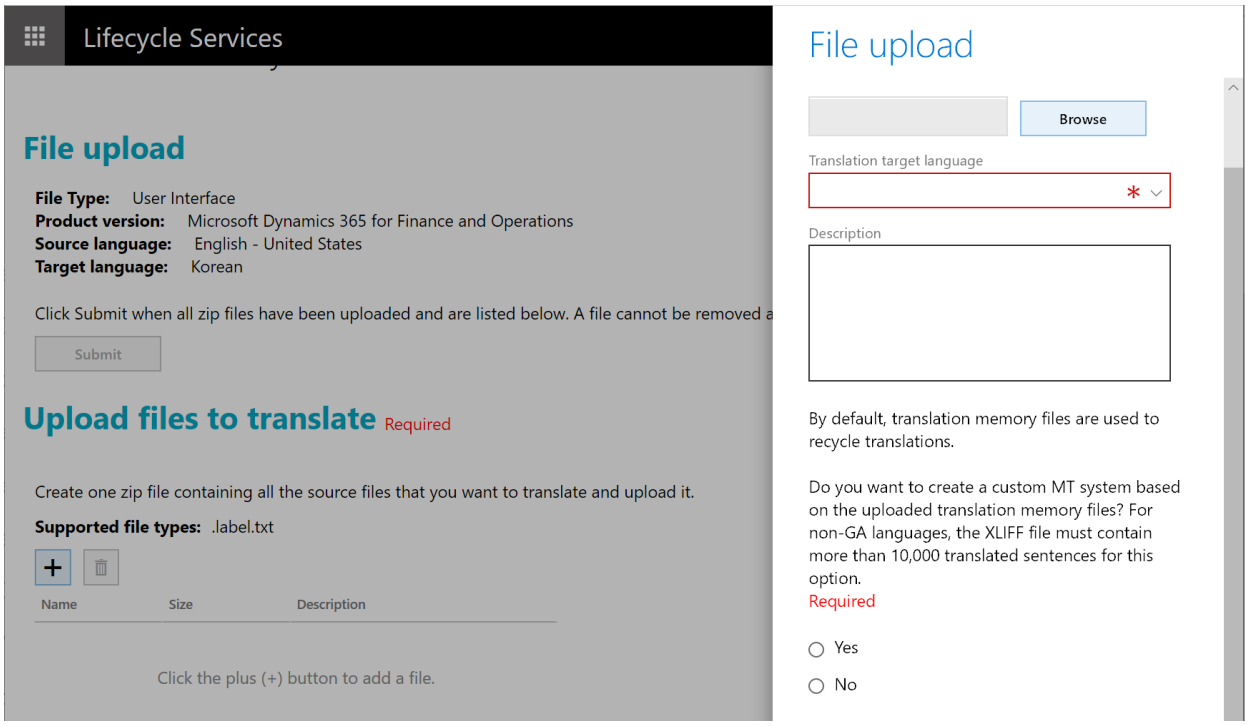
Select the plus sign (+) in each section to open the **File upload** page.

Upload files to translate (Required)

Currently, only files in Microsoft Word (.docx) format are accepted for translation. Create a zip file that includes all the .docx files in the source language that you want to translate from. You can upload only one zip file. Note that DTS doesn't change the source files that you upload. The source files are only used to create files in the corresponding target languages you requested.

Upload XLIFF or TMX translation memory files (Optional)

If you have a TM in Translation Memory eXchange (TMX) format from a previous DTS request, and/or if you have a XLIFF TM from UI file translation, you can attach those TMs so that they can be recycled in the new document that you're submitting. Create a zip file that includes all the TM files. You can upload only one zip file. If you created the translation request for multiple target languages, you must select which target language the TM file is for.



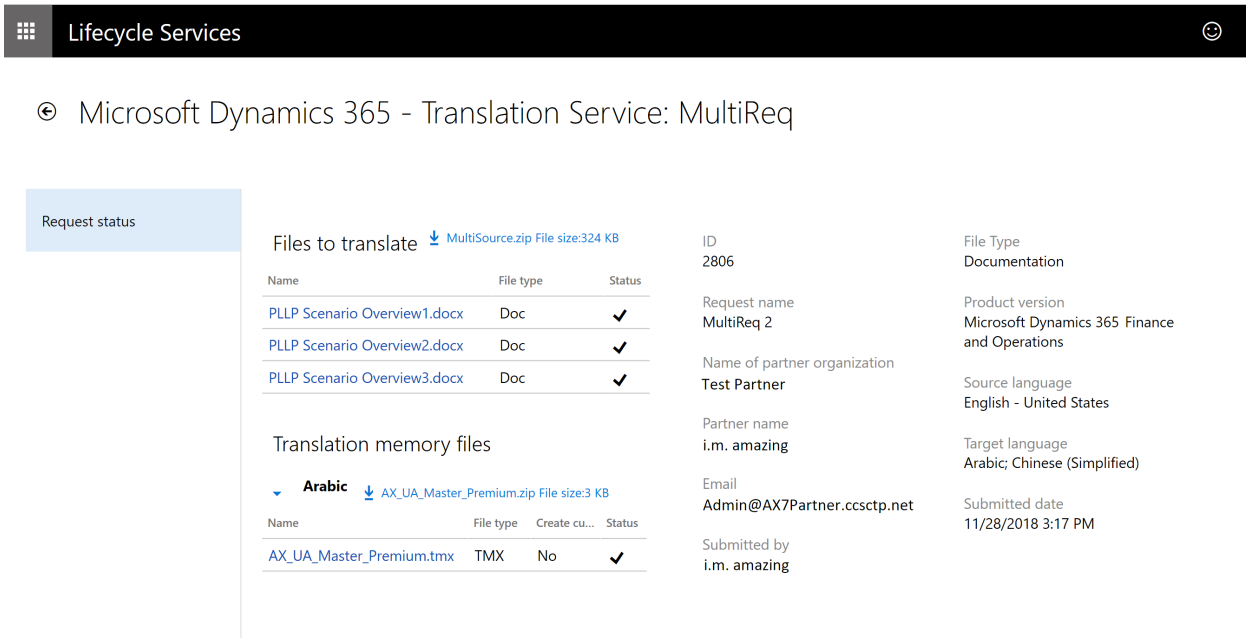
With the translation memory file you are providing, you have an option to decide whether you want to create a

custom MT system trained with it. This option may take longer time to complete the request. You must choose Yes or No to be able to continue with the TM file upload.

After you've finished uploading file, select **Submit** to start the translation process.

After you submit the request, a new request ID is created on the DTS dashboard. If you submitted the request with multiple target languages, you will see each target language status is displayed in a separate line with the same request ID. Selecting a line on the dashboard will extend the dashboard window to the right to show the request summary information.

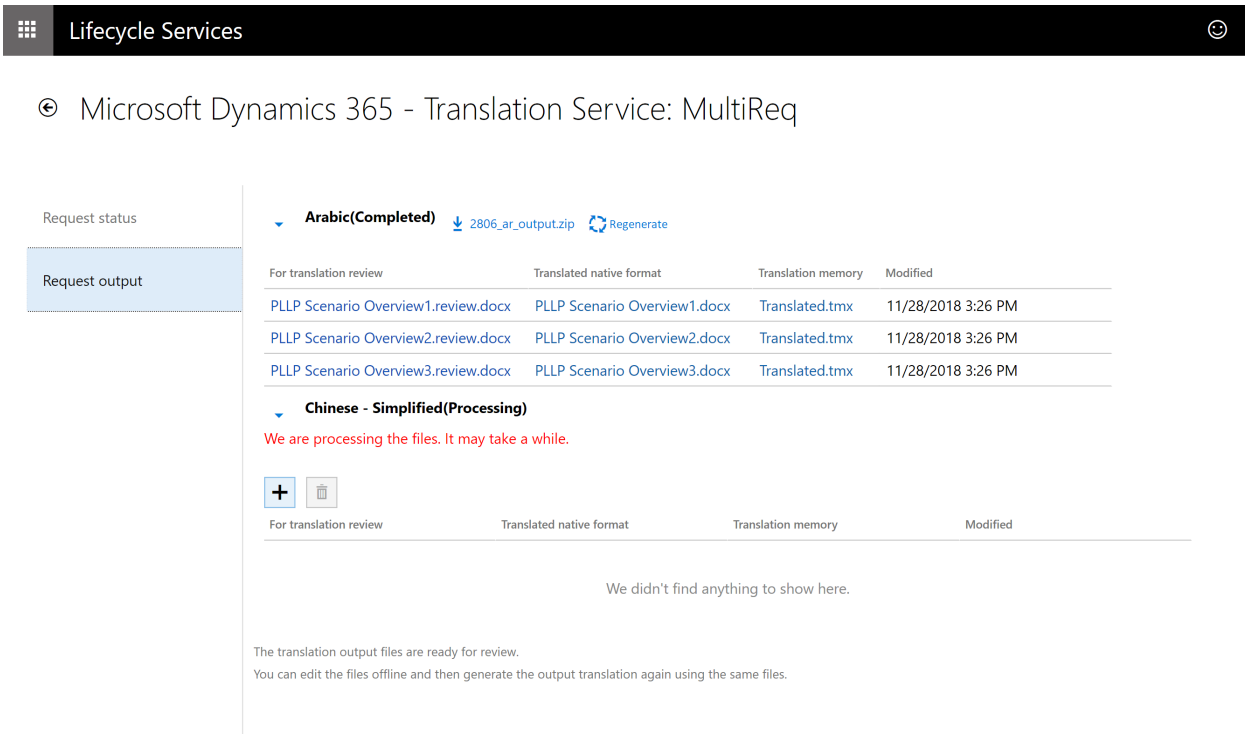
To see the request status, click a request ID link on the dashboard. The **Request status** tab shows the source files list you uploaded with the summary of the request information..



Note that the processing time depends on the number of requests that are in the DTS queue and the word count in the source files that you submit.

After translation is completed

When processing of your translation request is completed, you receive an email notification from DTS. You can then view the result on the **Request output** tab of the request details page. If your request was submitted for multiple target languages, there may be a difference between languages when each language process is done. Expand each language name to see the status.



For documentation translation requests, three types of output file are available after the translation process is completed:

- **For translation review** – Download this file to review and edit the translated document strings in a table view. The file shows the source and target languages segments side by side.
- **Translated native format** – Download this file if you don't intend to review or edit the translations but to pick up the translated file ready to use. This file has the dame formatting style (title, headings, tables, etc.) as the source .docx file that you submitted, and it's ready to be used.
- **Translation memory** – Download this file to recycle these translations the next time that you submit a translation request that uses a newer version of the source document.

Review and edit the translations

DTS provides the translation review file in .docx format. You can download the file from the **Request output** tab of the request details page and open it in Word. The file provides a convenient table view, as shown in the following illustration. Therefore, you can easily compare the text in the source and target languages side by side. After you've finished reviewing the file, you must save it and upload it back to DTS to generate the updated .docx file output in the original formatting style that you submitted.

Segment ID	Segment status	Source segment	Target segment
1	Draft (0%)	*<100/><117><110><103>*Pre Go-live checklist*</103><109></109></110><116>*Microsoft Dynamics 365 for Finance and Operations, Enterprise edition*</116></117>*	*<100/><117><110><103>*Avant mise en service liste de vérification*</103><109></109></110><116>*Microsoft Dynamics 365 pour Finance and Operations, Enterprise edition*</116></117>*
2	Draft (0%)	Customer:	Client :
3	Draft (0%)	Partner:	Partenaire :
4	Draft (0%)	LCS project name:	Nom du projet LCS :
5	Draft (0%)	LCS ID:	ID DE LETTRES DE CRÉDIT :
6	Draft (0%)	Go-live date:	Date de mise en service :
7	Draft (0%)	DD-MM-YYYY	AAAA-MM-JJ
8	Draft (0%)	Dynamics 365 for Finance and Operations, Enterprise edition customers should complete a Pre-Go-live review with the Microsoft FastTrack team before requesting their production environment.	Dynamics 365 pour Finance and Operations, clients Édition entreprise doit remplir une révision live donné avant avec l'équipe Microsoft FastTrack avant de demander son environnement de production.

When you edit the .docx review file, note of the following guidelines:

- Edit only the text in the **Target segment** column.
- Don't add or remove rows.
- Don't change the order of the rows or columns.
- Don't add or remove the red tags. Most red tags represent formatting and styles.
- If you must move the red tags, be careful that you don't switch a start tag (for example, **<116>**) and its end tag (**</116>**).

Regenerate output files

When you've finished reviewing and editing a .docx review file, you must regenerate the output file in the source document style. You can then apply the latest translations (that is, your edited versions of the translations) to the documentation files in the target language.

1. Click the **Regenerate** icon next to the target language section. It brings in the **File upload** slider.
2. Zip the edited .docx files, and then select **Upload**. Don't change the file names that DTS originally provided for the .docx review file.
3. You're prompted to confirm the upload action.
4. Once the regenerate is processed, the **Request output** tab refreshes the content. This process may take some time to complete.

You can repeat the regeneration process as many times as you require.

For more information about the Microsoft Dynamics 365 Translation Service (DTS), see Dynamics 365 Translation Service overview. For information about how to translate a user interface (UI) file, see Translate user interface files.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Translation memory files

2/18/2021 • 4 minutes to read • Edit Online

Microsoft Dynamics 365 Translation Service (DTS) uses a bilingual XML Localization Interchange File Format (XLIFF) file to store pairs of source languages and target languages. Because XLIFF is based on XML, you can open XLIFF files in any text editor. However, we recommend that you use XLIFF editors that are specifically designed to work with this format. For example, you can use the free Microsoft Multilingual Editor that is available in the Multilingual App Toolkit (MAT).

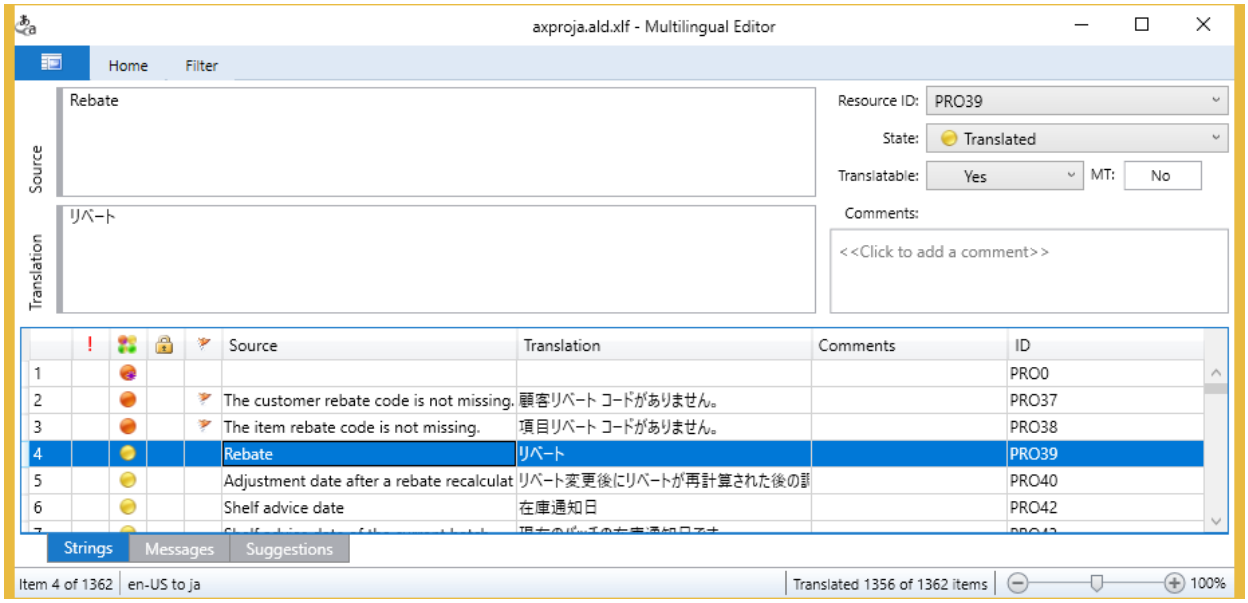
In DTS, you can obtain an XLIFF translation memory (TM) in two ways:

- **Run the Align tool** – When you have files that were previously translated, and you also have corresponding source files, you can use the Align tool to create an XLIFF TM. For more details, see the Creating a translation memory section later in this topic.
- **Complete a translation request** – When a DTS translation request is completed, it provides the XLIFF TMs as part of the request output. You can then use the files the next time that you submit a new translation request that includes the updated source files.

XLIFF files contain a series of translation units (TUs) that are extracted from the source files. The following illustration shows an example of a TU.

```
<trans-unit id="PRO39" translate="yes" xml:space="preserve">
    <source>Rebate</source>
    <target state="translated" state-qualifier="exact-match">リベート</target>
</trans-unit>
```

The following illustration shows the same TU (highlighted in blue) in the Multilingual Editor.



State

Each translation in the XLIFF file is associated with a state value. The state value that DTS assigns to each translation depends on the way that the string is translated. When an XLIFF TM is created by using the Align tool, all translations are marked as **Translated**, because the aligned TUs are produced from known good translations, such as a previous product version.

However, when the XLIFF files are generated through a translation request, two types of state values can be used:

- **Needs Review** – The string has been machine-translated.
- **Translated**, **Final**, or **Signed off** – The string has been recycled. The state value is inherited from the XLIFF TM.

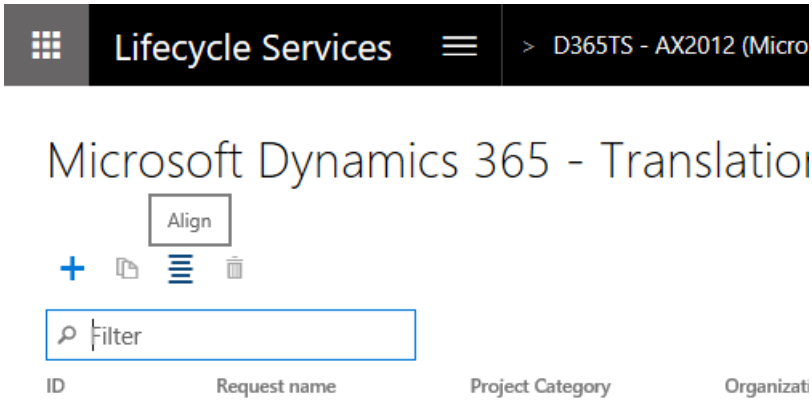
During the post-editing process, you can immediately identify the strings that are marked as **Needs Review**. After you've finished reviewing those strings, you should mark them as **Translated**, **Final**, or **Signed off**, so that they can be used for recycling. Translations that are marked as **Needs Review** aren't included for recycling.

Inherited state values for recycled strings are helpful, because you won't have to review the same string (that is, a string that has the same ID) again.

Creating a translation memory

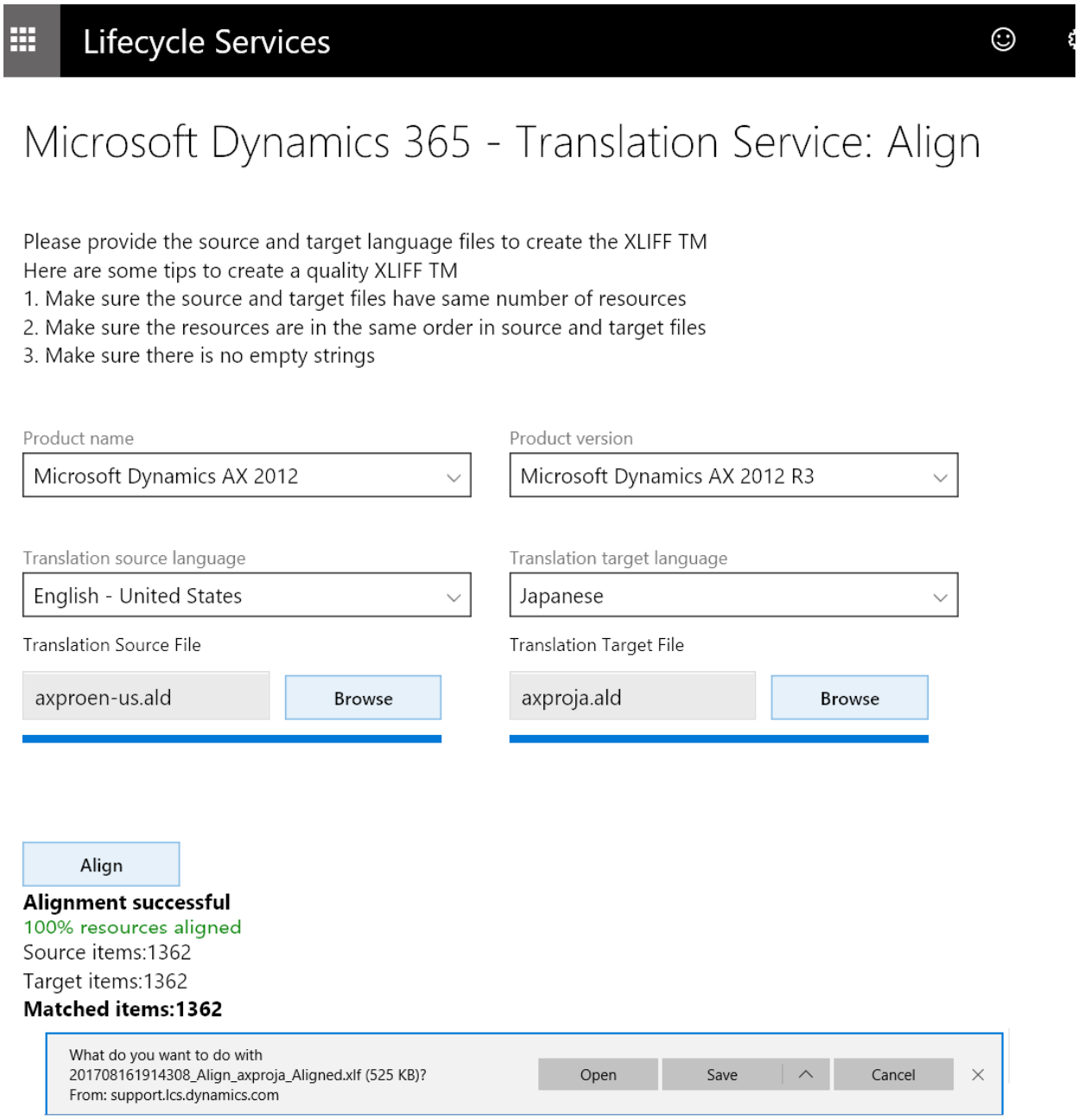
If you have files that were previously translated, you can recycle the translated files for a newer version of the source files by creating a TM that uses XLIFF.

1. On the DTS dashboard, select the **Align** button to start the Align tool.



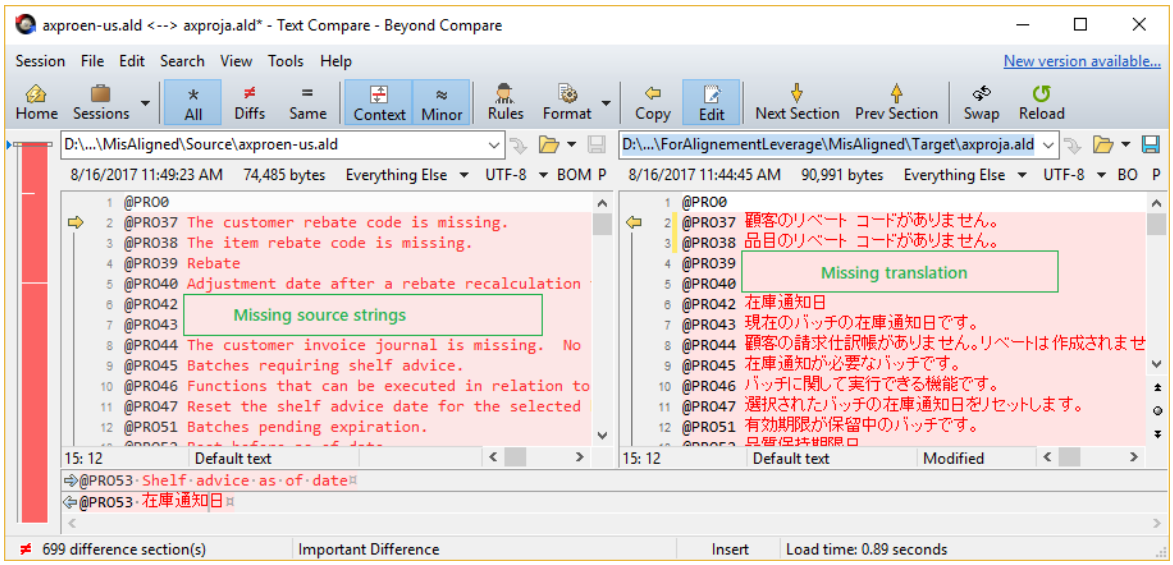
> **NOTE**
> - The Align tool currently supports only user interface (UI) files.
> - To start the Align tool, you might have to explicitly allow pop-up windows in your browser.

2. On the **Align** page, select the source language, the target language, and the files to align.
3. Select **Align** to complete the alignment. When the alignment is completed, a message summarizes the results.

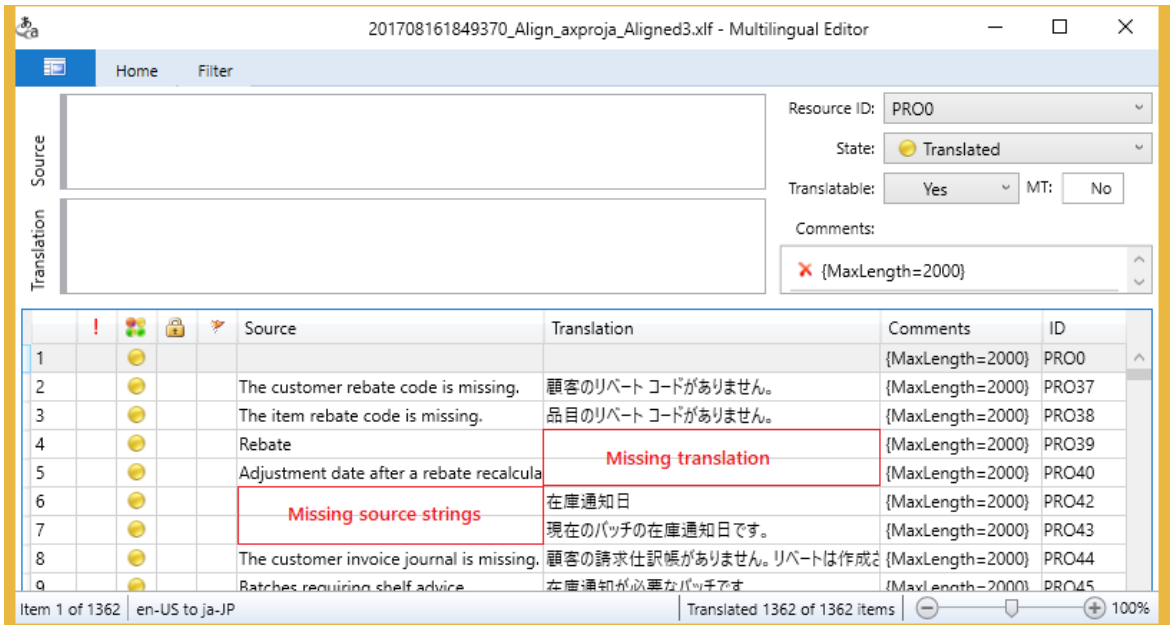


To create the best XLIFF TM, make sure that the following conditions are met:

- Both the source file and the target file have the same number of resources.
- The resources are in the same order in both the source file and the target file.
- There are no empty strings. The following illustration shows examples of empty strings in the source and the target.



Empty strings are inherited by the XLIFF TM. If a **Rebate** string in the source has an empty string in the target, it will likely be translated as an empty string if this XLIFF TM is used.



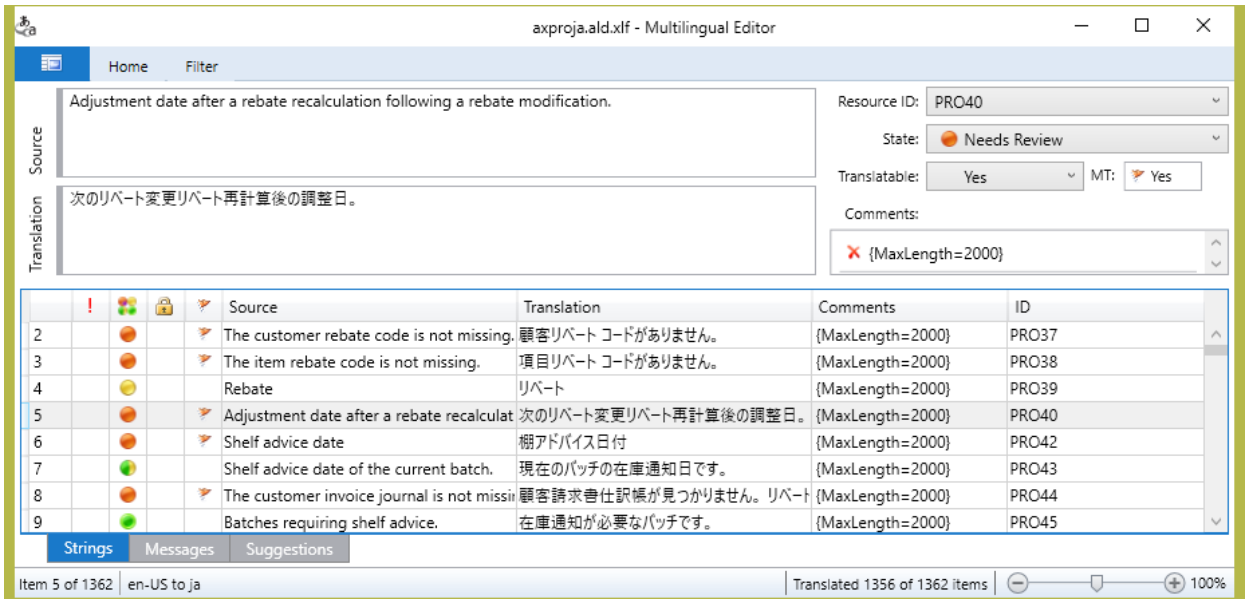
Although the Align tool can resolve some of these issues, it's easier if you prevent them before you see unexpected results in the output.

Review the aligned XLIFF file before you use it as a TM. TUs that have been reviewed should be marked as **Final** or **Signed off**, so that they aren't mistaken for unreviewed TUs.

Editing an XLIFF translation memory

We recommend that you use the free Multilingual Editor, or another XLIFF editor, to review and edit the translations in the XLIFF file that DTS provides. At a minimum, you should review the translations to verify that the translation output meets your product's quality standards.

When you open an XLIFF file in the Multilingual Editor, it resembles the following illustration.



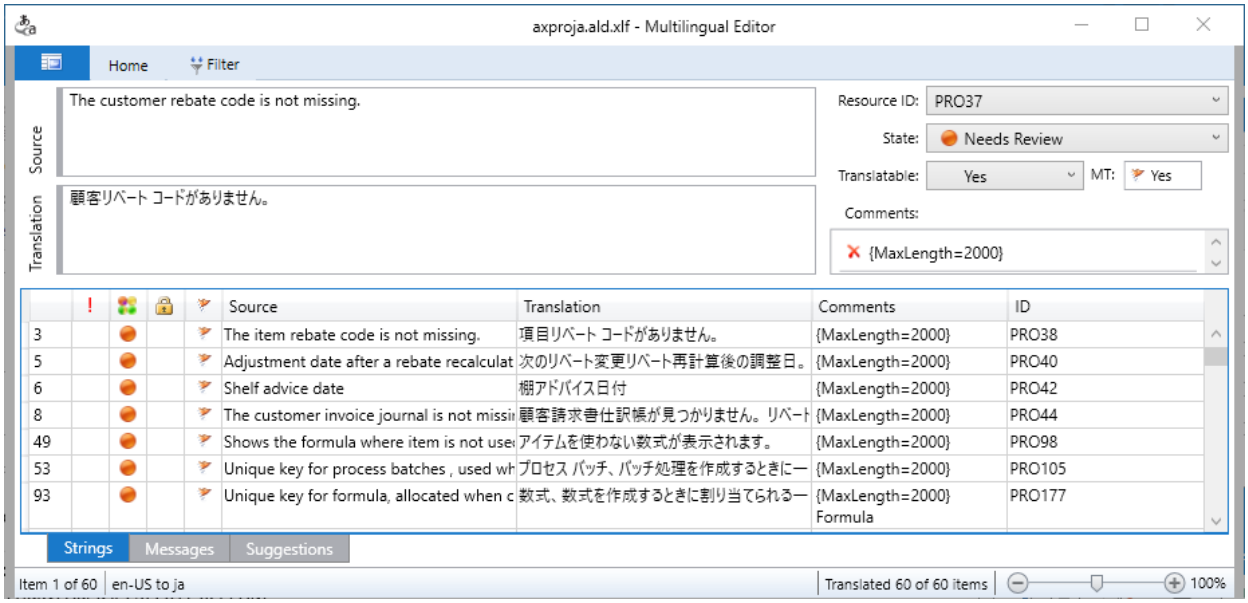
Notice that there is a circle near the beginning of each line. The color of the circle indicates the state of the translation. DTS automatically assigns these states, depending on where the string came from.

- **Red circle** – The string was machine-translated. DTS assigns the **Needs Review** state.

  > **NOTE**
  > The state value that is shown might differ slightly, depending on the XLIFF editor that you're using.

- **Yellow, green/yellow, or green circle** – The string was recycled. DTS inherited the state from the XLIFF TM that was used in the request.

To verify the translations, you can apply a filter to show only strings that are in the **Needs Review** state.



Strings that have been reviewed should be marked as **Translated**, **Final**, or **Signed off**, so that they can be used for recycling. Translations that are marked as **Needs Review** aren't included for recycling.

After you've finished editing the XLIFF TM, remember to have DTS regenerate the refreshed output file in the source format. For more information about how to regenerate the file, see Translate user interface files.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Microsoft Power Platform integration with Finance and Operations

2/18/2021 • 5 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

IMPORTANT

This functionality requires version 10.0.12 for Finance and Operations apps, while service update 189 is required for Dataverse. The release information for Dataverse is published on the latest version availability page.

Finance and Operations is a virtual data source in Dataverse, and enables full create, read, update, delete (CRUD) operations from Dataverse and Microsoft Power Platform. By definition, the data for virtual entities doesn't reside in Dataverse. Instead, it continues to reside in the application where it belongs. To enable CRUD operations on Finance and Operations entities from Dataverse, entities must be made available as virtual entities in Dataverse. The allows CRUD operations to be performed, from Dataverse and Microsoft Power Platform, on data that resides in Finance and Operations apps.

Prerequisite reading

To understand the architecture of virtual entities for Finance and Operations apps, you must understand how Dataverse and virtual entities work. Therefore, the following documentation is a prerequisite:

- What is Dataverse?
- Entity overview
- Entity relationships overview
- Create and edit virtual entities that contain data from an external data source
- What is Power Apps portals?
- Overview of creating apps in Power Apps

Virtual entities for Finance and Operations apps

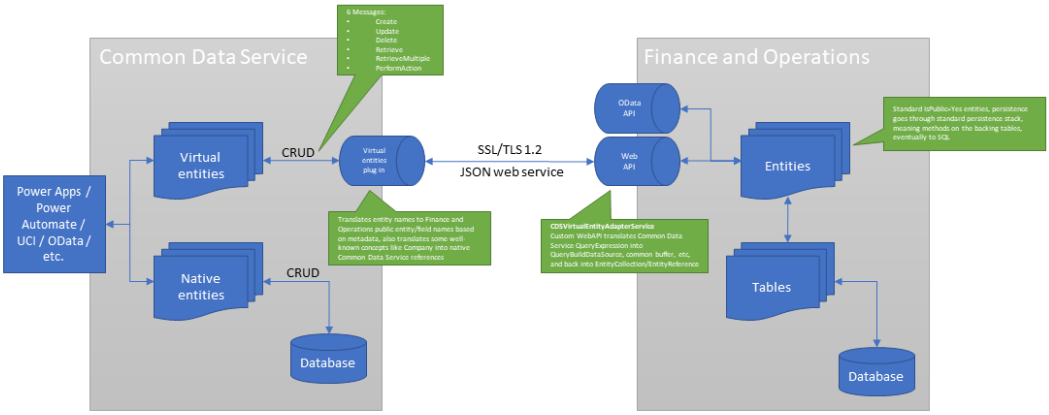
All Open Data Protocol (OData) entities in Finance and Operations are available as virtual entities in Dataverse, and therefore also in Power Platform. Makers can now build experiences in customer engagement apps with data directly from Finance and Operations with full CRUD capability and without copying to Dataverse. Power Apps Portals can be used to build external-facing websites that enable collaboration scenarios for business processes in Finance and Operations.

Virtual entities vs. dual-write

Virtual entities provide a mechanism to use Microsoft Power Platform with Finance and Operations without having to physically copy data to Dataverse. Use this guidance to determine if the requirements will need dual-write or data integrator or virtual entities. Virtual entities and dual-write/data integrator are complementary technologies, meaning that they can be used together if required. Each technology exists for specific scenarios, as explained in Integration strategies.

Architecture

Virtual entities are a Dataverse concept that is useful beyond Finance and Operations. The following illustration shows how the Finance and Operations provider for virtual entities is implemented. Six primary methods are implemented by the provider. The first five methods are the standard CRUD operations: **Create**, **Update**, **Delete**, **Retrieve**, and **RetrieveMultiple**. The last method, **PerformAction**, is used to call OData actions, as described later in this topic. Calls to the Finance and Operations Virtual Entity Data Provider (shown as "VE Plugin" in the illustration) will cause a Secure Sockets Layer (SSL)/Transport Layer Security (TLS) 1.2 secure web call to the CDSVirtualEntityService web API endpoint of Finance and Operations. This web service then converts the queries into calls to the associated physical entities in Finance and Operations, and invokes CRUD or OData operations on those entities. Because a Finance and Operations entity is directly invoked in all operations, any business logic on the entity or its backing tables is also invoked.



During calls, there are two points of translation from Dataverse to Finance and Operations apps. The first point of translation occurs in the VE Plugin, which translates concepts such as entity physical names into Finance and Operations entity names. It also converts some well-known concepts, such as Company references. The web service call still uses the EntityCollection, Entity, and QueryExpression objects to express the operations that are performed, by using the translated entity names and concepts from the VE Plugin. Finally, the CDSVirtualEntityAdapterService web API in Finance and Operations completes the translation from QueryExpression to QueryBuildDataSource and other internal Finance and Operations language constructs.

All calls between Dataverse and Finance and Operations as part of virtual entities are done as service-to-service (S2S) calls by using the Azure Active Directory (Azure AD) application that is specified in the configuration. The user of this application should have access *only* to the CDSVirtualEntityAdapterService web API and the Catalog entity, CDSVirtualEntityListEntity. These privileges are included in the out-of-box security role that is named CDSVirtualEntityApplication. During the S2S calls, Dataverse provides the identity of the user in Dataverse who is invoking the action. The CDSVirtualEntityAdapterService web API looks up the associated user in Finance and Operations and runs the query in the context of that user. Therefore, the S2S call doesn't have to have explicit access to all the Finance and Operations entities. Instead, it can rely on the privileges of the user who is invoking

the action to determine data access.

> **NOTE**
>
> We always recommend that you have both Finance and Operations and Dataverse co-located in the same Azure region to ensure optimal latency in virtual entity calls. When co-located, the virtual entity overhead is expected to be less than 30ms per call.

Power Apps Portal can also access virtual entities. Because Power Apps Portal authorization is based on contact records, a mapping between contact records and Finance and Operations users is maintained in the msdyn_externalportalusermapping table in Dataverse. This table should be editable only by highly privileged users in Dataverse, who have the rights to control the security access of portal users to Finance and Operations virtual entities. Any Finance and Operations user who is set up for Power Apps Portal access must have the CDSVirtualEntityAuthorizedPortalUser security role assigned, and can't have the System administrator or Security administrator role assigned. Regardless of the Power Apps Portal security setting that is applied to virtual entities, the resulting query to Finance and Operations apps is always run as the associated Finance and Operations user, and is subject to that user's entity and row security settings. Anonymous portal access is also supported. For information about this type of access and how it can be done, see Power Apps Portal reference.

Virtual entities for Core HR

Core HR entities can also be virtualized like Finance and Operations entities. For more information, see Core HR virtual entities.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Add-ins overview

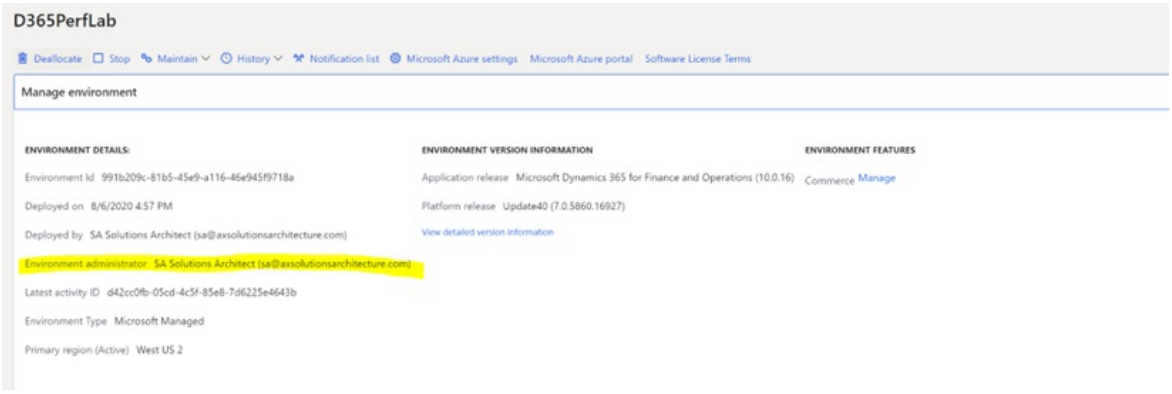
2/18/2021 • 2 minutes to read • Edit Online

Add-ins provide a way to extend the functionality of Finance and Operations apps. The following topics provide some examples of add-ins:

- Planning Optimization overview
- Inventory Visibility Add-in
- Configure export to Azure Data Lake
- IoT Intelligence home page

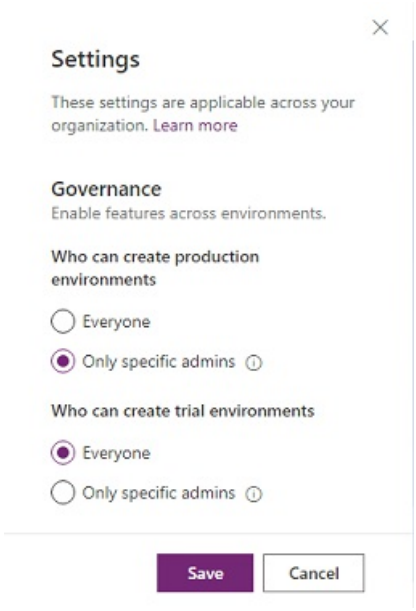
Prerequisites for setting up add-ins

- Make sure that at least 1 gigabyte (GB) of Microsoft Dataverse space is available. Otherwise, setup will fail. You can view your capacity in the Power Platform admin center.
- Identify your Finance and Operations environment administrator. You can find that information in the **Environment details** section.



- Validate your Power Platform environment governance policy. To validate, you must be a **Global administrator** or **Power Platform administrator**.

  1. Sign in to the Power Platform admin center.
  2. Select the gear icon in the upper-right corner of the Power Platform site.



- For organizations that do not allow **Everyone** to create Power Platform production environments, the Finance and Operations environment administrator account must be added as the admin.

  The Finance and Operations environment administrator must be added to one of the following roles. You will need a Global Administrator to perform this action.

  - Global admins
  - Dynamics 365 admins
  - Power Platform admins

  For more information, see Use service admin roles to manage your tenant.

Set up add-ins

> **NOTE**
> If the add-ins that you're planning to install don't require "dual-write linking," you don't have to complete this procedure.

1. After the Finance and Operations environment has been deployed through LCS, open the **Environment details** page in LCS.
2. In the **Power Platform integration** section, select **Setup**.
3. In the **Power platform environment setup** dialog box, select the check box, and then select **Setup** at the bottom of the dialog box.

   > **NOTE**
   > The Dataverse environment is set up so that it includes dual-write functionality. Typically, this setup consumes about 1 GB of Dataverse space.

4. When you receive a message that states that the Microsoft Power Platform environment is being provisioned, select **OK**.

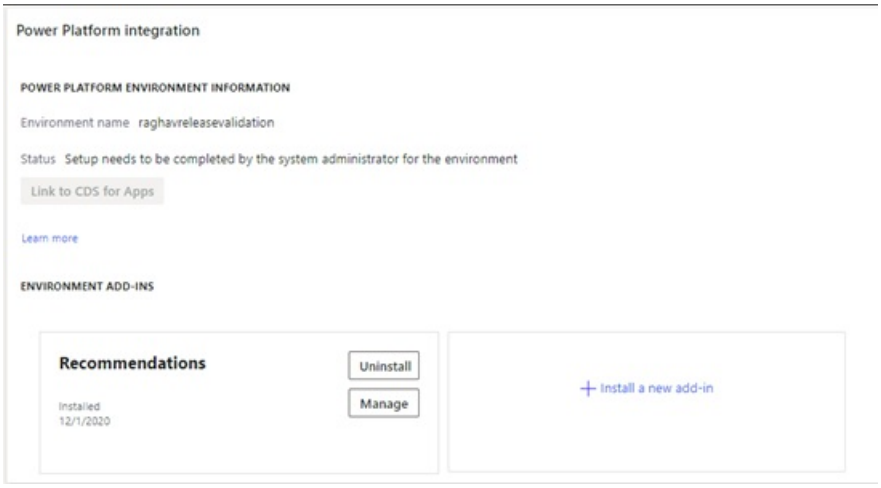
   The **Power Platform integration** section of the **Environment details** page now shows a message that states that the Microsoft Power Platform environment is being provisioned.

5. After a few minutes, refresh the **Environment details** page.

6. In the **Power Platform integration** section, notice that the value of the **Status** field is **Environment setup is in progress**.

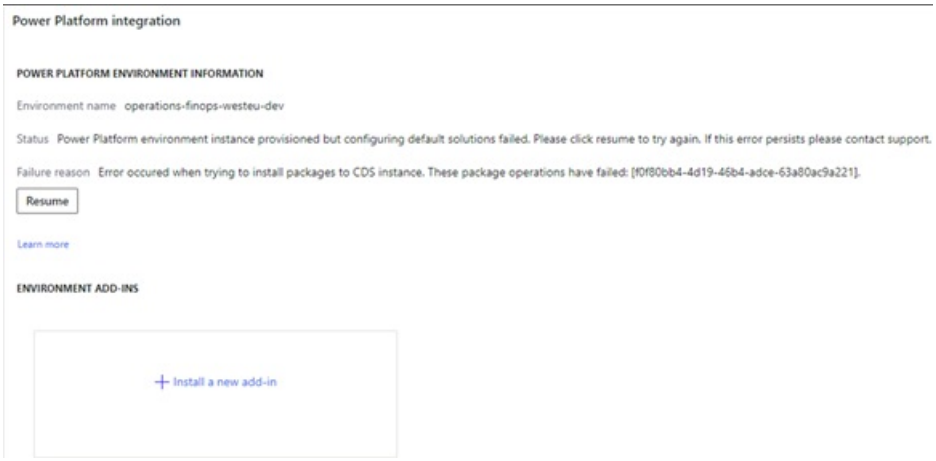
   Typically, the setup takes between 60 and 90 minutes.

   After the Dataverse environment is provisioned, the **Install a new add-in** button becomes available in the **Power Platform integration** section.



Troubleshoot the setup

- Because the setup process tries to set up dual-write functionality, the dual-write setup might sometimes fail after the Dataverse environment is provisioned, as shown in the following illustration. In these cases, the **Install a new add-in** button is still available, so that you can unblock the installation of add-ins. If you want to continue to set up dual-write functionality, select **Resume**.



- Provisioning of the Dataverse environment might sometimes fail because of capacity and licensing issues. In these cases, sign in to the Power Platform admin center, and fix the issues. Then, in the **Power Platform integration** section of the **Environment details** page in LCS, select **Resume**.

Entity modeling

2/18/2021 • 17 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

IMPORTANT

This functionality requires version 10.0.12 for Finance and Operations apps, while service update 189 is required for Dataverse. The release information for Dataverse is published on the latest version availability page.

> The public entity name that is exposed in Dataverse metadata for the Finance and Operations virtual entity uses the physical name of the Finance and Operations entity. This could be different from the public name of the entity as exposed by the OData metadata in Finance and Operations apps.

Building an app requires capabilities to perform relational modeling between entities that are being used in the app. In the context of virtual entities, there will be scenarios where virtual entities and native entities in Dataverse must work together to enable the desired user experience. This topic explains concepts of relational modeling that can be implemented using virtual entities for Finance and Operations.

Generating virtual entities

By default, virtual entities for Finance and Operations apps don't exist in Dataverse. A user must query the catalog entity to view the entities that are available in the linked instance of Finance and Operations. From the catalog, the user can select one or more entities, and then request that Dataverse generate the virtual entities. This procedure is explained in later sections.

Entity fields

When a virtual entity is generated for a Finance and Operations entity, the system tries to create each field in the Finance and Operations entity in the corresponding virtual entity in Dataverse. In an ideal case, the total number of fields will be the same in both entities, unless there is a mismatch in supported data types between Finance and Operations and Dataverse. For data types that are supported, the field properties in Dataverse are set based on the properties in Finance and Operations.

This rest of this section describes supported and unsupported data types. For more information about fields in Dataverse, see Fields overview.

DATA TYPE IN FINANCE AND OPERATIONS	MODELED DATA TYPE IN DATAVERSE

DATA TYPE IN FINANCE AND OPERATIONS	MODELED DATA TYPE IN DATAVERSE
Real	Decimal For information about the possible mismatch, see the next table.
Long	Decimal, where the precision equals 0 (zero)
Int	Integer
String (non-memo), String (memo)	String – single line of text, String – multiple lines of text
UtcDateTime	DateTime (DateTimeFormat.DateAndTime, DateTimeBehavior.TimeZoneIndependent) An empty date (January 1, 1900) in Finance and Operations is surfaced as a null value in Dataverse.
Date	DateTime - (DateTimeFormat.DateOnly, DateTimeBehavior.TimeZoneIndependent) An empty date (January 1, 1900) in Finance and Operations is surfaced as an empty value in Dataverse.
Enum	Picklist Finance and Operations enumerations (enums) are generated as global OptionSets in Dataverse. Matching between the systems is done by using the **External Name** property of values. Enum integer values in Dataverse aren't guaranteed to be stable between the systems. Therefore, you should not rely on them, especially in the case of extensible enums in Finance and Operations, because these enums don't have a stable ID either. OptionSet metadata is updated when an entity that uses the OptionSet is updated.

Fields of the *real* and *long* data types in Finance and Operations are modeled as the *decimal* data type in Dataverse. Because of the mismatch in precision and scale between the two data types, the following behavior must be considered.

USE CASE	RESULTING BEHAVIOR
Dataverse has higher precision.	This use case should never occur unless the metadata is out of sync.
Finance and Operations has higher precision.	During a read operation, the value is rounded to the closest precision value in Dataverse. If the value is edited in Dataverse, it's rounded to the closest precision value in Finance and Operations. During a write operation, the value that is specified in Dataverse is written, because Finance and Operations supports higher precision.
Dataverse has higher scale.	Not applicable

USE CASE	RESULTING BEHAVIOR
Finance and Operations has higher scale.	Dataverse shows the Finance and Operations value, even if it exceeds 100 billion. However, there will be a loss of precision. For example, 987,654,100,000,000,000 is shown in Dataverse as "987,654,099,999,999,900". If the value of this field is edited in Dataverse, Dataverse validation throws an error that the value exceeds the maximum value before that value is sent to Finance and Operations.

The following data types in Finance and Operations aren't supported in Dataverse. Fields of these data types in Finance and Operations entities won't be made available in the corresponding virtual entities in Dataverse. If fields of these data types are used as parameters in Open Data Protocol (OData) actions, those actions won't be available for use in the corresponding virtual entities. For more information about OData actions, see the OData actions section later in this topic.

- AnyType
- BLOB
- Class
- Container
- Guid
- Record
- Time
- UserType
- VarArg
- Void (Void return types on OData actions are supported.)

Data types that are supported in Dataverse but not in Finance and Operations aren't supported in virtual entities for Finance and Operations.

Entity key/primary key

In Finance and Operations, entities can have one or more fields of various data types as the entity key. An entity key uniquely identifies a record in a Finance and Operations entity. Additionally, a record in an entity can be uniquely identified by a record ID primary key of the Int64 type.

In Dataverse, the primary key is always a globally unique identifier (GUID). The GUID-based primary key enables a record in an entity in Dataverse to be uniquely identified.

To bridge the implementation gap between Finance and Operations and Dataverse, the primary key of a virtual entity for Finance and Operations is a GUID (to comply with Dataverse). This GUID consists of the data entity ID in the first 4 bytes, and the record ID of the root data source in the entity as the last 8 bytes. This design satisfies Dataverse's requirement that a GUID be used as the entity key. It also enables the table ID and record ID to be used to uniquely identify the entity record in Finance and Operations.

When using entities in Finance and Operations, you need to ensure that the root data source will always have a unique RecID. If this design is violated, duplicate GUID's will show up in Dataverse for the corresponding virtual entity. Aggregate views are not supported via virtual entities for the same reason because these views may not have unique RecIDs.

Primary field

In Dataverse, each entity must have a primary field. This field must be a single field of the string type. The primary field is used in Dataverse in the following scenarios:

- The default views that are created for an entity include the primary field.
- The quick view form for an entity includes the primary field.
- A lookup to another entity is added to a page and shows the data from the primary field.

Based on this use of the primary field in Dataverse, the primary field for a virtual entity for Finance and Operations is designed to use the entity key of the corresponding entity in Finance and Operations.

Because the primary field in Dataverse is expected to have only one field of the string type, whereas the entity key in Finance and Operations can have multiple fields of various data types, the entity key fields are converted to strings. The strings are concatenated and separated by a pipe (|), to a maximum length of 255 characters. Any value that exceeds 255 is truncated. This virtual entity field that represents the primary field is named **mserp_primaryfield**.

Relations

> **IMPORTANT**
> A write transaction that spans a virtual entity and a native entity is not supported. We do not recommend using this form of transaction, as there is no way to ensure consistency.

Relations in Finance and Operations entities are modeled as one-to-many (1:n) or many-to-one (n:1) relations. These relations are modeled as relationships in the virtual entity in Dataverse. Note that many-to-many (n:n) relations aren't supported in Finance and Operations.

For example, in Finance and Operations, if Entity A has a foreign key to Entity B, this relation will be modeled as an n:1 relationship in virtual entity Entity A in Dataverse. The schema name of this relationship in Dataverse uses the naming convention **mserp_FK_<source entity name>_<relation name>**. This naming convention has a maximum string length of 92 characters. Any relation where the schema name will produce a name that exceeds 92 characters won't be generated in the virtual entity in Dataverse.

The external name of this relationship uses the naming convention **FK_<relation name>**. The external name is used to determine the relation in Finance and Operations when the query that is sent to Finance and Operations is built.

When a relationship is generated for a virtual entity in Dataverse, a new field of the lookup type is also added to the source entity. In the preceding example, when the relationship is created, a new lookup field that uses the naming convention **mserp_fk_<target_entity>_id** is added to source entity Entity A. Because there can be several relations in an entity in Finance and Operations, the same number of lookup fields (one per related entity) will be created in the source virtual entity. When this lookup field is added to a page or a view, it will show the primary field value from the related entity.

A relationship in the virtual entity in Dataverse will be generated only if the related entity in the relation already exists as a virtual entity in Dataverse. In the preceding example, if Entity B doesn't exist as a virtual entity in Dataverse, the relation to Entity B won't be created in Entity A when Entity A is generated as a virtual entity. This relation will be added to Entity A only when Entity B is generated as a virtual entity. Therefore, when a virtual entity is generated for Finance and Operations, validations are done to ensure that only relationships that can be complete and functional are generated in the virtual entity that is being generated.

In summary, a relationship to another Finance and Operations virtual entity might not exist in the virtual entity for either of the following reasons:

- The Finance and Operations entity that is participating in the relationship doesn't exist as a virtual entity.
- The length of the name of the relationship exceeds 92 characters.

Note that if an error is encountered when any part of a Finance and Operations virtual entity is generated in

Dataverse, the virtual entity won't be created at all. If relationships don't exist for either of the preceding reasons, the situation isn't considered an error.

Native entity–to–native entity relationships

Native entity–to–native entity relationships are the standard Dataverse functionality, where relationships are resolved by using the GUID of the related entity. (This GUID is the entity key.) The GUID identifies the unique entity record in the related entity.

Virtual table–to–virtual table relationships

The relationships between two Finance and Operations virtual entities are driven by the relation metadata in the Finance and Operations entities. As was explained earlier, these relations are generated as relationships in Dataverse when the virtual entity is generated. As in the behavior for native entities in Dataverse, these relationships use the GUID to identify the unique record of the entity in Finance and Operations. Semantically, the GUID on the Finance and Operations virtual entity behaves like the GUID on the native Dataverse table. For information about the implementation of the GUID in Finance and Operations virtual entities, see the Entity key/primary key section earlier in this topic.

In the preceding example, the GUID of the related entity is the entity key of Entity B and will be used to build queries to identify a record in Finance and Operation. The relation that Entity A has to Entity B will be used.

Therefore, in effect, the entity name is the only information that is used in a relation that comes from Finance and Operations. The entity name gives access to the primary field in the related entity, so that it can be shown in the lookup. It also gives access to the GUID of the related entity, so that it can be used in other queries, as was explained earlier. The actual field that the relation is built on in the Finance and Operations entity isn't used at all.

Virtual table–to–native table relationship

As was explained earlier, the GUID is the only information that is used to uniquely identify a record in a native Dataverse table (including in native entity–to–native entity relationships) or in a Finance and Operations virtual entity (including in virtual entity–to–virtual entity relationships). However, consider an example where you want to show sales orders from Finance and Operations for Account A in Dataverse. The query that is sent to Finance and Operations for this relationship will have a WHERE clause on the GUID of the entity key of the native accounts entity in Dataverse, because the sales orders must be filtered for a specific account in Dataverse. However, because Finance and Operations doesn't have any information about the GUID of the entity in Dataverse, the query won't return any sales orders. The query will be successful only if the WHERE clause has conditions that are based on the fields that Finance and Operations understands.

Therefore, how can the GUID of the accounts entity in Dataverse be replaced with fields that are in Finance and Operations, in such a way that the query that is sent to Finance and Operations will return the correct list of sales orders?

To solve this issue and enable a rich set of scenarios that allows for virtual entity–to–native entity relationships, relationships can be added to this type of entity. The relation will appear as a relationship when the virtual entity is synced.

In the above example, the relationship between the SalesOrderHeader virtual entity and the Account native entity should be based on the Account Number and Company fields. By default, the native account entity in Dataverse does not have a company field. For this example, we will add a company lookup field named new_testcompany to the native Account entity.

Next, we add a new key named new_accountcompanyidx, which specifies that (accountnumber, new_testcompany) together represent a unique row in the account entity in Dataverse.

The next step is to define this relationship in X++. The following example shows sample X++ code. The names of the fields, index, and mapping information should match the names of the fields and indexes created in Dataverse. In this example, a relationship named “synthaccount” will be created between the virtual SalesorderHeader entity and the native account entity in Dataverse. The mapped fields make up the

new_accountcompanyidx index. The display name for the relationship will be @SYS11307. Note the backslash at the start of the display name. This ensures that the label defines the relationship, so that it is appropriately translated.

The field mapping indicates which field on the virtual entity maps to the field on the native entity. In the field mapping, the key is the virtual entity field, and the value is the native entity field.

```
[CDSVirtualEntitySyntheticRelationshipAttribute('synthaccount', 'account', 'accountcompanyidx',
'\@SYS11307')]
    public static Map syntheticAccountRelationship()
    {
        Map fieldMapping = new Map(Types::String, Types::String);

        // Assumes the Dataverse account entity has a key on [msdyn_accountnumber, msdyn_companyid]
        // Also assumes that the Dataverse cdm_Company entity has a key on [msdyn_companycode]
        fieldMapping.insert(fieldStr(CDSVirtualEntityTestEntity, StringField), 'msdyn_accountnumber');
        fieldMapping.insert(fieldStr(CDSVirtualEntityTestEntity, DataAreaId), 'msdyn_companyid');

        return fieldMapping;
    }
```

The next step is to generate or refresh the virtual entity to get the new relationship. Note that relationships between a virtual entity and a native entity cannot be updated in Dataverse once it is created. The only way to make an update is to physically remove the relationship, refresh the entity, and then physically re-add the relationship in order to resolve the issue.

This relationship looks like a typical GUID-based relationship, but has extra metadata to translate query filters on the relationship into restrictions on the backing fields. The query that is now generated will have a WHERE clause that is based on the fields that Finance and Operations apps recognize. That query will then return the filtered list of sales orders, as expected.

Native entity–to–virtual entity relationships

Native entity–to–virtual entity relationships works much like native entity–to–native entity relationships. Users associate native records with virtual records in Finance and Operations, and the GUID of the virtual entity is saved on the native entity record. As was explained earlier, the entities that participate in a relationship will have the GUID field of the related entity on them. Therefore, when a quotation in Dataverse is associated with a customer in a Finance and Operations virtual entity, the GUID of the customer virtual entity will be saved in the quotation entity. This behavior enables records to be retrieved as expected, by using standard Dataverse functionality.

Enums

Enums in Finance and Operations are modeled as OptionSets in Dataverse. When a virtual entity for Finance and Operations is generated, the required enums are generated as OptionSets. If an OptionSet already exists, it's used instead.

Company

An entity in Finance and Operations can be bound to a company, or it can be global. The virtual entity for a Finance and Operations entity that is bound to a company will have a relationship to the cdm_company entity in Dataverse. The cdm_company entity is a native entity in Dataverse and is part of the Dynamics365Company solution. As always, when a relationship is created, a lookup field is also created in the virtual entity for the related entity (cdm_company in this case). This lookup field is named **Company**, and it must be used to provide an optimal user experience where users can select a value in a list or go to the details of the related record. A field that is named **Company Code** is also added in the virtual entity. The value is a four-character string. This field must be used in programming.

Attachments

Attachments in Finance and Operations entities are supported on a per-entity basis. For example, an invoice header entity will implement an invoice-related attachments entity to enable attachments via entities.

Entities of this type will have relations with the corresponding attachments entity in Finance and Operations. Therefore, they will follow the same pattern as the other relations that were discussed earlier. In other words, Finance and Operations entities that have implemented attachments functionality will also make attachments available by using virtual entities. Finance and Operations entities that don't support attachments also won't support attachments when they are virtualized in Dataverse.

Note that Finance and Operations virtual entities support only the reading of attachments. They don't currently support the creation, update, or deletion of attachments by using virtual entities.

OData actions

OData actions in the Finance and Operations entities are made available as custom actions in Dataverse. For more information about custom actions and what they enable in Dataverse, see Custom actions.

Input and output parameters of the following types are supported. If an input or output parameter is of a different type, the OData action doesn't appear as the SDK message in Dataverse.

- Integer
- String
- Guid
- Boolean
- Date/Datetime

Here are some examples of OData actions that are supported in Finance and Operations entities, but that aren't supported in the corresponding virtual entities in Dataverse:

- RetailStoreTenderTypeTable.queryDistinctTenderTypeIdAndName (a collection of RetailStoreTenderTypeTable entity)
- DocumentRoutingClientApp.syncPrinters (DocumentRoutingClientApp entity)
- DocumentRoutingClientApp.updateJobStatus (DocumentRoutingJobStatus enum)
- DimensionCombination.getCombinationDisplayValue (LedgerJournalACType enum)

Labels and localization

Labels that are defined on metadata, such as entity names and field names in Finance and Operations, are retrieved when virtual entities are generated in Dataverse. The labels are retrieved by passing the list of language locales that are installed in Dataverse. Finance and Operations returns each label as a list of locale/value sets that are then used to construct a label instance in Dataverse. Only the language packs that exist at the time of entity generation or update are included. Additionally, only labels that Finance and Operations has provided a translation for are included. Any missing translations revert to the label ID, such as **@SYS:DataEntity**. After a new language pack is installed in Dataverse, existing entities must be updated to pick up the new label information, if labels in that language exist in Finance and Operations.

Any runtime labels are returned in the language of the current user context. In other words, they are returned in the language that is specified on that user's UserInfo record in Finance and Operations. This behavior also applies to error messages.

Error handling

Finance and Operations create, read, update, and delete (CRUD) business logic on entities and backing tables is

run when it's called through the virtual entity in Dataverse. If any exception is thrown on the Finance and Operations side, the last message in the error log is returned to Dataverse and is thrown as an InvalidPluginExecutionException exception that contains the message from Finance and Operations. Because the Finance and Operations code runs in the context of the user, the language of the error message is based on the language that is specified on the UserInfo record in Finance and Operations. If any messages that are written to the info log in Finance and Operations don't result in an exception, they aren't shown in Dataverse.

Calculated/unmapped fields

Calculated and unmapped fields in Finance and Operations entities are also available in the corresponding virtual entities in Dataverse.

Application lifecycle management for solutions that use virtual entities

2/18/2021 • 5 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

IMPORTANT

This functionality requires version 10.0.12 for Finance and Operations apps, while service update 189 is required for Dataverse. The release information for Dataverse is published on the latest version availability page.

The application lifecycle for an end-to-end solution using Finance and Operations virtual entities will encompass both Finance and Operations as well as Dataverse. This topic explains this in detail.

Solution management

Virtual entities for Finance and Operations don't exist in Dataverse until they are created. Virtual entities must be created inside a solution. The MicrosoftOperationsERPVE solution is used for this purpose. This solution will contain all the virtual entities that are created from an instance of Finance and Operations.

MicrosoftOperationsERPVE is a managed solution. By definition, a managed solution can't be modified after it has been generated. However, MicrosoftOperationsERPVE is a managed solution that grants privileges to update the components (that is, virtual entities) that are inside it. Therefore, new virtual entities can be added to the solution as they are created, and existing virtual entities can be updated as required. Nevertheless, the privileges to modify the managed solution are available only to the platform itself. Users can't make changes directly to the solution.

Because MicrosoftOperationsERPVE is a managed solution, solutions from customers, partners, and independent software vendors (ISVs) can take a dependency on it. This capability allows for consistent application lifecycle management (ALM) for solutions that use and depend on the virtual entities for Finance and Operations.

When a solution that depends on MicrosoftOperationsERPVE is exported, placeholders for the virtual entities that are used in the solution are added in the exported solution. When that solution is imported into another Dataverse environment, the import process also generates the dependent Finance and Operations virtual entities in the MicrosoftOperationsERPVE solution for the Finance and Operations instance that is connected to the Dataverse environment. Therefore, MicrosoftOperationsERPVE must already exist before a solution that depends on it is imported. Otherwise, an error message is shown. Additionally, if a dependent entity isn't available in the Finance and Operations instance, the virtual entity for that entity won't be generated. Virtual entities are generated only for entities that are available.

> **NOTE**
> If a virtual entity already exists in Dataverse and a solution is being imported that now references new fields in the virtual entity which does not exist in Dataverse, a manual refresh must be performed on the virtual entity to get the latest metadata from Finance and Operations.

The following list describes other solutions that Finance and Operations virtual entities require to work, and that must be available in the Dataverse environment:

- **MicrosoftOperationsERPCatalog** – This solution provides a catalog of the available entities in a Finance and Operations instance. It also provides the connection that is used to set up a configuration. For more information, see the later sections of this topic.
- **MicrosoftOperationsVESupport** – This solution provides the virtual entity provider for Finance and Operations apps. The provider can communicate with Finance and Operations apps and Dataverse. For more information, see the next section.
- **Dynamics365Company** – This solution adds the Company entity, which is referenced by all Finance and Operations entities that have a **PrimaryCompanyContext** metadata value.

All these solutions must be present in an environment. Otherwise, virtual entities won't work with Finance and Operations apps. These solutions are packaged together to allow for easier portability across environments.

Managing entities from multiple environments

The MicrosoftOperationsVESupport solution consists of the **msdyn_financeandoperationsvirtualentity** entity. This entity represents the virtual entity data source for Finance and Operations that captures connection setup information. Each record in this entity represents a connection to a Finance and Operations instance.

A catalog is used to list all the entities in a Finance and Operations instance that are available for virtualization in Dataverse (in other words, all the entities in Finance and Operations that are enabled for Open Data Protocol [OData]). The catalog is part of the default MicrosoftOperationsERPCatalog solution and is applicable to a Finance and Operations instance.

Note that each Dataverse environment must point to only one Finance and Operations instance at any time, and each Finance and Operations environment must point to only one Dataverse environment. Therefore, there should be only one record in the **msdyn_financeandoperationsvirtualentity** entity.

The **mserp_financeandoperationsentity** entity that represents the catalog can be queried to list the entities in a Finance and Operations instance. Because this entity is a virtual entity, the catalog is never persisted in Dataverse.

Notice that the name of the catalog entity has the "mserp_" prefix. This prefix identifies the entities in the catalog as Finance and Operations entities. The same prefix is also added to the system names of the virtual entities that are generated for Finance and Operations in the MicrosoftOperationsERPVE solution. Therefore, the maker can distinguish Finance and Operations virtual entities from other entities. The prefix is set in the managed solution and can't be changed.

Managing entities from multiple ISV solutions

One or more ISV solutions will take a dependency on the MicrosoftOperationsERPVE solution to use virtual entities for Finance and Operations. Because custom entities in Finance and Operations use the same catalog as out-of-box entities in Finance and Operations, the virtual entities for custom Finance and Operations entities will also be generated in the MicrosoftOperationsERPVE solution.

The established guidelines and ALM for entity development in Finance and Operations ensure that there are no conflicting entity names across ISV solutions. Therefore, no conflicts of this type can occur when virtual entities are generated in Dataverse for custom Finance and Operations entities from multiple ISV solutions. All virtual

entities for Finance and Operations entities, including custom entities, will have the "mserp_" prefix that was mentioned earlier.

Managing a Finance and Operation instance in a Dataverse environment for virtual entities

One Finance and Operations instance must be linked to a Dataverse environment for virtual entities. The connection setup information that is required is captured in a virtual entity data source for Finance and Operations. This data source is included in the MicrosoftOperationsERPCatalog solution.

Finance and Operations and Dataverse admin reference

2/18/2021 • 5 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

IMPORTANT

This functionality requires Platform updates for version 10.0.12 of Finance and Operations apps and service update 189 for Dataverse. The release information for Dataverse is published on the latest version availability page.

This topic provides step-by-step instructions about how to set up and configure virtual entities for Finance and Operations apps in Dataverse.

Getting the solution

The Dataverse solution for Finance and Operations virtual entities must be installed from Microsoft AppSource virtual entity solution. For more information, see Finance and Operations virtual entity.

Ensure the following solutions are installed in Dataverse.

- **Dynamics365Company** - This adds the **Company** entity, which is referenced by all Finance and Operations entities with a PrimaryCompanyContext metadata value.
- **MicrosoftOperationsVESupport** - This provides the core support for the Finance and Operations virtual entity feature.
- **MicrosoftOperationsERPCatalog** - This provides a list of available Finance and Operations entities through the mserp_financeandoperationsentity virtual entity.
- **MicrosoftOperationsERPVE** - This is the API-managed solution, which will contain the generated virtual entities as they are made visible.

Authentication and authorization

After the solutions are imported in the Dataverse environment, both environments must be set up to connect to each other. Dataverse will call Finance and Operations using Service-to-Service (S2S) authentication, based on an Azure Active Directory (AAD) application. This new AAD application represents the single instance of the Dataverse environment. If you have multiple pairs of Dataverse and Finance and Operations environments, separate AAD applications for each pair must be created to ensure connections are established between the correct pair of Finance and Operations and Dataverse environments. The following procedure shows the creation of the AAD application.

> **IMPORTANT**
> The AAD application must be created on the same tenant as Finance and Operations.

1. Go to https://portal.azure.com > **Azure Active Directory** > **App registrations**.
2. Select **New Registration**. Enter the following information:
   - **Name** - Enter a unique name.
   - **Account type** - Enter **Any Azure AD directory** (single or multi-tenant).
   - **Redirect URI** - Leave blank.
   - Select **Register**.
   - Make note of the **Application (client) ID** value, you will need it later.
3. Create a symmetric key for the application.
   - Select **Certificates & secrets** in the newly created application.
   - Select **New client secret**.
   - Provide a description and an expiration date.
   - Select **Save**. A key will be created and displayed. Copy this value for later use.

The AAD application created above will be used by Dataverse to call Finance and Operations apps. As such, it must be trusted by Finance and Operations and associated with a user account with the appropriate rights in Finance and Operations. A special service user must be created in Finance and Operations with rights *only* to the virtual entity functionality, and no other rights. After completing this step, any application with the secret of the AAD application create above will be able to call this Finance and Operations environment and access the virtual entity functionality.

The next steps walk through this process in Finance and Operations apps.

1. In Finance and Operations, go to **System Administration** > **Users** > **Users**.
2. Select **New** to add a new user. Enter the following information:
   - **User ID** - Enter **dataverseintegration** (or a different value).
   - **User name** - Enter **dataverse integration** (or a different value).
   - **Provider** - Set to **NonAAD**.
   - **Email** - Enter **dataverseintegration** (or a different value, does *not* need to be a valid email account).
   - Assign the security role **CDS virtual entity application** to this user.
   - Remove all other roles including **System user**.
3. Go to **System Administration** > **Setup** > **Azure Active Directory applications** to register Dataverse.
   - Add a new row.
   - **Client ID** - The **Application (client) ID** created above
   - **Name** - Enter **Dataverse Integration** (or a different name).

- **User ID** - The user ID created above.

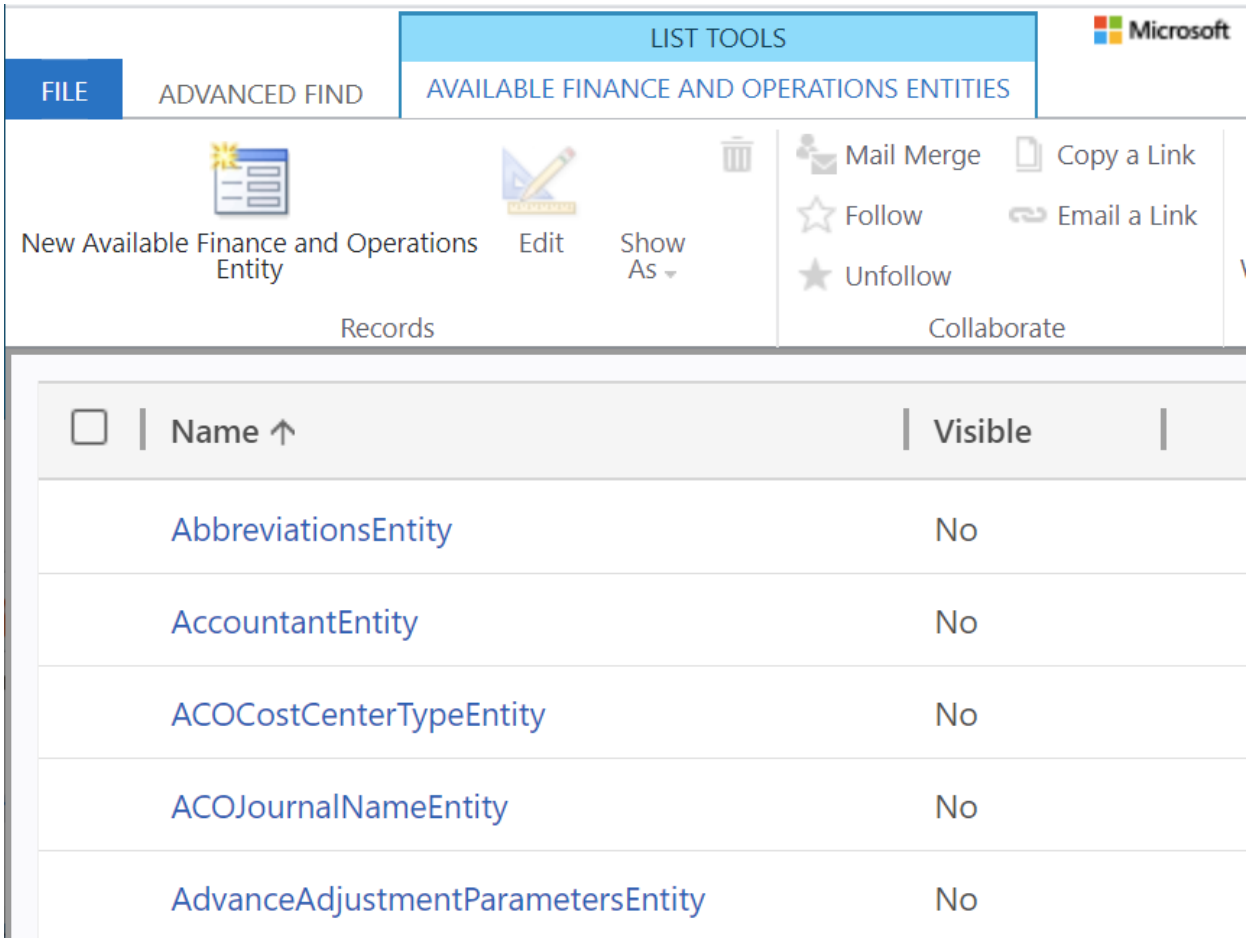
The next step in the process is to provide Dataverse with the Finance and Operations instance to connect to. The following steps walk through this part of the process.

1. In Dataverse, go to **Advanced Settings > Administration > Virtual Entity Data Sources**.
2. Select the data source named "Finance and Operations".
3. Fill in the information from the steps above.
   - **Target URL** - The URL at which you can access Finance and Operations.
   - **OAuth URL** - https://login.windows.net/
   - **Tenant ID** - Your tenant, such as "contoso.com".
   - **AAD Application ID** - The **Application (client) ID** created above.
   - **AAD Application Secret** - The secret generated above.
   - **AAD Resource** - Enter 00000015-0000-0000-c000-000000000000 (this is the AAD application representing Finance and Operations, and should always be this same value).
4. Save the changes.

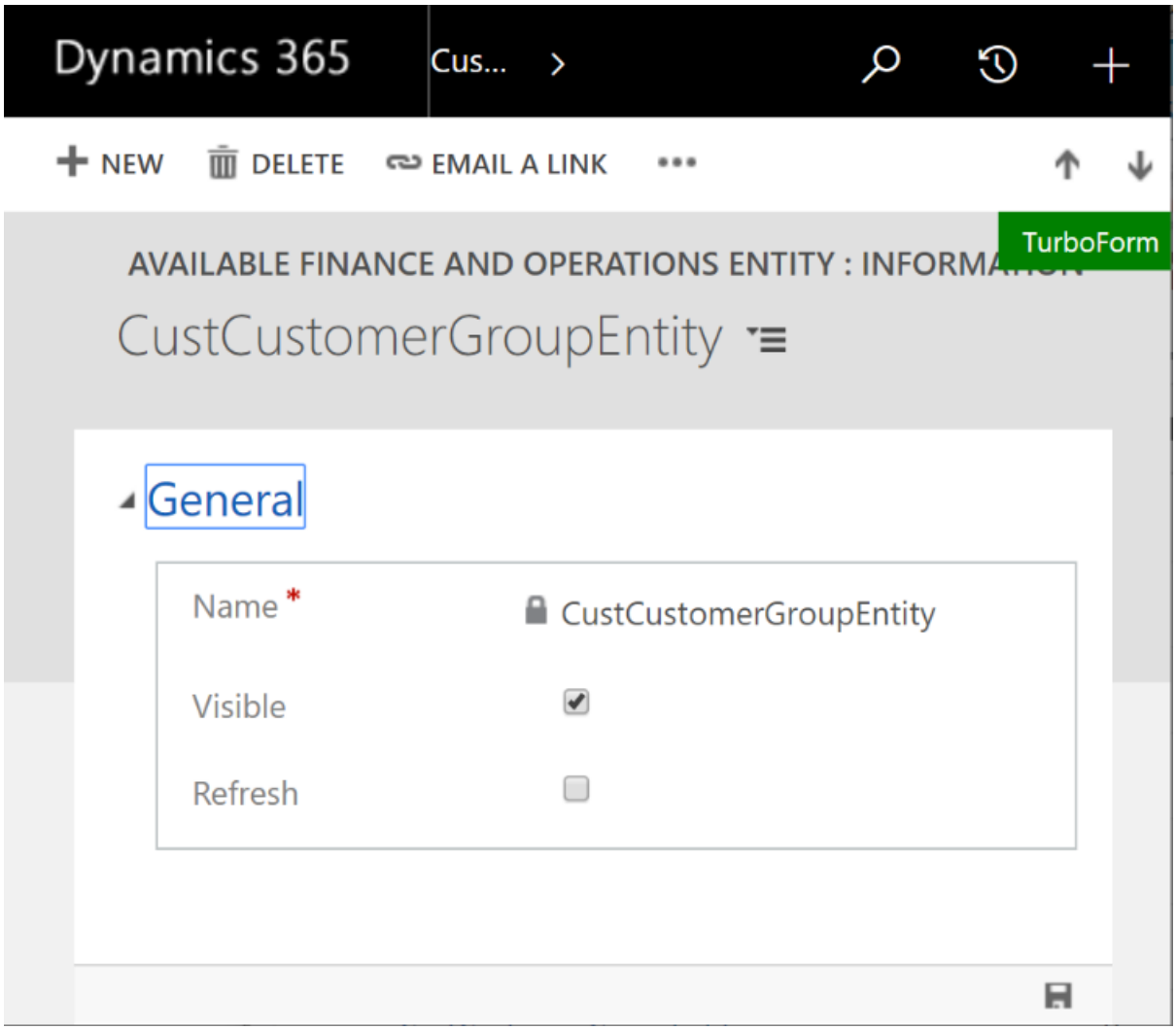
Enabling virtual entities

Due to the large number of OData enabled entities available in Finance and Operations, by default, the entities are not available as virtual entities in Dataverse. The following steps allow for enabling entities to be virtual, as needed.

1. In Dataverse, go to **Advanced find** (filter icon).
2. Look for "Available Finance and Operations Entities" and select **Results**.



3. Locate and open the entity that you want to enable.
4. Set **Visible** to **Yes** and save. This will generate the virtual entity, so that it will appear in all of the appropriate menus, such as the advanced find dialog box.



Refreshing virtual entity metadata

The virtual entity metadata can be force-refreshed when it is expected for the entity metadata in Finance and Operations to have changed. This can be done by setting **Refresh** to **Yes** and saving. This will sync the latest entity definition from Finance and Operations to Dataverse and update the virtual entity.

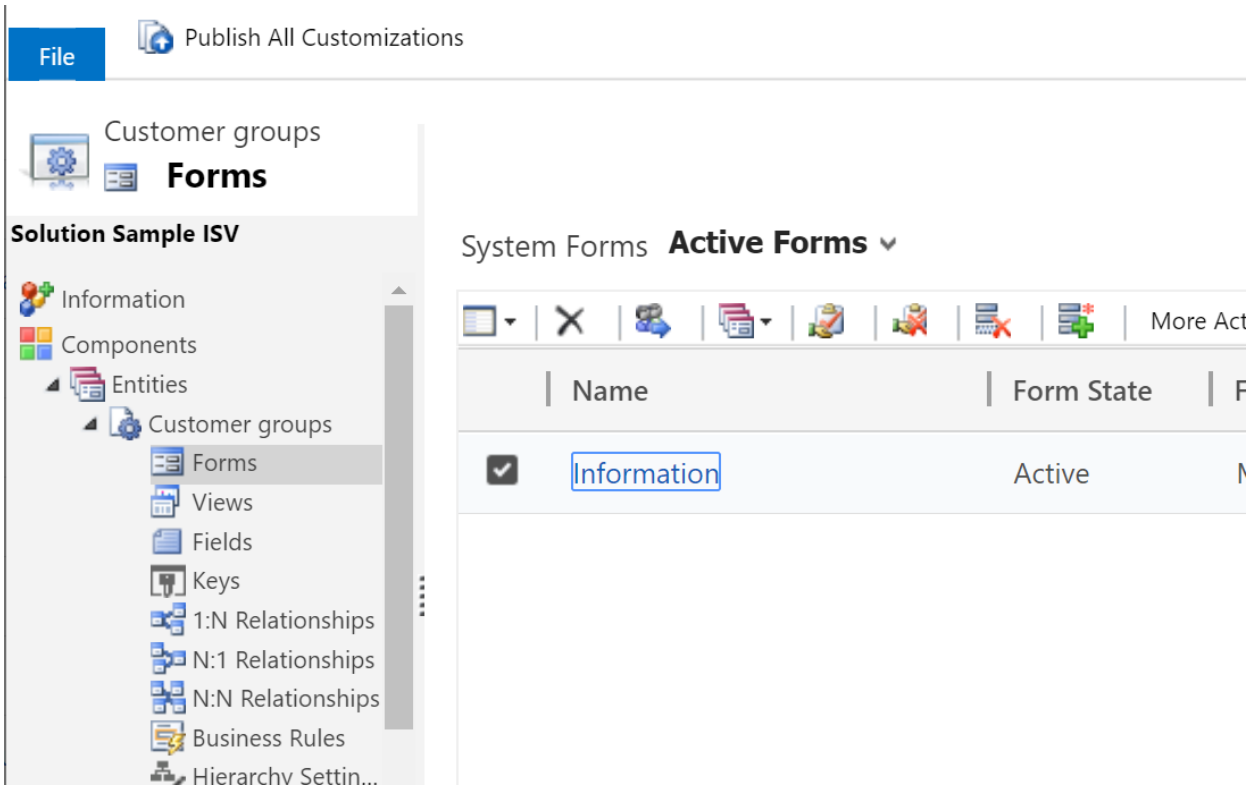
Referencing virtual entities

The virtual entities are all generated in the MicrosoftOperationsERPVE solution, which is API Managed. That means the items in the solution change as you make entities visible/hidden, but it is still a managed solution that you can take dependency on. The standard ALM flow would be to just take a standard reference to a virtual entity from this solution with the **Add existing** option in the ISV solution. It will then show as a missing dependency of the solution and be checked at solution import time. During import if a specified virtual entity does not yet exist, it would automatically be made visible without needing additional work.

To consume virtual entities:

1. Create a separate solution as usual in Dataverse, which will contain the consuming logic.
2. Select **Entities** > **Add Existing**. Select the virtual entity that you want to reference from the list.
3. When prompted to select assets to add, select any forms, views, or other elements that you want to customize, then select **Finish**.

From the development tooling, existing elements such as forms can be modified for the virtual entity. Additionally, new forms, views, and other elements can also be added.



When the solution is exported, it will contain hard dependencies on the virtual entity generated in the MicrosoftOperationsERPVE solution.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Power Apps portals with Finance and Operations

2/18/2021 • 3 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

IMPORTANT

This functionality requires version 10.0.12 for Finance and Operations apps, while service update 189 is required for Dataverse. The release information for Dataverse is published on the latest version availability page.

Power Apps portals will enable create, update, and delete (CRUD) operations to Finance and Operations entities that are available as virtual entities in Dataverse. This topic explains the scenarios that are implemented in Power Apps portals for Finance and Operations apps.

Anonymous access from Power Apps portals

Collaboration scenarios in business processes such as bidding or onboarding of prospects in Finance and Operations require that external users participate from the Power Apps portal, even though they aren't users in Finance and Operations apps. The simplicity of anonymous access is appealing in these types of scenarios because the users, who might not be Finance and Operations apps users, don't have to sign in. However, they are expected to perform CRUD operations in Finance and Operations to complete any meaningful tasks in the business processes.

To ensure that only the required entities are enabled for anonymous access, a user in Finance and Operations must be designated as the user who is used for anonymous access. This designation is configured in the **Anonymous portal access user ID** field on the **Virtual entity** tab on the **System parameters** page (**System administration > System parameters**). The designated user can then be assigned to duties and security roles to control access to specific data that must be made available to all users who will interact anonymously from the Power Portal.

Note that because this scenario involves anonymous access, the only user context that matters, from a security perspective, is the user who is designated in the **Anonymous portal access user ID** field.

Authenticated access from Power Apps portals

Fully authenticated user access from Power Apps portals to Finance and Operations lets users in Finance and Operations also interact from Power Apps portals. A user who signs in to the Power Apps portal is also a known user in Finance and Operations who has appropriate security roles based on job requirements. These roles govern the security access to data for the authenticated user in Power Portal. In addition, any Finance and Operations user that is expected to also use Power Apps portal to access Finance and Operations data must also belong to the **CDSVirtualEntityAuthenticatedPortalUser** security role. This provides an additional layer of security and also provides a way to know the total users that are authorized to access from Power Apps portals.

Because Power Apps portals authentication is linked to the Contacts entity in Dataverse, a mapping must be established between the Dataverse contact and the corresponding user in Finance and Operations. This mapping can be done by adding entries to the **msdyn_externalportalusermapping** entity. From a security perspective, the scope of virtual entities that are made available to authenticated users must be configured as **Global** in the Power Apps portal.

When authenticated users from a different tenant need to be added to Finance and Operations as users, you must use the Create new user process in Finance and Operations. This process adds cross-tenant users as Microsoft Azure Active Directory (Azure AD) business-to-business (B2B) guest users.

> **NOTE**
>
> Access from the Power Apps Portal will fail if the user (authenticated or anonynous) has been assigned the System administrator role in any Finance and Operations apps.

Finance and Operations virtual entities FAQ

2/18/2021 • 7 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

IMPORTANT

This functionality requires version 10.0.12 for Finance and Operations apps, while service update 189 is required for Dataverse. The release information for Dataverse is published on the latest version availability page.

This topic is a collection of frequently asked questions about Finance and Operations virtual entities.

Do Tier 1 Finance and Operations environments or demo topologies work?

Yes, Tier 1 and DEVTEST and DEMO topologies should work.

What version of Finance and Operations do I need?

10.0.12 is the minimum version that is required.

Can a solution from an independent software vendor (ISV) take a dependency on virtual entities? What does the application lifecycle management (ALM) look like?

Yes. The virtual entities are all generated in the MicrosoftOperationsERPVE solution, which isAPI-managed In other words, the items in the solution change as you make entities visible or hidden, but the solution is still a managed solution that you can take dependency on. The standard ALM flow just takes a standard reference to a virtual entity from this solution with the **Add existing** option in the ISV solution. Missing dependency of the solution will be checked when the solution is imported and during import, if a specified virtual entity doesn't yet exist, the virtual entity is automatically made visible.

Which entities from Finance and Operations do users see in the catalog in Dataverse?

Generally, users see all entities where **IsPublic** is set to **Yes**. These entities are the same entities that are currently visible in Open Data Protocol (OData).

Do all Microsoft Power Platform users have to be users in Finance and Operations?

Any **interactive user** of Microsoft Power Platform who tries to access Finance and Operations data through a virtual entity must also exist as a user in Finance and Operations. Therefore, technically, not *all* users have to be users in Finance and Operations. Only those users who access Finance and Operations data through virtual entities must be users in Finance and Operations.

A **S2S application user** can also be used to call into virtual entities. For this kind of integration, the application user must be set up in **System administration > Setup > Configure Azure Active Directory Applications**. This allows for applications to integrate with Finance and Operations using virtual entities.

Where do I find the catalog entity?

In the **Advanced find** window, the entity is named **Available Finance and Operations Entities**.

Is there a way to specify a company when I perform data operations on a virtual entity?

Yes. Although the company is implicit in Finance and Operations, it's an explicit field on each company-striped entity in Dataverse. You can use either the **Company Code** field, where the value is a four-character string, or the **Company** field, which is a lookup to cdm_Company. Both approaches provide the same information.

Can I change the prefix for the virtual entities?

No. All Finance and Operations virtual entities should be generated in the MicrosoftOperationsERPVE solution, and they should all have the "mserp_" prefix. This prefix should not be changed. If you have a scenario where you believe the prefix has to be changed, you should share that scenario with Microsoft.

How can I filter data in an app that is created by using Power Apps, based on the current user or any other dynamic criteria, such as today-10?

You can write a pre-operation plug-in on the RetrieveMultiple message of the entity and change the criteria on the query in it. Alternatively, you can write a post-operation plug-in to filter the results before they are returned.

Can I pin a model-driven app into Finance and Operations?

No, it isn't currently possible to pin a model-driven app into Finance and Operations.

How can I show, in the same grid, data from multiple virtual entities that are joined to a physical entity record in Dataverse?

This approach isn't currently possible in Dataverse.

How do I add subcomponents in the new Power Apps experience?

As in the previous Power Apps user interface (UI), you must redo the **Add Existing** operation. After the solution is selected, and Customer Groups has already been added as an entity, follow these steps.

1. Select **Add existing** > **Entity**.
2. Select customer group entity, and then select **Next**.
3. Under **Components**, select **Select components**.
4. Select the fields, relationships, and forms that you want, and then select **Add**.

If I want a default value to be entered in a field during pre-create, will an initValue on the data entity work?

Yes. Here is the order of calls:

1. Dataverse sends a create or update message.
2. All the existing logic on the Finance and Operations entity and backing tables is invoked. This logic includes default value entry that might change values.
3. Dataverse sends another Retrieve (single) message to get the latest copy of the data, including any fields that default values were entered for.

Can I debug Finance and Operations when we do a create, read, update, and delete (CRUD) operation from Dataverse? If so, which process do I have to attach?

Yes, to debug in Finance and Operations, open Visual Studio as an admin. Typically, Finance and Operations apps run under w3wp.exe as a process. However, when you open Visual Studio as an admin, IISExpress.exe is automatically opened, and Finance and Operations is hosted there. You can attach to IISExpress.exe (or to w3wp.exe if not running Visual Studio as an admin). To set breakpoints in the virtual entity code, find the **CDSVirtualEntityAdapter** and **CDSVirtualEntityController** classes. The adapter class is the first class that is called, and it only does serialization/deserialization. It then delegates to the controller class to do the actual queries. Therefore, the controller class is usually the easiest place to put breakpoints.

Does the form business logic in Finance and Operations get called through virtual entities?

Finance and Operations business logic that resides on forms isn't invoked through virtual entities. Instead, you should expect the same behavior that you get through OData access to the same entities. The expectation is that an entity that is exposed to OData (that is, **IsPublic** is set to **Yes**) has appropriate protections to ensure that data

can't be corrupted. If any entity lacks this protection, that situation represents a bug in the entity. If you see differences in entity behavior between OData and virtual entities, that situation represents a bug in the virtual entity feature.

If I develop a new Finance and Operations entity and want to see it in Dataverse, do I have to select Refresh entity list in Finance and Operations? Do I have to do anything in Dataverse?

In theory, no, you don't have to refresh the entity list. At most, you might have to either reset Internet Information Services (IIS) or restart IIS Express, depending on where Application Object Server (AOS) is running. The fact that the list of entities is accurate is cached in SysGlobalObjectCache, which is a per-process cache. Any time that this cache doesn't indicate that the list is accurate, the list is rebuilt. The rebuild process takes about five seconds. Therefore, when you restart your AOS process (w3wp.exe or iisexpress.exe), the list will be accurate the next time that you query it from Dataverse. Additionally, although recompilation *should* flush the SysGlobalObjectCache cache, it might not. In that case, an AOS restart will flush it.

Do you have guidance on when to use a virtual entity and when to use dual-write?

Dual-write is only provided for a few key data entities where the data needs to be natively in Dataverse. Those data entities are not available as virtual entities.

When adding records using virtual entities is there any way to use number sequences?

Yes, if the Finance and Operations entity can auto generate number sequences, then it will work the same way from the virtual entity.

Why does 'search view' not work in Power Apps?

If there are no fields added in the quick find view for the entity, then the search box does nothing. The workaround is to add one or more fields of the entity to the quick find view.

Mobile platform resources

2/18/2021 • 11 minutes to read • Edit Online

By using mobile apps, you can reuse business logic and modeling. Mobile apps enable rich offline and mobile interactions, and provide an easy-to-use designer experience. Developers can create simplified forms in Microsoft Visual Studio and then design mobile apps that expose this functionality. The mobile platform makes it easy to change the forms and mobile app definitions to include customizations that are made to your cloud app.

Get started

- Getting started
- Architecture
- Page design guidelines
- Action design guidelines
- Form design requirements

Check out the following series of how-to videos that show how to create a mobile app.

- Tutorial 1: Building the sales order page
- Tutorial 2: Building the sales order details page
- Tutorial 3: Building the create new sales order action
- Tutorial 4: Adding a lookup to the create new sales order action
- Tutorial 5: Adding a lookup and hiding pages using mobile business logic

Common configurations

These topics describe some common customizations that you can add to your mobile app.

- Localize mobile workspaces
- Help secure mobile workspaces
- Set up clickable fields
- Set up mandatory fields through workspace classes
- Display item counts in a field

Client-side development

Client-side APIs are used in the business logic file, which provides an extensibility layer to the mobile workspace that allows for customization. Some things that you can access and influence through the client-side APIs include:

- Metadata
- Runtime control/page instances
- Business data
- Offline-first business behaviors
- Layout and style

The process for client-side development is described in these topics:

- Client-side design APIs overview

- Business logic events overview
- Client APIs

You can download a sample business logic file (with a .js file name extension) for the Reservation management workspace. Go to Dynamics365-for-Operations-mobile-FleetManagementSamples, open the **business_logic folder**, and locate the FM.js file

Server-side development

Workspace attributes and classes are used to create, configure, and publish workspaces on the server. These server-side X++ APIs can be used instead of using the task recorder-based mechanism to build a workspace. Workspaces created using either mechanism can then be styled and augmented using the client-side APIs.

Server-side development is described in these topics:

- Workspace class overview
- Server APIs (X++)

You can download the sample project (with an .axpp file name extension) for the Fleet Management mobile app. Go to Dynamics365-for-Operations-mobile-FleetManagementSamples and download the **FMMobileApp.axpp** file.

Debugging during development

During development it can be useful to attach a debugger to get more detailed information and insight into what is happening in the background. A web debugger can be used with the client-side JavaScript logic and styling and the Visual Studio debugger can be used with the server-side X++ business logic.

Debugging the client side

Prerequisites

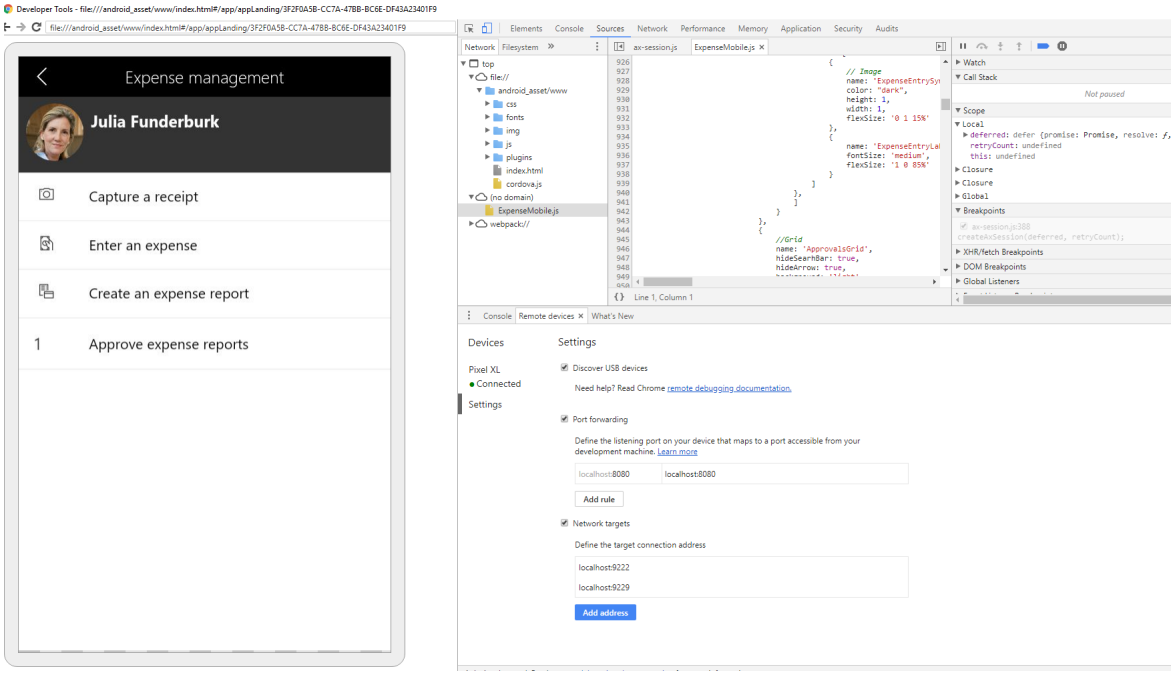
- Android device plus PC
- Azure-hosted development machine (so the mobile device can point to it)

Steps to debug the client side

1. On the web client that is exposed by the Azure-hosted development machine, ensure that there are mobile workspaces published for the Finance and Operations app. For information about publishing a mobile workspace, see Publish mobile workspaces.
2. Install the Android debug apk for the Finance and Operations app on an Android device:
    - One time only, allow the installation of apk files - Go to **Menu** > **Settings** > **Security** and then check **Unknown Sources** to allow the phone to install apps from sources other than the Google Play Store.
    - Uninstall the Finance and Operations app - Ensure that any previous version of the Finance and Operations app has been uninstalled.
    - Download the apk file - From the device's browser, navigate to the latest Finance and Operations Android debug apk on GitHub and click **Download** (or use this direct link to the file).
    - Install the Finance and Operations apk file - Confirm install of the Finance and Operations app via the apk file.
    - Run the debug Finance and Operations app on the device and sign in.
3. Connect to the device from the debugging machine.
    - On the Azure-hosted development machine or a separate PC, follow Android developer instructions to Get Started with Remote Debugging Android Devices. You can also find a wide selection of instructional videos on YouTube by searching for Chrome for Android remote debugging.
4. After you connect the debugger, find the active tab on your device. You may need to click **View more**

tabs on Android. One of the tabs should look similar to `/www.index.html#/app/appList` or `/www.index.html#/app/app_landing`.

Expand the nodes to find the workspace JavaScript, such as **File** > (no domain) > **ExpenseMobile.js**. Click the JavaScript file to view it and add breakpoints.



5. Reflect the mobile device on your desktop so that you can interact with it on the desktop screen.
6. Go to through the desired workspace and forms.
7. If breakpoints are encountered, then the browser developer tools will allow you to control the flow of execution and see the values and parameters being passed.
8. To change styling at runtime, use the elements tab to alter the styling. This will help you determine what elements JavaScript should target and how those elements should be styled.
9. If a needed change is identified, make those changes in JavaScript, and then push those changes into the environment.
10. If more changes or validation is needed, repeat the process.

Debugging the server side

Prerequisites

- Azure-hosted development machine (so the mobile device can point to it)

Steps to debug the server side

1. On the web client exposed by the Azure-hosted development machine, ensure that there are mobile workspaces published for the Finance and Operations app. For information about publishing a mobile workspace, see Publish mobile workspaces.
2. Open the app on your device, point to the Azure-hosted development machine, and sign in.
3. Open Visual Studio on the Azure-hosted development machine and attach the debugger to the w3wp process.
4. After you connect the debugger, find the desired business logic, and insert breakpoints as needed.
5. Either use the app on your device as usual, or reflect the mobile device on your desktop so you can interact with it on the desktop screen.
6. Navigate through the desired workspace and forms.

7. If breakpoints are encountered, then Visual Studio will allow you to control the flow of execution and see the values and parameters being passed.
8. If a needed change is identified, make those changes in X++, and push those changes into the environment.
9. If more changes or validation is needed, repeat the process.

Troubleshooting the app

[Resolved] - No support for iOS14 due to issues with date and time controls

Version 2.2.8 of the Finance and Operations mobile app fixes the known issues with the date and time pickers in iOS14. Ensure that you have the latest version of the app if you are experiencing issues running the application on iOS14.

The Mobile Client app is not working on particular devices

Sometimes the cache associated with the app becomes corrupt or obsolete and needs to be cleared. Unfortunately, the only way to clear the data associated with the app is to uninstall the app. To completely uninstall the app, don't use the "long-press wiggle and x on the app icon" method. Instead, completely uninstall the app by navigating to **Settings** > **General** > **iPhone Storage** > **Finance and Operations (Dynamics 365)**, and then click **Delete App**. After 10-15 seconds, the app can be reinstalled.

On Android devices with non-English regions, the comma can't be used as the decimal separator in an amount field

On Android devices with non-English regions, using a comma as the decimal separator is standard practice. Problems using a comma in an amount field is an Android-specific problem because iPhone works as expected. On Android, use of the comma in an amount field is a problem with the default "gboard" keyboard and some other keyboards. Installing the SwiftKey keyboard (published by Microsoft) allows the entry of commas just like on iPhone: SwiftKey Keyboard.

Change needed for ADFS to support Mobile Client in on-premises environments

If Active Directory Federation Services (ADFS) is in use on the domain and the environment is on-premises, then **ADFS must be configured to provide a regular forms-based authentication screen** instead of using Windows Integrated Authentication (WIA). The Finance and Operations apps for iOS and Android require the regular forms-based authentication screen. ADFS should be configured to only provide WIA for browser clients (use cases). For more information, see Configure intranet forms based authentication for devices that do not support WIA.

Using multi-factor authentication with the Finance and Operations app

The Finance and Operations (Mobile Client) app facilitates user authentication with Azure Active Directory (Azure AD) by presenting the Azure AD sign-in web page within an embedded browser. After a successful sign in, it will retrieve the user token from the cookies and use that when communicating with the user interaction service that it shares with the web client. Some multi-factor authentication mechanisms that involve switching to a different app on the same device will cause the embedded browser to close, so the sign in will fail. The workarounds for this include:

- Different device - Use a different device for the multi-factor authentication response so the app remains active on the original device.
- Multi-factor authentication via phone call - Use a phone call for the multi-factor authentication response so an app switch is not needed.
- Use the "touch and hold" gesture on the authentication notification and then select the **Accept** option. Because the notification acceptance will not require an app switch, the sign in will proceed as usual.

If there are continued problems with MFA authentication, it is helpful to submit the Microsoft Authenticator app

logs and provide support with the resulting Incident ID.

Intune support and conditional access

The Finance and Operations (Mobile Client) app does not have Microsoft Intune policies implemented, so it does not support Intune. Manually adding the app (following Add iOS store apps to Microsoft Intune) is also not supported because the device identifier cannot be passed.

Trouble signing out of the app and signing in with new credentials

If you experience trouble signing out of the app and signing in with new credentials, then you might need to "forget old credentials" on the Azure AD sign-in screen.

- To sign out of the app, follow these steps:
  - Open the app.
  - Sign out of the app.
  - Force close the app.
- To forget old credentials, follow these steps:
  - Open the app.
  - Connect to the server.
  - On the Azure AD sign-in screen, if there are saved credentials, select the ellipsis (...) button on that card, and then select **Forget the credential**.
  - Force close the app.
- To sign in to the app, follow these steps:
  - Open the app.
  - Connect to the server
  - Sign in using the Azure AD sign-in screen.

Troubleshooting app content

I can't figure out how to build or change something in my Mobile Client content

There are many resources that you can leverage to figure out how to build or change content for the Mobile Client.

- Review the documentation provided in the Help system.
- Review the Fleet Management Samples for examples.
- Publish and review the Expense Management workspace, and other standard workspaces, for examples. Demo data for the USSI company is useful when using the Expense Management workspace. The forms and X++ code that make up the Expense Management workspace can be found in the Application Explorer by searching for the "ExpenseMobile" prefix.
- Leverage the Dynamics Community forums by searching for answers and asking questions when needed.

Tips for workspace creation and modification

Here are some tips for workspace creation and modification:

- Create new simplified forms for recording rather than recording large complex forms.
- After recording a form, you have to close the form instead of clicking **Done**, otherwise the form remains open.
- Verify that recordings are correct using the "Job steps".
- Play back recordings using task recorder playback to verify them.
- Don't navigate to a page before starting the recording, because the context from the previous page might need to be captured.
- If you re-record a page with a grid then you need to re-record the link to the Details page because otherwise it won't be there.

- When recording an action, change the value of the fields to add them. When recording is complete, close the form instead of clicking **Save**.
- Lookups in mobile are list pages that have been recorded. **Select field data** and **Select field to display** are used to select the **field to use as the value to save** (data) and the **field to show the user** (display).
- When adding a lookup field, select a value in a lookup instead of just adding the lookup field. This will ensure that the correct value is selected.
- If you re-record a lookup, all the references also need to be re-recorded because the GUID for the lookup will change.
- If you want to add a field to a page, you need to add all the fields again, because the list is cleared at the beginning of each edit. This is a limitation of task recorder. Note that reordering is also not possible.
- In the workspace XML, GUIDs are used as references to forms and controls instead of names. GUIDs are used to ensure uniqueness, but this comes at the cost of maintainability. Those GUIDs are regenerated on each modification, so partial edits are very difficult. The use of GUIDs would be very costly to change, so it is unlikely that changes would be made in the future to use simpler string name references.
- Relationships between form datasources need to be via RecordId instead of string. For example, primary keys of the datasources should not be strings.
- Customers and partners can fork a workspace by creating a copy of it and then make changes as needed.
- There is no check box in mobile. You have to manually bind the field to a Yes/No enum in JavaScript.

Common problems with form recordings

Avoid using forms with these patterns and controls when creating workspace recordings:

- Datasources with DelayedJoin (common on transaction forms).
- FastTabs (common on existing forms).
  - Recorded forms should not have FastTabs because the FastTabs expansion state can interfere with playback.
- Any user interface (UI) that has state, like an expandable or hide/show region.
- There is no check box in mobile. You have to manually bind the field to a Yes/No enum in JavaScript.

Get started with the mobile platform

2/18/2021 • 2 minutes to read • Edit Online

After you acquire a development environment, complete the following procedures to get started with development.

Get the Fleet Management mobile forms

We have created new, purpose-built forms in the **Fleet Management** module. These forms are used specifically for the mobile app and aren't meant to be used through the web client.

1. Download the file that contains the Fleet Management project (.axpp file).
2. Extract the contents of the zip file to a temporary location on the development computer.
3. Import the project (.axpp) file by using Microsoft Visual Studio (click **Finance and Operations** > **Import Project**).
4. After you've imported the project file, build the project or module.

Get the sample workspace

We provide a sample workspace for Reservation management. This workspace is based on the **Fleet Management** module.

1. Download the file that contains the sample workspace (.xml file).
2. Sign in to your non-production client. (You must sign in as an administrator.)
3. In the address bar, add **&mode=mobile** to the end of the URL, and then press Enter.
4. In the client, go to **Settings** > **Mobile app**. The mobile app designer will appear docked next to the client.
5. Click the **Overflow** button (...), and then click **Import**.
6. Click the **Browse** button that appears at the bottom of the page.
7. In the file selection dialog box that appears, select one of the XML files that you previously extracted from the zip file.
8. After the app has been loaded into the mobile app designer, click **Done** at the bottom of the page.
9. Click **Publish workspace**.

Get the mobile app

The mobile app is being made available for the most popular mobile operating systems. You must have a Dynamics 365 Unified Operations instance and valid user credentials in order to log in to the app.

- Android (available now) - Finance and Operations mobile app on the Google Play Store
- iPhone (available now) - Finance and Operations mobile app on the iTunes apps store

You're done! Launch the app from your mobile device to see the sample workspace.

Additional resources

Architecture

Client APIs reference

Server APIs reference

NOTE

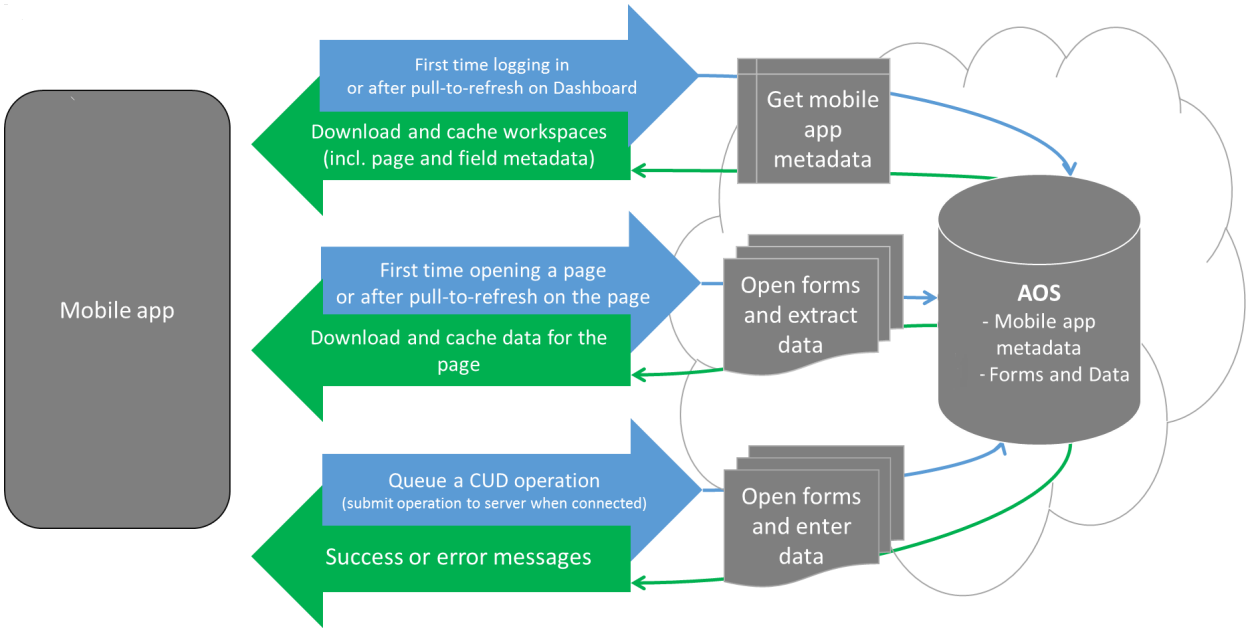
Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Architecture and design considerations for the mobile platform

2/18/2021 • 3 minutes to read • Edit Online

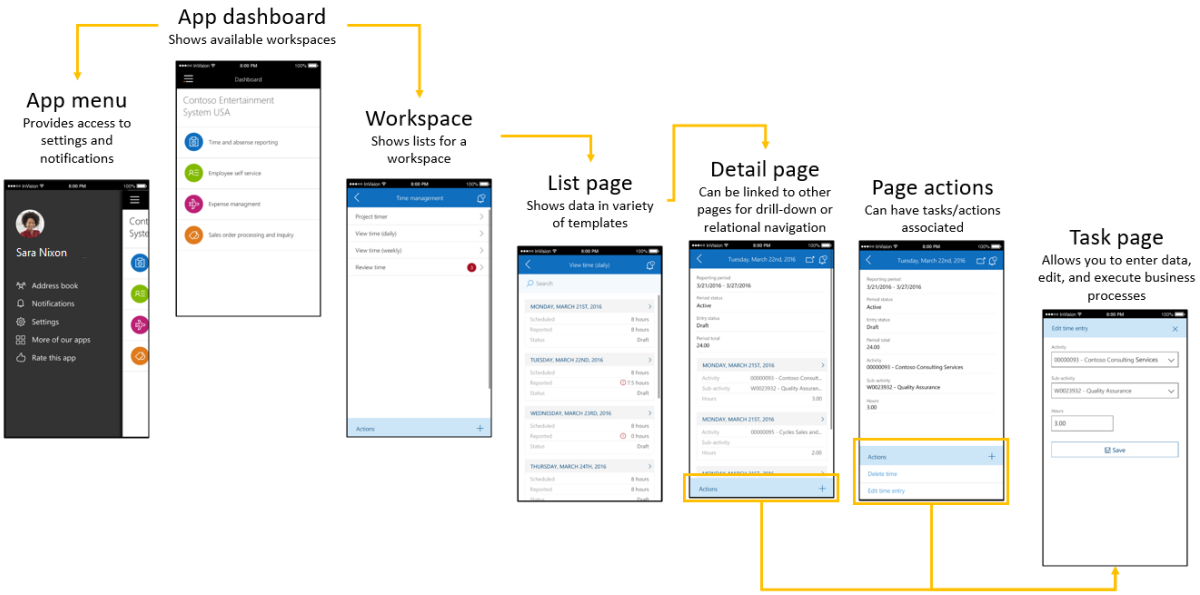
The mobile app communicates with Application Object Server (AOS) to get the metadata for the mobile workspaces (and the pages and the fields that appear on the page), and to get the data for the fields on the pages. Each time that the mobile app requests data for a page, AOS creates a new session that uses the context of the user who is using the mobile app. AOS then uses the user's context to open the corresponding forms (by using the corresponding menu items). AOS can open multiple forms in quick succession and perform actions on those forms (for example, filtering, opening FactBoxes, changing tab pages, and clicking buttons). Any business logic on the forms is also run as usual. Through that process, AOS collects the data values from the requested fields and then sends that data back to the mobile app.



The mobile app platform doesn't assume connectivity to Finance and Operations apps. Activities such as navigation, data view, and data entry don't require server connectivity after data has been cached.

Understanding navigation in the mobile app

Navigation in the mobile app consists of four simple concepts: the dashboard, workspaces, pages, and actions.



- When you start the app, you land on the **dashboard**. On the **dashboard**, you can see a list of **workspaces** that are published in your environment.
- In each **workspace**, you can see a list of **pages** that are available for that workspace.
- On a **page**, you can view data that is collected from one or more forms.
- From a **page**, you can navigate to other **pages** for related data, such as an entity details or lines.
- On a **page**, you can see a list of **actions** that are available for that page.
- **Actions** let you create or edit existing data.

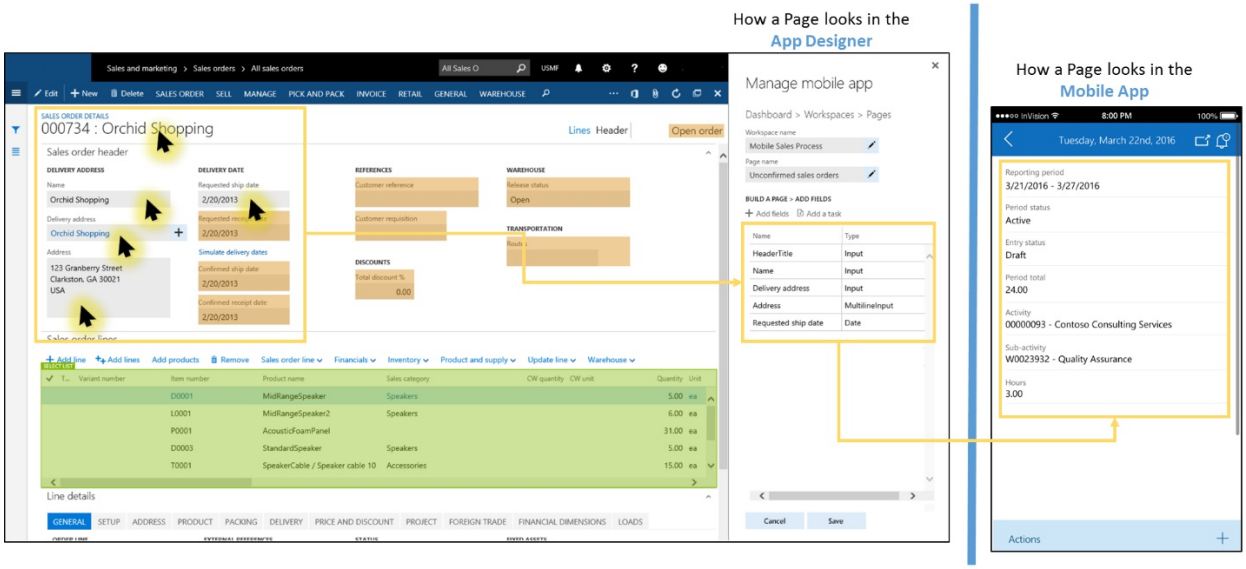
Notes

At any time, you can pull-to-refresh in the mobile app to make the mobile app update its data or metadata. After you edit an existing workspace or publish a workspace, be sure to pull-to-refresh in the mobile app, in either the list of workspaces (if you added a workspace or business logic) or the list of pages (if you modified a page or an action). Workspaces that have been published are visible to all users. In Platform update 3, menu item security automatically hides pages that the user doesn't have access to. If a user doesn't have access to any pages in a workspace, the workspace itself is hidden.

Using the mobile app designer

The mobile app designer lets you select the specific data fields from forms that should appear in the mobile app.

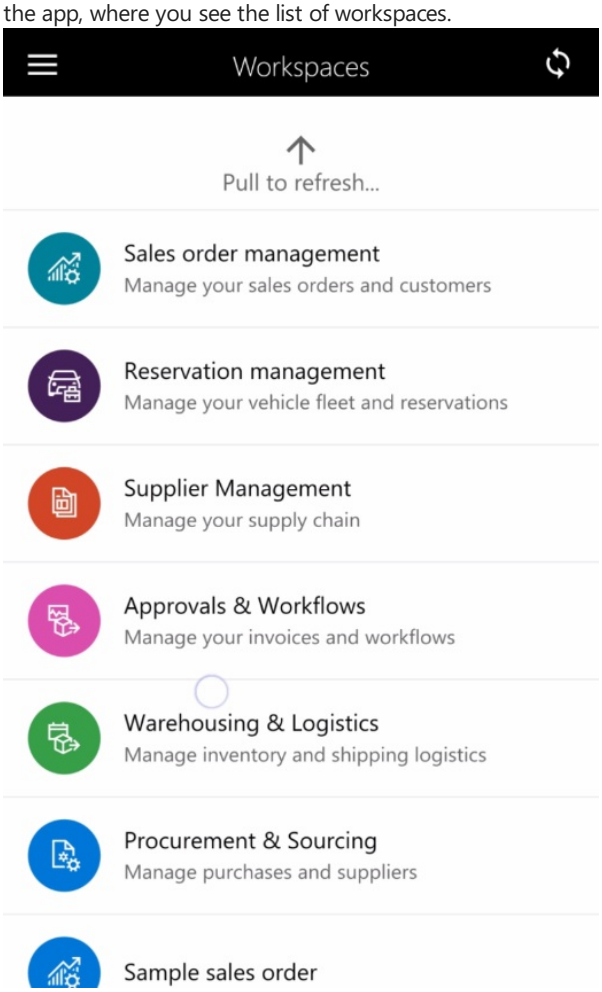
A mobile workspace can be created through designer, using X++ attribute APIs or a combination of both. See Configure workspaces by using the SysAppWorkspace class for more details on using X++ APIs for building a mobile workspace.

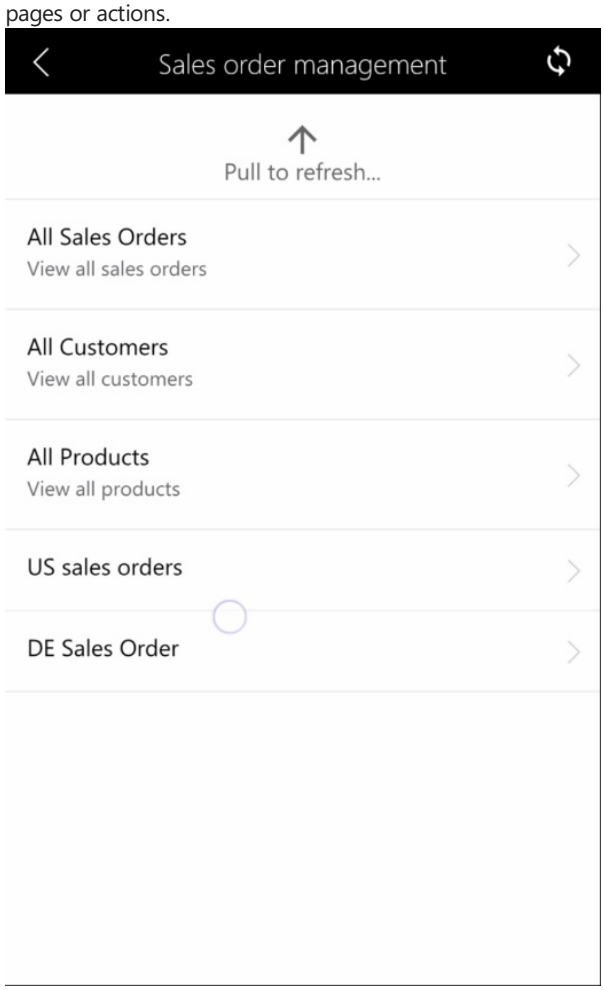


1. Open the client.
2. Go to **Settings** > **Mobile app**.
3. Create a new workspace, or select an existing workspace to edit.
4. Specify the name of the workspace, an icon, and a color.
5. Add pages to the workspace, or edit an existing page.
6. Specify the name of the page.
7. Click **Select Fields** to select the data fields to add to the page.
8. Open the forms that have the data fields that you want to add, and then click the yellow plus sign (+) that appears next to the fields. The fields are added in the order that you select them in. You can add fields from multiple forms, in any order.
9. When you've finished selecting fields, click **Done**.
10. If you've added a field list to the page, you will see that the **List** type is specified for one of the items in the field list. You can optionally add a details page for items in that list by following these steps:
    a. Select the list by clicking on it in the designer.
    b. Click **Add details page**.
    c. Repeat steps 6 through 10 as you require.

Refreshing the app after you make changes

TYPE OF CHANGE	DESCRIPTION

TYPE OF CHANGE	DESCRIPTION
New workspaces, deleted workspaces, or changes to the name, color, or icon of a workspace	Pull-to-refresh from the main landing page (dashboard) of the app, where you see the list of workspaces. 

TYPE OF CHANGE	DESCRIPTION
All other changes (new or changed pages or actions, or changes to business logic)	Pull-to-refresh from the workspace that has the edited pages or actions. 

Additional resources

Page design guidelines

Action design guidelines

Form design requirements

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Business logic events

2/18/2021 • 2 minutes to read • Edit Online

Mobile workspace business logic events provide places for developers to specify workspace configuration to enhance capability, and implement business-scenario-specific behaviors. All mobile business logic executes in the process of the mobile app, and business logic execution flow is controlled by the Operations mobile app framework.

Code that executes in business logic can make runtime modifications to the metadata of Pages, Actions, and Controls. These runtime modifications overlayer the static metadata that is cached in the app, but the modifications do not directly change the cached static metadata, nor do they affect the static metadata that is stored on the server. These runtime modifications only persist until the app is closed.

Code that executes in the business logic of the app can make runtime modifications to business data that is stored in the app. These modifications (when performed according to correct practices) participate in the normal data processing framework as other business data in the app. That means adhering to the offline-first capabilities and asynchronous save behaviors provided by the framework.

Page design guidelines

2/18/2021 • 7 minutes to read • Edit Online

Before you begin to use the designer to build pages and actions, it's important that you plan the overall design of the mobile workspace that you want to build. We recommend that you orient your design around the entities that you plan to use in the mobile workspace. Don't begin by thinking about the forms that you want to use. From the perspective of the mobile app, the forms are just a mechanism for retrieving data, and the run-time UI behavior of a form isn't applicable to the mobile app. Therefore, you should first identify your entities and the relationships between them. For each entity, the following questions will help you decide how you should design your forms and pages.

How do I create a list view for an entity in the mobile app?

1. Identify or create a form in the web client that contains a grid for the entity.
2. Make sure that the grid is bound to the table that represents the entity.
3. Make sure that the form has a menu item that is root-navigable.
4. Make sure that the form can be opened directly via a URL that includes the menu item parameter.
5. Make sure that the filter pane enables the grid to be filtered based on the desired fields.
6. In the designer, create a page for the entity.
7. In the designer, put only a list on the page.
8. In the designer, put the desired fields in the list on the page.

What if I don't want a list view for this entity?

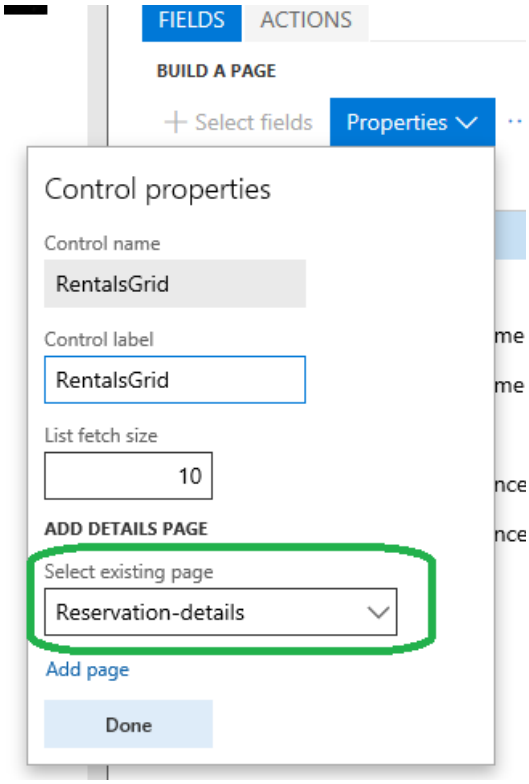
If you want just a details view for an entity, it's likely that the entity is a singleton for a given context (such as a given user or a given company). This pattern applies, for example, to a details view for an employee's own profile in a self-service workspace or a details view for the company context that is used for the current session. See the guidelines for creating a page for a details view.

How do I create a details view for an entity in the mobile app?

1. Identify or create a form in the web client that contains the details view for this entity.
2. Make sure that the Master Root Data Source on the form is bound to the table that represents the entity.
3. Make sure that the form has a menu item that is root-navigable.
4. Make sure that the form can be opened directly via a URL that includes the menu item parameter.
5. In the designer, create a page for the entity.
6. In the designer, put the desired fields on the page.

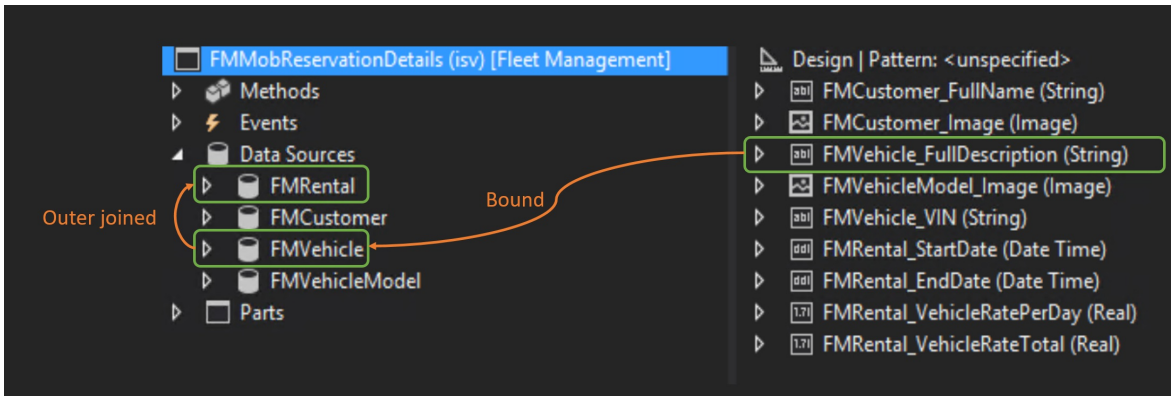
How do I create list-to-details navigation for an entity in the mobile app?

1. Make sure that you've created both a list view page and a details view page for the entity by using the designer.
2. Make sure that the entity for the list view is the same as the entity for the details view. In other words, the table that is bound to the grid on the form that is used for the list view must be the same table that is the Master Root Data Source on the form that is used for the details view.
3. Make sure that the form that is used for the details view can be filtered on a unique key field by using the filter pane.
4. In the designer, make sure that the list view page is linked to the details view page. Click the list, open the properties, and then set the details view page by using the lookup.

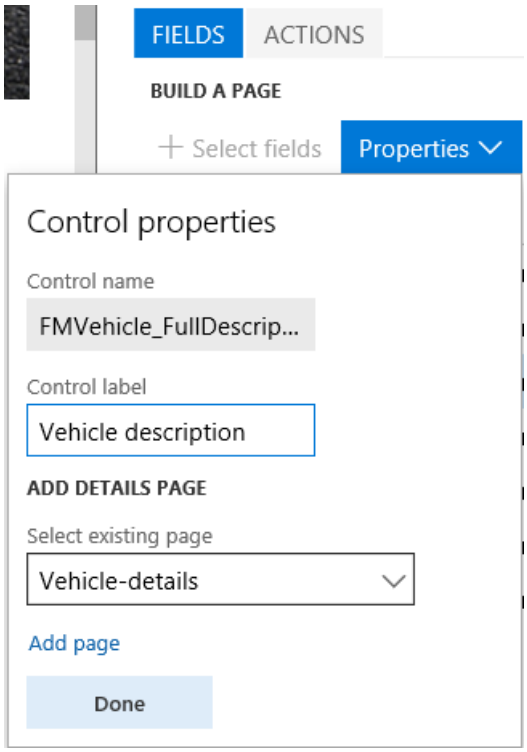


How do I add a reference field that enables navigation to a related entity?

1. Make sure that either a list view page or a details view page exists for the entity that contains the reference.
2. Make sure that the page contains the reference field from the entity that is being referenced.
3. Make sure that the referenced field is bound to the referenced entity's data source, and that the referenced entity is *outer joined* (1-0..1) or *inner joined* (1-1) to the data source for the entity that contains the reference. For example, in the following illustration, FMRental is the entity that contains the reference, and FMVehicle is the referenced entity.



4. Make sure that you've created a separate details view page for the entity that is being referenced.
5. Make sure that the reference field has been added to the page.
6. In the designer, make sure that the reference field has been linked to the details view for the referenced entity. For example, in the following illustration, Vehicle-details is the details view page for the referenced entity.



How do I add a list that contains items from a related entity to a details view page?

How do I make the list show up in-line in the details view?

1. Identify or create a form in the web client that contains the details view for the entity, and make sure that the form adheres to the guidelines in the "How do I create a details view for an entity in the mobile app" section.
2. Make sure that the form contains a grid that is bound to the table that represents the related entity.
3. Make sure that the table for the related entity is *active joined* to the table for the entity that contains the reference.
4. Create a details view page that contains the desired fields for the entity, and that also contains a list that has the desired fields from the related entity.

How do I make the list accessible from a link in the details view (instead of in-line)?

1. Identify or create a form in the web client that contains the details view for the entity, and make sure that the form adheres to the guidelines in the "How do I create a details view for an entity in the mobile app" section.
2. Make sure that the form contains a grid that is bound to the table that represents the related entity.
3. Make sure that the table for the related entity is *active joined* to the table for the entity that contains the reference.
4. Use the form to create a details view page that contains the desired fields for the entity.
5. Use the same form to create a separate list view page that contains only a list that has the desired fields from the related entity.
6. On the details view page, add a PageLinkControl that links to the list view page. Currently, you must use business logic to add the PageLinkControl. The following example show the code that Fleet Management uses.

```
function main(metadataService, dataService, cacheService, $q) {
    return {
        appInit: function (appMetadata) {
            metadataService.addLink(
                'Customer-details', // the Page to add the link to
                'Customer-rentals', // the Page the link goes to
                'cust-rentals-nav-control', // unique name for the control
                'Rentals', // text to display for the link in the UI
                true, // show/hide the count for items on the linked page
                );
        },
    };
}
```

How do I read data from a hidden page?

1. Identify or create a page that contains the controls with the data that you want.
2. Refer to the following code example, which hides the page from the navigation menus, and accesses data on the page using the provided APIs. Note that 'My-Hidden-Page' and 'My-Field-Id' are the names of the page and control, respectively, and can be found when viewing the corresponding page in the designer.

```
function main(metadataService, dataService, cacheService, $q) {
    myField1Value = ''; // This variable will be populated in appInit, and can then be used elsewhere
in the business logic.
    return {
        appInit: function (appMetadata) {
            var myHiddenPage = metadataService.findPage('My-Hidden-Page');
            if(myHiddenPage) {
                var dataPromise = dataService.getPageData(myHiddenPage.Id,'','',0);
                dataPromise.then(function (result) {
                    var myField1Id = metadataService.findControl(myHiddenPage, 'My-Field-1').Id;
                    myField1Value = result.data[myField1Id];
                }
            }
    };
}
```

How do I adjust the number of records returned in a list page using list fetch size?

The number of records returned in a list page is controlled by the **List fetch size** value. The default is 50 records. The **List fetch size** indicates the maximum number of records returned by a page when it first loads, and the maximum number of records returned when search is used to find a specific set of records. Be careful not to make the value too large or it may negatively affect the user experience.

1. In the Mobile App designer,add a page containing a grid and select some fields from the grid.
2. Click the **Grid** node and then click **Properties**.
3. The **Control properties** dialog box will contain a default fetch size of 50 records.
4. Adjust the fetch size as needed.

Action design guidelines

2/18/2021 • 6 minutes to read • Edit Online

Actions let users create, update, or delete data, and also run business processes on that data (such as *submit*, *confirm*, and *post*). A user who completes an action first supplies the data for the action (if the action accepts data input). When the user has finished supplying the data, the action is put into a queue of similar actions (which are sometimes referred to as *data sync operations*). If the device is connected/online, the queue is processed immediately. Otherwise, it's processed the next time that the device is connected. The queue is processed asynchronously and doesn't require the user's attention unless there is an error during data synchronization. Errors of this type can occur because of server-side data validation. Actions are powered by a server-side mechanism that resembles Task recordings. This mechanism extracts the user's input from the action and then automatically runs the business process steps on the server by using the input values that the user supplied. The mechanism automatically opens forms, clicks buttons on the forms, and enters the user's input into controls on the forms. This process of playing back the action against the forms on the server occurs asynchronously, against "headless" forms. The mobile app informs the user when the process is completed, and shows the user any info, warning, or error messages that the forms logged. When you design an action, it's important that you first consider what entity the action is related to. In the current framework, an action must operate on only one entity. An action should not update multiple entities at the same time. For example, an action to create a new sales order should create only the header for the order. It should not also try to create lines, because the lines are separate entities. When you decide to design the action, consider the following questions to determine how to proceed.

How do I design an action that enables an entity to be created?

1. Identify or create a list view page for the entity.
2. Make sure that the form that is used for the list view page includes a **New** button that can be used to add new records to the list.
3. Use the designer to create a new action for the page. While you're designing the action, be careful not to perform any unnecessary actions. Enter data only in those fields that should be available to the user, and click only those buttons that are required (for example the **New** button and the **Save** button).

How do I design an action that enables an entity to be edited?

1. Identify or create a details view page for the entity.
2. Make sure that the form that is used for the details view page includes an **Edit** button that can be used to edit the visible record.
3. Make sure that the form that is used for the details view page lets users open a specific record by applying filters in the filter pane.
4. Use the designer to create a new action for the page. While you're designing the action, be careful not to perform any unnecessary actions. Enter data only in those fields that should be available to the user, and click only those buttons that are required (for example, the **Edit** button and the **Save** button).

How do I design an action that enables an entity to be deleted?

1. Identify or create a details view page for the entity.
2. Make sure that the form that is used for the details view page includes a **Delete** button that can be used to delete the visible record.
3. Make sure that the form that is used for the details view page lets users open a specific record by applying filters in the filter pane.
4. Use the designer to create a new action for the page, and just click **Delete** as a part of the process of designing the action.

How do I design an action that enables a business action to be performed on an entity?

1. Identify or create a details view page for the entity.
2. Make sure that the form that is used for the details view page includes a **Delete** button that can be used to delete the visible record.
3. Make sure that the form that is used for the details view page lets users open a specific record by applying filters in the filter pane.
4. Use the designer to create a new action for the page, and just perform the steps that are required in the business action. You don't necessarily have to enter data in fields as part of the action. For an action such as *submit*, you just have to click the **Submit** button (and acknowledge any confirmations that appear).

How do I design an action that enables a field value to be set via a rich lookup?

Lookups for fields in the mobile app don't have a correlation to the advanced lookup behaviors in the cloud version of the app. Regardless of whether you have a custom lookup in the cloud vesion of the app or an automatic lookup that uses a simple query, the mobile app doesn't run existing lookup code when it must determine which UI to show the user. (Remember that the user might be offline while using the app, and server-side code isn’t run until the action is synchronized.) However, lookup/control overrides such as *modified* are run when the value is set by the mobile back end as it synchronizes the data from the action. When the mobile app detects that a field on an action was selected from a lookup field on a form, it shows a device-native combo box/list picker control, and populates the items by directly querying the backing table of that lookup field. The items in the list show the user data from the TitleField for records in that table. Follow these steps to add the rich lookup experience to your action. This lookup experience includes a full-page multi-column lookup selector that has offline search.

1. Identify or create a list view page for the entity behind the lookup. You can reuse existing list view pages that you've already created.
2. After you've finished designing the action, select the field to add rich lookup functionality to, and then click **Properties**.
3. In the **Control properties** dialog box, select the list view page that you identified or created in step 1, and set the other related properties.

Manage mobile app

Action title

New Reservation

Action description

Create a new reservation

Action name

Add-Reservation

BUILD AN ACTION

+ Select fields | Properties | ...

Control properties

Control name

FMRental_Customer

Control label

CustomerSS

ADD LOOKUP PAGE

Select existing page

All-Customers

Select field to use

Record-ID

Select field to display

Customer

Done

4. Save and publish your changes to the action.

If you don't see the property for the existing list view page or can't access the **Control properties** dialog box when you're designing your action, you might be using an older build of the app. In this case, you can still add rich lookup functionality by using a business logic file.

```
function main(metadataService, dataService, cacheService, $q) {
    return {
        appInit: function (appMetadata) {
            metadataService.configureLookup(
                // specify the name of the Action to add the lookup to
                'Add-Reservation',
                // specify the name of the Action's field to add the lookup to
                'FMRental_Customer',
                {
                    // specify the name of the Page for the Entity for the lookup
                        lookupPage: 'All-Customers',
                    // specify the Page's field which contains the value to set on the lookup
                // this value should be the same value you can type into the field on the Form
                        valueField: 'FMCustomer_RecId',
                        // specify the Page's field which contains the value to display to the user
                        // this value is only used for display. The value field is passed to the Form
                        displayField: 'FMCustomer_FullName',
                        // set this to true to enable the rich lookup
                        showLookupPage: true
                }
            );
        },
    };
}
```

How do I prevent an action from appearing in the list of actions for a page?

To prevent an action from appearing in the list of actions for any page, call the following code from the **appInit()** section of your business logic. In this code, **action-name** is the name of your action (as specified in the **Action name** field in the designer).

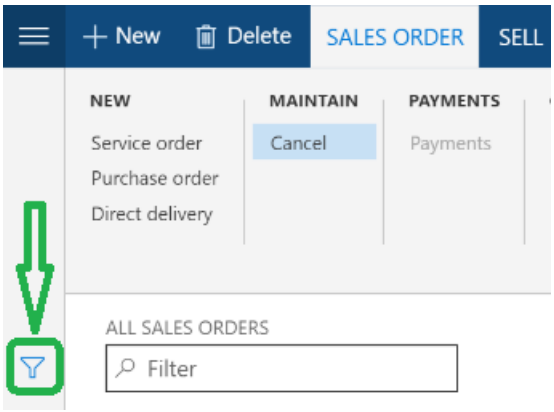
```
function main(metadataService, dataService, cacheService, $q) {
    return {
        appInit: function (appMetadata) {
            metadataService.configureAction('action-name', { visible: false });
        },
    };
}
```

Form design requirements

2/18/2021 • 6 minutes to read • Edit Online

This section provides valuable guidelines for building forms that work well with the mobile app.

- Each form must have an associated Display Menu Item.
  - The Display Menu Item's **Allow Root Navigation** property must be set to **Yes**. This setting enables the mobile framework to open the form that is referenced by the menu item.
- Each form must be directly accessible via its Display Menu Item.
  - To verify accessibility, open the menu item via a URL. Just append **&mi=** to the URL, where the value is the Application Object Tree (AOT) name of your menu item.
  - If the form doesn't open or show data when you access it in this way, the form won't work with the mobile app.
- Each form that shows data must have one Master Root Data Source.
  - This data source must be the first data source on the form (top-most in the designer).
  - This data source must not be joined to any other data sources.
- Each form must work with the data source filters.
  - After you open the form in the web client, open the filter pane by using the **Show filters** button.



Then click **Add a filter field**, and verify that the Master Root Data Source appears as the table for fields in the list of available fields. Other tables can also appear, but the Master Root Data Source **must** appear in this list. Otherwise, the mobile app won't enable searches and navigation that uses context.

  - Searching: The mobile app does online searches against data by using the Filters framework behind the scenes.
  - Navigation that uses context: The mobile app enables list-to-details navigation (and other context-aware navigation) by first opening the target form via the menu item and then using the Filters framework to show only the specified record context.
  - List-to-details navigation: The table that the grid is bound to on form A (the list form) must be the Master Root Data Source on the details form (form B). When a user selects a record in the list on form A, the mobile framework navigates with record context by applying filters on form B that uniquely identify the record.

Design considerations

Using data methods

You can use display methods to show data on pages (both list type pages and detail type pages). However, there

are two key points to remember when you use display methods:

- **Searching** – When a user performs an "online" search (that is, a search that is run against data in the web client instead of locally cached data), the search won't match against display methods, because the Filtering framework in the web client doesn't support searches against data methods. However, when a user does a search against locally cached data, the search will match against display methods, provided that the records have been cached on the device.
- **Offline** – If a user creates or updates data while their device isn't connected to the server, temporary records are created in the local cache. Because these temporary records haven't yet been processed in the web client, if the records have any fields that are automatically populated or defaults by server-side business logic, these fields will remain empty until the records have been synced with the web client. Display methods fall into this category of fields that will be empty for a temporary record.

Designing for offline

Unlike the web client, which is highly connected to the server and maintains an open user session that has open forms on the server, the mobile app creates user sessions (and opens forms) only in short bursts while the app is being synced with the server (via data read for pages, or via data write/update for actions). If there are no actions to sync with the server, and if the local data cache is up to date, the mobile app won't communicate with the server as a user navigates around the app (unless the user triggers an explicit pull-to-refresh). It's important that you keep this data flow pattern in mind while you design pages and actions in the mobile app. You should not expect form logic to run every time that a page is loaded or an action is started. You should also never expect form logic to run while a user is completing an action. Form logic is run only when the action is being synced with the server. The following list describes the only times when you should expect Form logic to run.

Form logic runs right before a page is opened on the mobile app for the first time.

1. When a user first opens a page, the mobile app reaches out to the web client and opens the associated forms. *During this process, logic such as form init and data source init is all run in the usual manner.*
2. The mobile app framework reads the required data directly from the controls on the forms and sends the data back to the mobile app.
3. The mobile app caches the data and shows it in the page on the mobile app.
4. Future attempts to open the page will load the cached data. *These attempts won't run the form logic again, unless the user explicitly refreshes the page or the cache expires. (Currently, the cache set to expire after 30 minutes.)*

Processing an action that has been submitted to the server from the mobile app

1. When a user opens an action and fills in the data in that action, *no form logic is run*. A user can complete an action either offline or online. The system behaves the same way in both cases.
2. After the user clicks **Done/Save** on the action, the mobile app queues a data synchronization operation. This operation will be synced with the server when the mobile app is connected to the Internet.
3. When an Internet connection is detected (which can happen immediately after the action is completed) the mobile app sends the data synchronization operation to the server for processing.
4. While the operation is processed on the server, the framework opens the associated forms and enters the data from the action by passing values into the form controls. *During this process, form logic is run in the usual manner (init, modified, clicked, and so on, are all run).* However, the mobile user might have moved to a different part of the app while this processing is occurring. *Any form logic that shows/hides controls will have no effect on the UI that is seen in the mobile app.* Therefore, to minimize synchronization times, it's best not to include any UI logic on the form.

Building mobile versions of existing forms

If you decide to modify existing forms so that they work with the mobile framework, instead of building new mobile-specific forms, you might have to conditionally change the form's behavior for mobile-specific scenarios. You can use the following static X++ application programming interfaces (APIs) in your X++ code to determine whether the code is being accessed during a session where a web client user is designing pages/actions or

during a session that the mobile framework back end created to load pages/actions for a mobile user. **When a form is being used with the mobile designer**

```
SysTaskRecorderController::isDesigningApp()
```

When a form is being used by the mobile framework back end to load pages and run actions

```
SysTaskRecorderController::isExecutingApp()
```

Form control support

The form controls for the various base data types (strings, dates, and numbers) and grids are supported. However, a few common controls have limited support. **Reference groups** Fields from within Reference groups controls are compatible when you design pages. However, they aren't compatible when you design Actions. Although you might be able to select these fields without experience any issue, Reference groups have a fundamental incompatibility with the mobile framework. We recommend that you not use Reference groups. Instead, add a control directly to the form, and then bind the control directly to the surrogate foreign key (SFK) by using the property sheet.

Configure workspaces by using the SysAppWorkspace class

2/18/2021 • 4 minutes to read • Edit Online

Workspace class, **SysAppWorkspace**, is the starting point to create, configure and publish workspaces on the server. The following categories of APIs are available for use in sysAppWorkspace

- **Workspace attributes** - This is used to create pages, tasks, entities, lookups, relationships in order to build mobile workspaces.
    - Download the sample project for Fleet Management Mobile App. This is an .axpp file found at Dynamics365-for-Operations-mobile-FleetManagementSamples.
    - After downloading the file, open Visual Studio on your Operations development environment, select **Dynamics 365 > Import Project**, and browse for the downloaded project file. On the same dialog box, select **Overwrite** and select **Create a new solution**. After the import is complete, build the solution (or build the Fleet Management model).
    - To review the example, start by reviewing the FMReservationManagementWorkspace class to see all the pages and actions included in the workspace. Use Solution Explorer to find page and task classes, and all the assets included in each. Use the API reference for more details on each API.
    - A mobile workspace can be created through designer pane, using X++ attribute APIs or a combination of both. See the "Use the workspace class to publish workspaces from AOT resources" section below for more details about how to import mobile app metadata from designer to AOT. The sample project Fleet Management Mobile App is a complete mobile app built using X++ attribute APIs.
- **Workspace metadata classes** - This is used to inspect and apply server-side business logic to metadata for mobile workspaces.

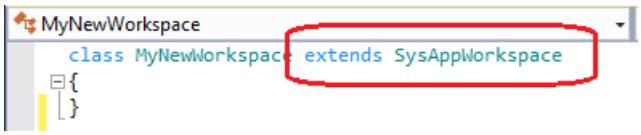
For a complete list of server-side APIs, see Server-side development (workspace X++ APIs).

Create a new workspace class

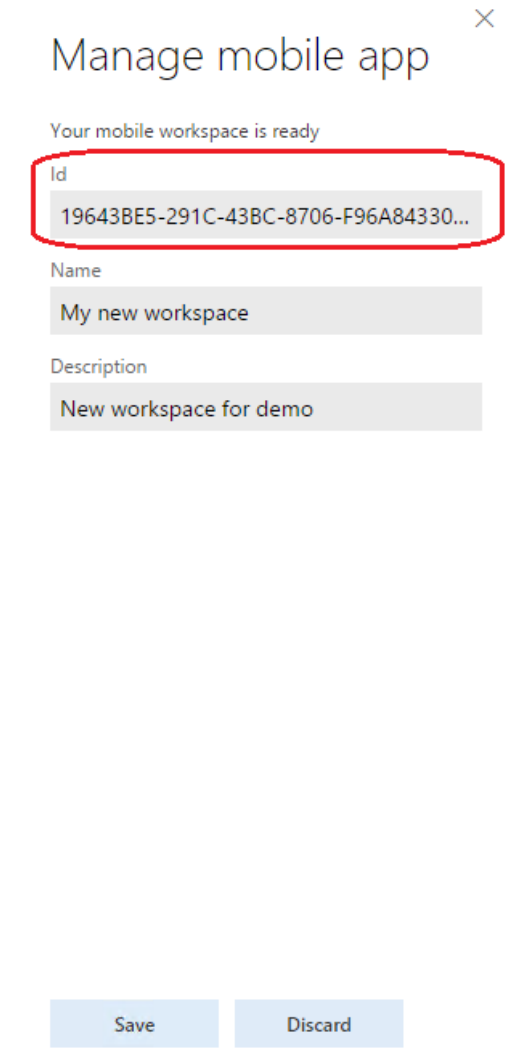
To use the **SysAppWorkspace** class for your workspace, you must create a new class for the workspace by extending the **SysAppWorkspace** class. You can then use the new class to modify workspace metadata. The new class also provides hooks for life cycle management of the mobile app.

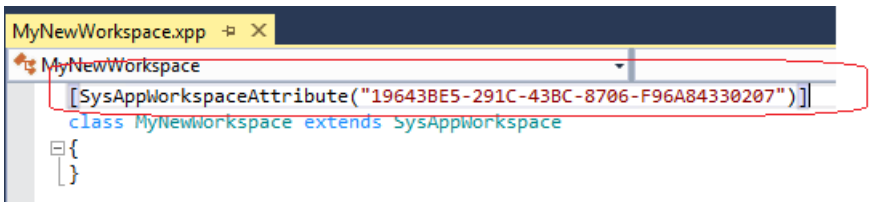
Follow these steps to create a new workspace class for your workspace.

1. Create a new class for your workspace, and extend it from the **SysAppWorkspace** class.

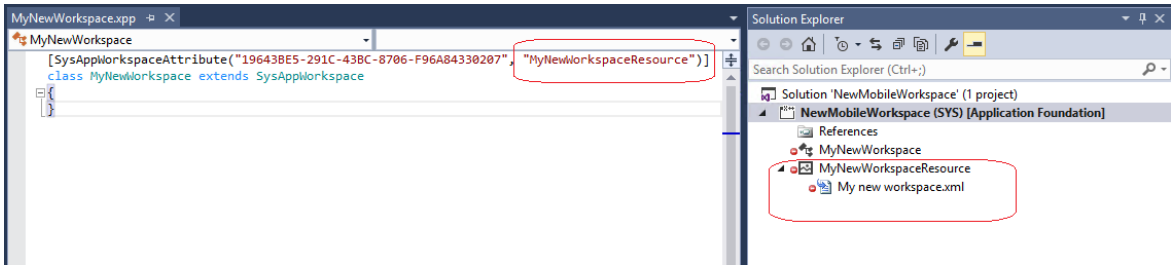


2. Add the **SysAppWorkspaceAttribute** attribute to the new class, and provide the **AppID** value of your workspace. You can find the app ID for your app on the **Summary** page in the mobile app designer.





3. Optional: If your workspace is an Application Object Tree (AOT) resource, provide the AOT resource name as the second parameter for the **SysAppWorkspaceAttribute** constructor.

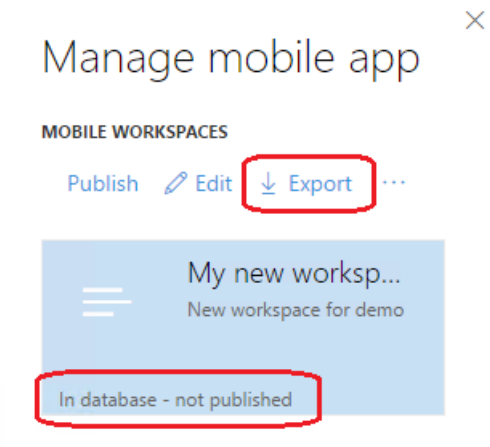


Use the workspace class to publish workspaces from AOT resources

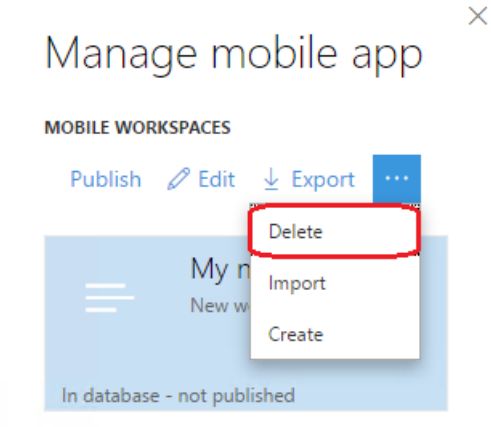
Workspaces can reside in the database. They can also reside in the AOT as resources. To provide visibility into workspaces that are stored in AOT resources, you must create a workspace class and point it to the name of the AOT resource that contains the workspace. Workspaces that are stored as AOT resources can't be edited or deleted by using the mobile app designer. Those workspace can only be exported.

Follow these steps to publish a workspace that resides in an AOT resource.

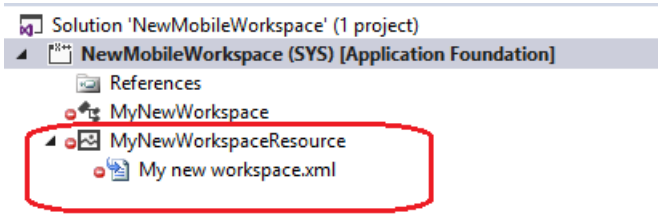
1. When you're developing a workspace that is stored in the database, you must export it from the mobile app designer so that it can be stored as an AOT resource. The workspace is exported as an XML file.



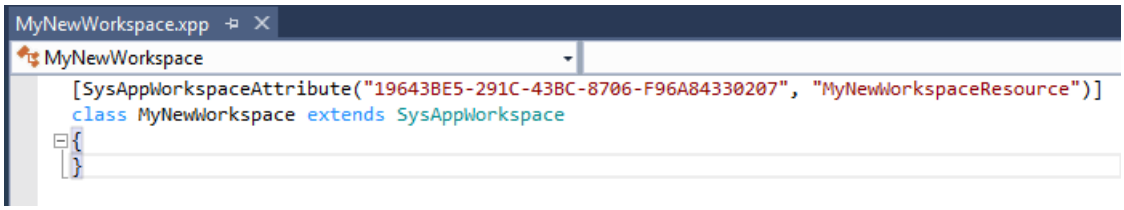
2. Delete the workspace from the mobile app designer. You will load it from an AOT resource later.



3. Create a new AOT resource, and select the exported workspace for the resource.

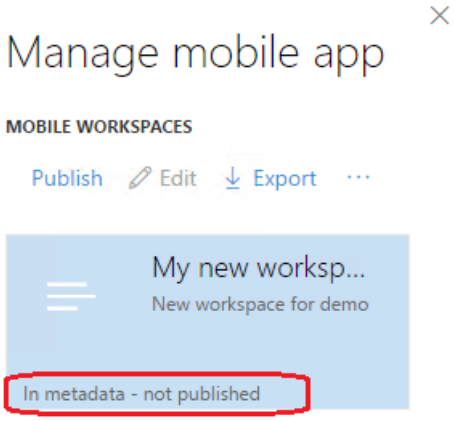


4. Create a new class for your workspace. This class should extend **SysAppWorkspace**. Apply the **SysAppWorkspaceAttribute** attribute to the class, and provide the app ID and the AOT resource name that contains the resource.



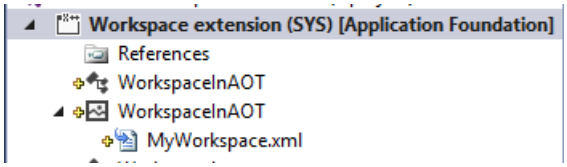
5. Build the class, and reopen the mobile app designer.

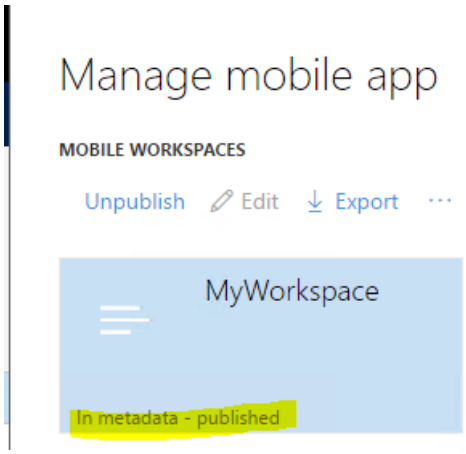
   The workspace is now published. It appears in the designer, but can't be edited or deleted. Note that the workspace is loaded from metadata.



Update a workspace that has already been published

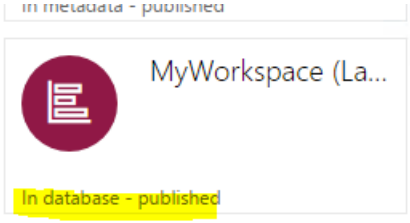
If your workspace is part of an AOT resource, you can't edit it by using the mobile app designer. In the following example, a workspace that is named **MyWorkspace** exists in the AOT, and it has a backing class that is named **WorkspaceInAOT**.

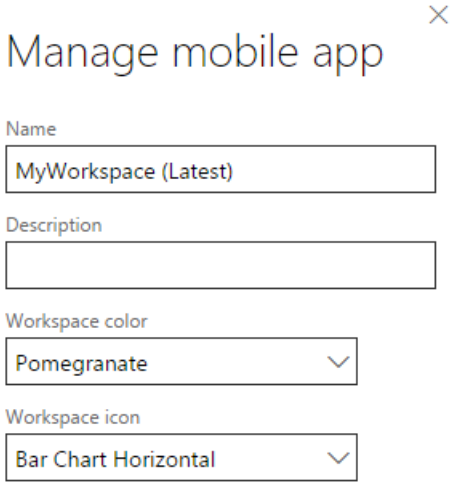




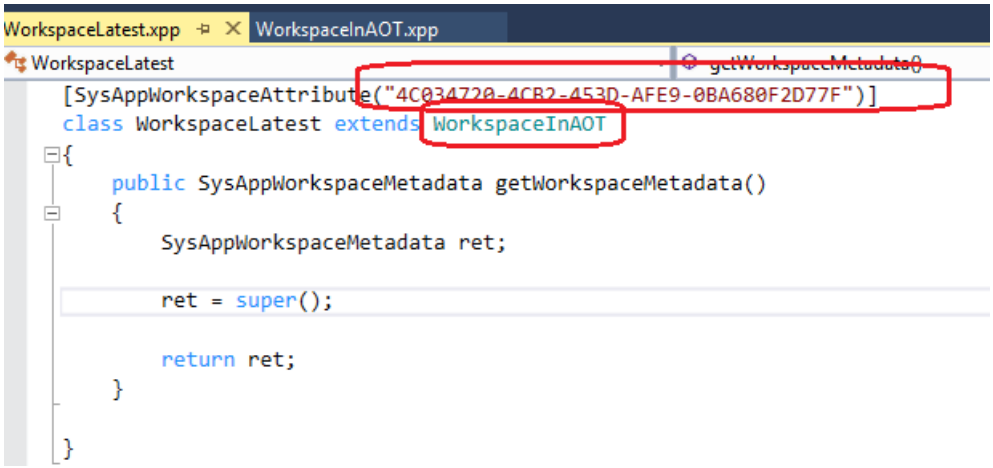
Follow these steps to edit the workspace.

1. Export the workspace by using the mobile app designer. The designer automatically creates new app IDs for workspaces that are stored in the AOT.
2. Import the newly exported workspace by using the mobile app designer.
    a. Optional: Change the name so that the newly added workspace can be distinguished from other workspaces.
    b. Copy the app ID of the newly created workspace.





3. Create a new class that extends your backing class, apply the **SysAppWorkspaceAttribute** attribute, and specify the new app ID.

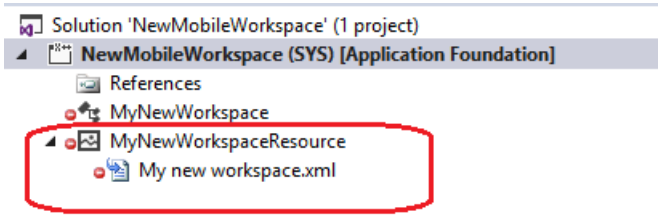


You can now continue to work with your new workspace and the backing class. After you've finished making your changes, you can merge them with the AOT-based workspace.

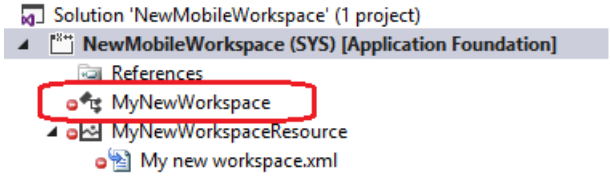
Delete a workspace that is an AOT resource

When a mobile workspace is stored as an AOT resource, you can't delete it by using the mobile app designer. Follow these steps to delete a workspace that exists as an AOT resource.

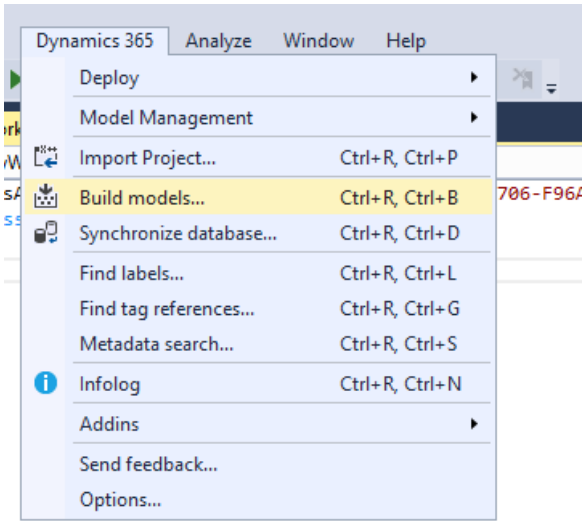
1. Delete the AOT resource that contains the workspace.

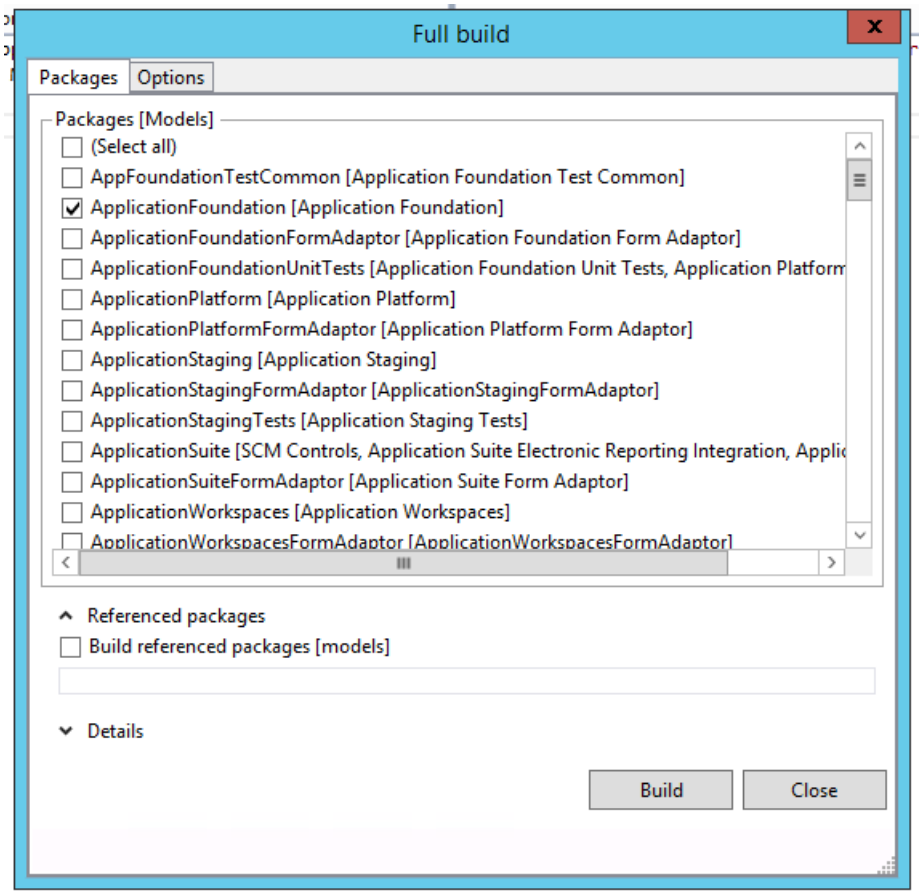


2. Delete the workspace class that was created for the workspace.



3. Do a full model build that contains the AOT resource and the class. The following illustrations shows a full build of the Application Foundation model. The Application Foundation model also contains the AOT resource and workspace class. To speed up the full build, you can clear all the selections on the **Options** tab.





4. When the build is completed, reopen the mobile app designer, and verify that the workspace is no longer there.

Make fields on mobile app pages clickable

2/18/2021 • 2 minutes to read • Edit Online

The fields on a mobile app page can be customized so that they are shown as email addresses, phone numbers, or URLs.

Email field

You can mark a field as an email address field by using business logic. Then, when a user clicks the field, the default mobile email app starts, and the field value appears as the email address in the app.

Phone field

You can mark a field as a phone number field by using business logic. Then, when a user clicks the field, the mobile dialer app starts, and the field value appears as the phone number in the app.

> **NOTE**
> On iOS, if the phone number isn't valid, it might not trigger the mobile dialer app.

URL field

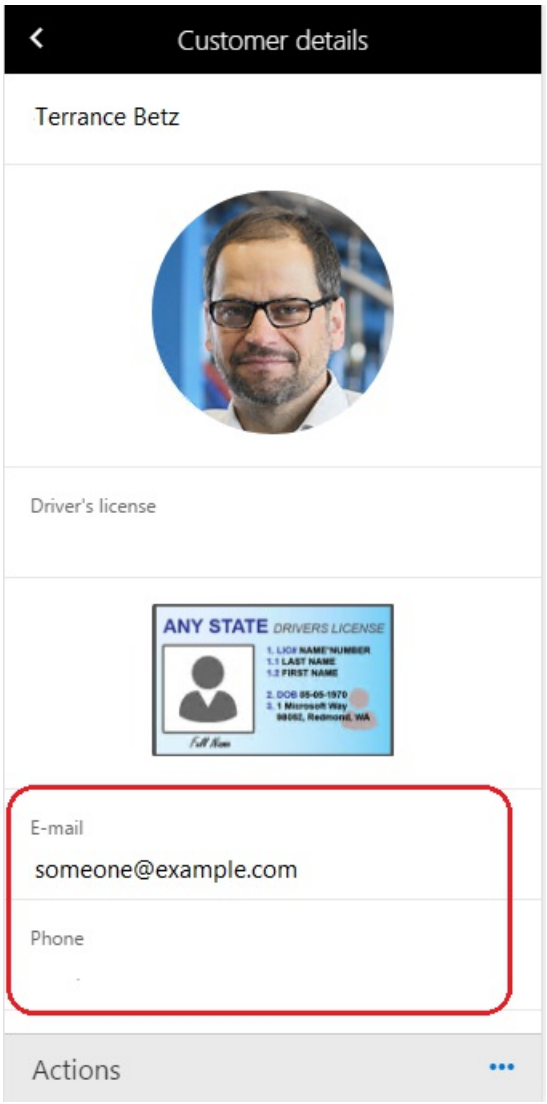
You can mark a field as a URL field by using business logic. Then, when a user clicks the field, the URL opens in the default mobile browser, and the field value appears in the address bar.

> **NOTE**
> On iOS, you must provide a complete URL (that is, a URL that starts with a protocol, such as **https**). Otherwise, the URL isn't opened in the browser. A URL such as `www.microsoft.com` doesn't work. Instead, the URL must be specified as `https://www.microsoft.com`.

Example

This example shows how to configure the customer email address and phone number fields so that they can clicked and opened in the appropriate iOS apps.

Before the fields are customized, they can't be clicked, as shown in the following image.



Follow these steps to specify that a field is a link.

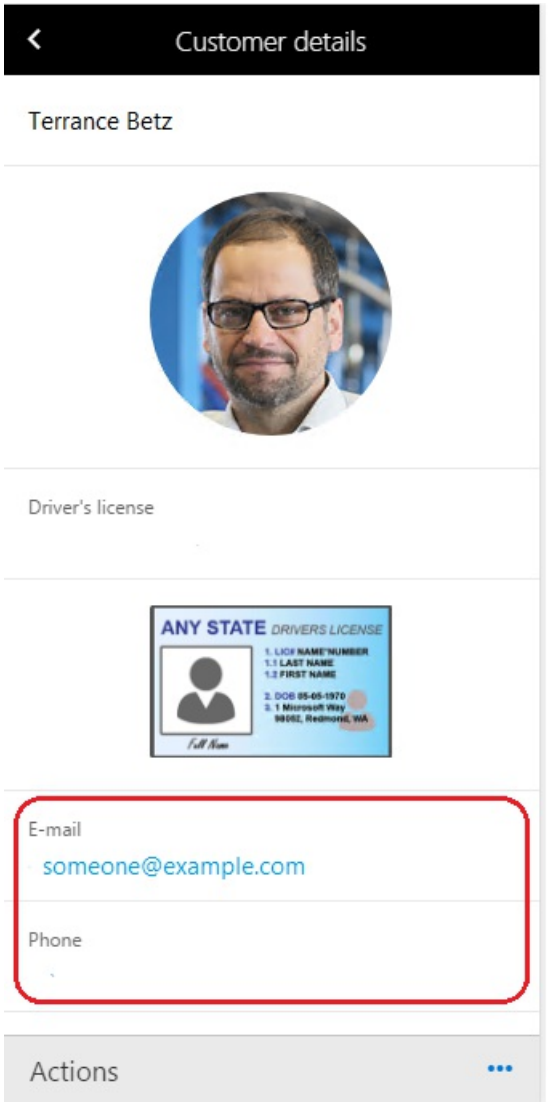
1. Add the following lines to the **appInit** method. You call the **configureControl** method, and pass in the page name and control name. You then supply the **LinkType** value for the control. The following values are supported: **Telephone**, **Email**, and **Url**.

   ```
   metadataService.configureControl('PageName', 'ControlName', { LinkType: 'Telephone' });
   metadataService.configureControl('PageName', ' ControlName ', { LinkType: 'Email' });
   metadataService.configureControl('PageName', ' ControlName ', { LinkType: 'Url' });
   ```

2. Upload the updated business logic file by using the mobile app designer.

3. Update the workspace metadata in the mobile client.

The fields now appear as links.



NOTE

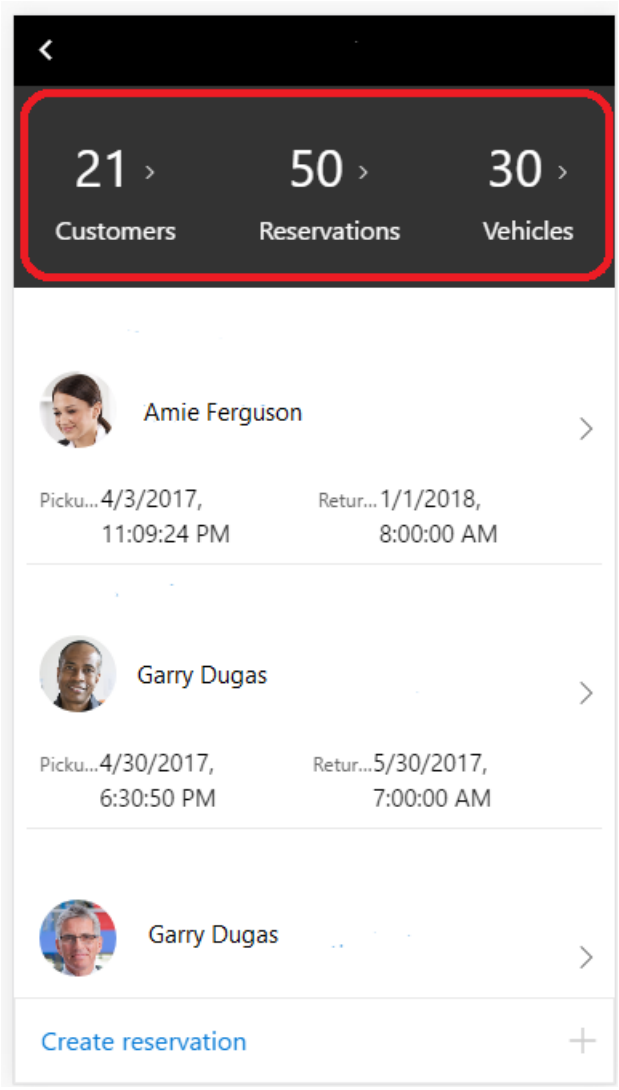
Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Show counts in fields

2/18/2021 • 2 minutes to read • Edit Online

Although a **pageLink** control can be used to show counts (totals), it can be slow, because it must load the target page before it counts the number of rows. Additionally, the count that is calculated can be incorrect, because there is a limit on the number of rows that are retrieved.

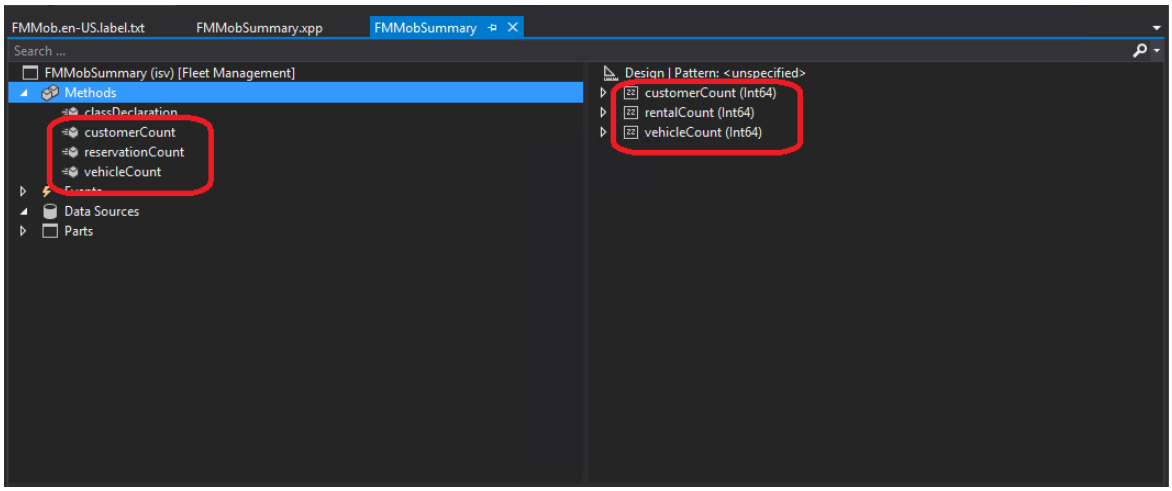


If you want to make mobile workspaces work more quickly, we recommended that you use a regular field to show the count and then model the field as a **pageLink** control in the mobile client.

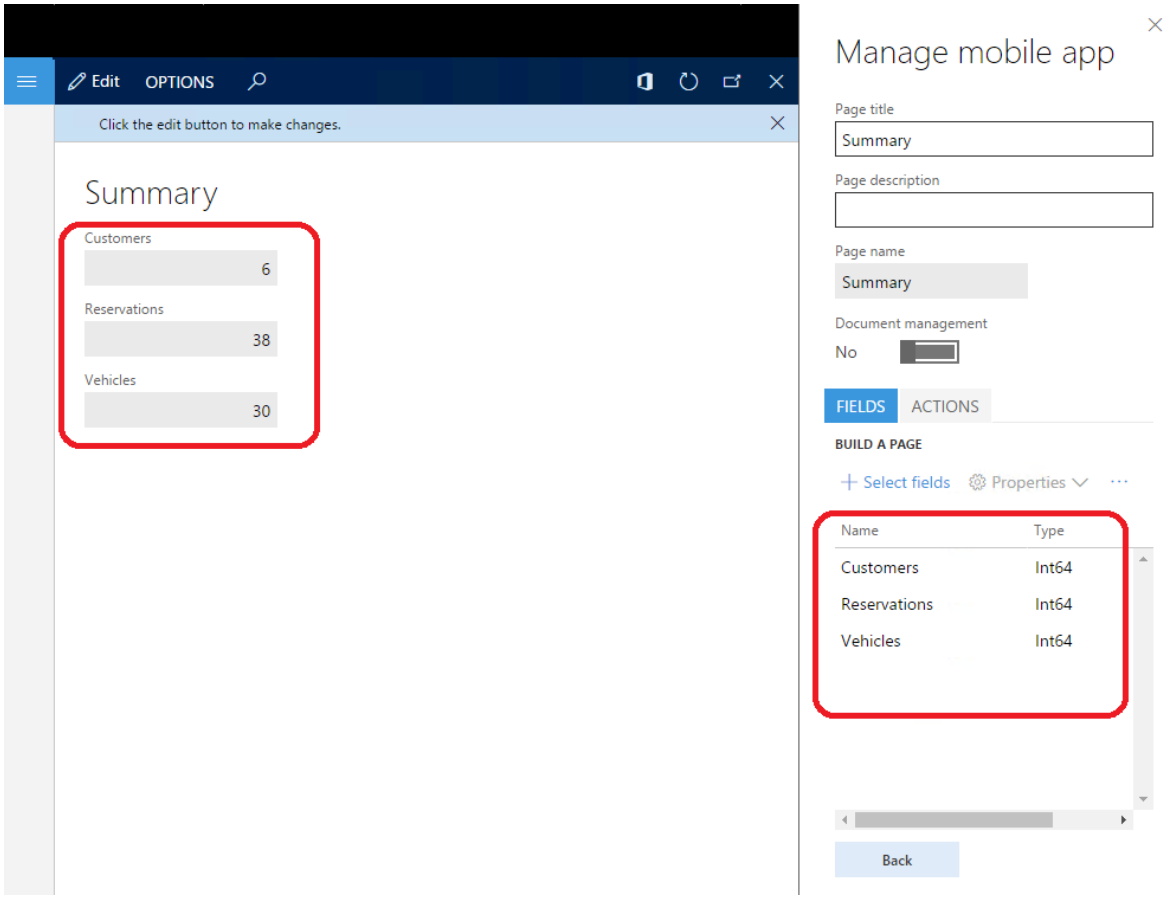
The following example uses the Fleet Management app. In the Fleet Management app, the workspace shows the total number of customers, reservations, and vehicles. Previously, these counts came from a **pageLink** control that had AllCustomers, AllReservations, and AllVehicles as targets. The **pageLink** control loaded the rows and did the count. (This approach isn't the recommended approach.)

Follow these steps to configure the workspace page to use the recommended approach.

1. On the server, create a new form to contain fields that are also on the server. (You can also add the new fields to an existing form). In the following illustration, a new **FMMobSummary** form is created that has three fields.



2. Create a page by using the mobile app designer for the **FMMobSummary** form.



3. Update the business logic to transform the fields into a **pageLink** control. Use the **configureControl** method to add a navigation target to the fields. The fields are then configured as **pageLink** controls. The arguments for the **configureControl** method are the page name, the control name, and an object of properties that must be updated.

4. Update the workspace design. Embed the summary page as a part in the workspace page. Reference the fields that are now configured as **pageLink** controls. Provide a style, and set the **showCount:true** property, so that the count is shown on the **pageLink** control.

```
{
    "type": "Part",
    target: "Summary",
    "flexFlow": "row nowrap",
    "background": "dark",
    "color": "light",
    design:
    {
        "flexFlow": "row nowrap",
        "fontSize": "large",
        "border": "none",
        "justifyItems": "space-around",
        "items":
        [
            {
                name:"customerCount",
                style: "button",
                showCount: true,
            },
            {
                name:"rentalCount",
                style: "button",
                showCount: true,
            },
            {
                name:"vehicleCount",
                style: "button",
                showCount: true,
            },

        ]
    }

},
// lower half - grow or shrink
```

By using this approach, you also get the localized labels for **pageLink** controls. The result is a much faster experience when workspaces are loaded.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Help secure mobile workspaces

2/18/2021 • 2 minutes to read • Edit Online

This topic describes how to limit a user's access to a workspace.

Assign a menu item to workspace

Workspaces can be tied to a menu item. Users who don't have access to the menu item can't use the workspace, because the workspace is shown only to users who have rights to the menu item.

If a menu item isn't assigned to a workspace, the workspace is always shown to the user.

Follow these steps to help secure your workspaces by assigning a menu item.

1. Add a **SysAppWorkspaceSecurityAttribute** attribute to the workspace class, and specify the menu item to assign to the workspace.

```
[
    SysAppWorkspaceAttribute("4C034720-4CB2-453D-AFE9-0BA680F2D77F"),
    SysAppWorkspaceSecurityAttribute("BatchGroup")
]
```

2. Build the menu item and workspace. To test your changes, sign in to mobile app by using a user account that does'nt have access to the menu item.

Override the workspaceHidden method

You can also specify whether the workspace is hidden or shown, based on parameters. By overriding the **workspaceHidden** method, you enable your code to control the visibility of the workspace, as shown in the following code example.

```
/// <summary>
/// Used to control if a workspace is hidden or not
/// </summary>
/// <returns>Returns true if the workspace is hidden; otherwise false</returns>
public boolean workspaceHidden()
{
    boolean ret;

    ret = super();

    return ret;
}
```

Add a menu item and override the workspaceHidden method

You can use both the preceding methods in your app. The menu item provides a security check, and the **workspaceHidden** method contains additional logic that is related to the visibility of the workspace.

Localize mobile workspaces

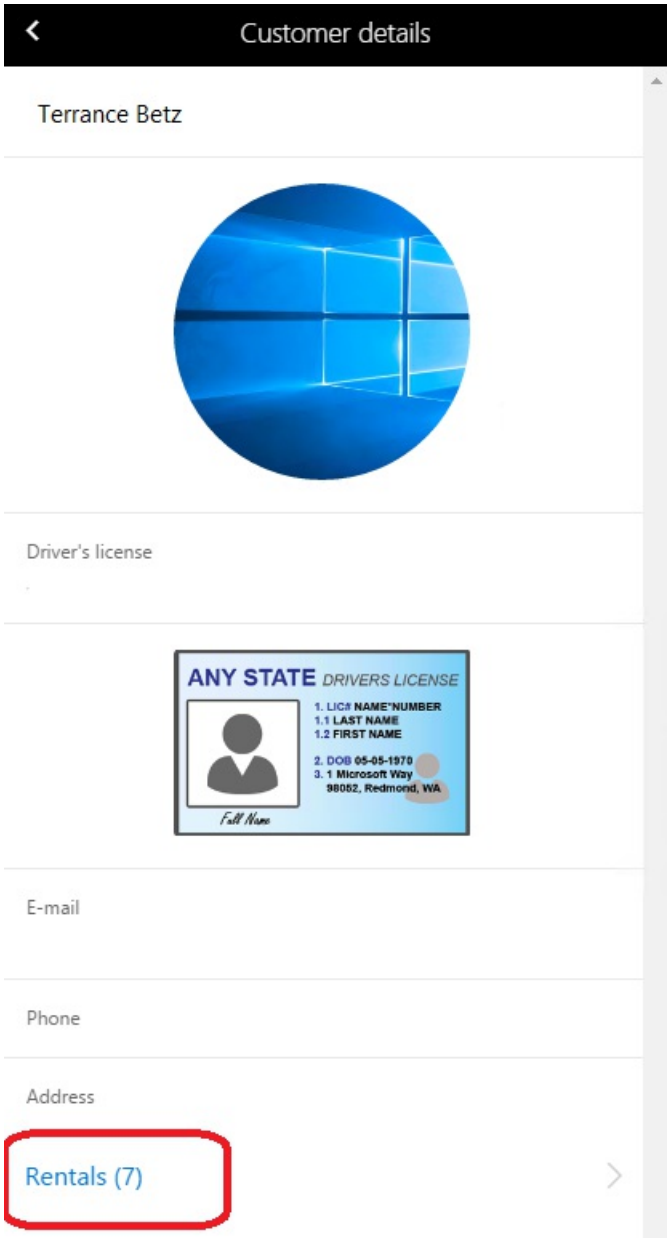
2/18/2021 • 2 minutes to read • Edit Online

You can use workspace classes in several ways to provide localization support to workspaces.

Use config objects to pass localized labels

A config object can be added to the workspace metadata when it's requested by the mobile app. Later, the config object can be used to provide localization support.

For example, the following workspace requires localized labels for the **pageLink** control that you added via the business logic.



The business logic for the app contains a call to the **addLink** method, as shown in the following illustration. This **addLink** method adds a link to the **Rentals** page for the current customer. In this case, the label for the link is **Rentals**. However, because there isn't a localized label for the link, the link always appears as **Rentals**.

```
metadataService.addLink(pageNames.CustomerDetails, pageNames.CustomerRentals, controlNames.CustomerRentals, 'Rentals', true);
```

To use a config object to provide localized labels, follow these steps.

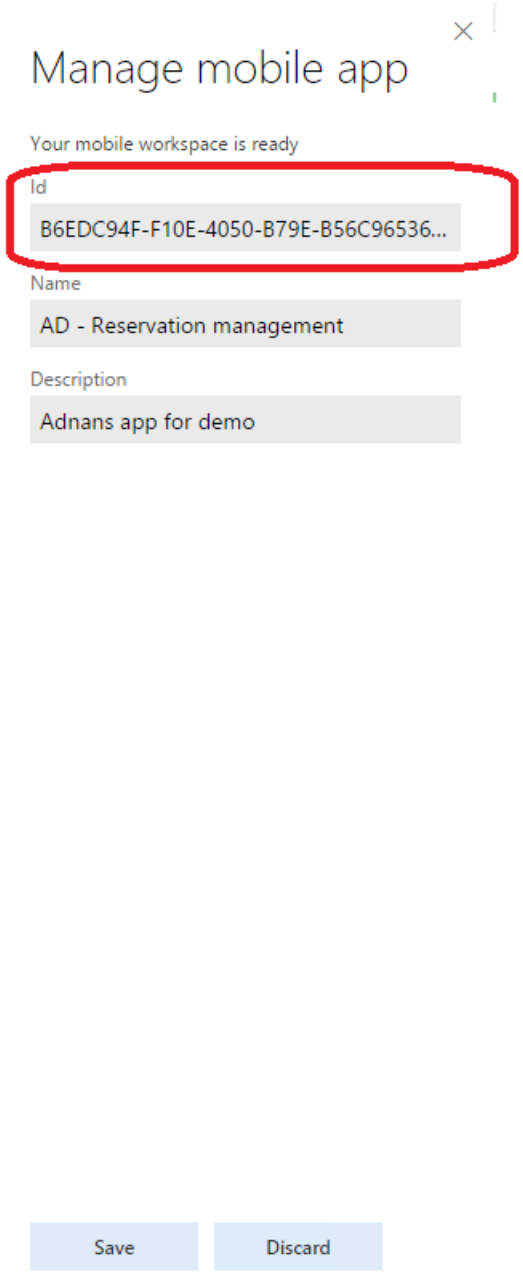
1. Create a config class that contains the fields for the labels. One field, **rentalLinkLabel**, is added to the class that will contain the label for the **pageLink** control. The config class must be a data contract class.

```
[DataContractAttribute('ReservationManagementLabels')]
class ReservationManagementWorkspaceLabels
{
    str rentalLinkLabel;

    [DataMemberAttribute('RentalLinkLabel')]
    public str rentalLinkLabel(str _rentalLinkLabel = rentalLinkLabel)
    {
        rentalLinkLabel = _rentalLinkLabel;
        return rentalLinkLabel;
    }

}
```

2. The config class is used by a workspace class for the workspace. The workspace class requires the **appId** value of the workspace. You can find the app ID in the App designer, as shown in the following illustration.



The following illustration shows what the workspace class looks like when the **appId** value is set on the attribute. The class also contains a method, **addConfig**, that sets a config object that contains the value for the label.

```
[SysAppWorkspaceAttribute("B6EDC94F-F10E-4050-B79E-B56C965368AD")]
class ReservationManagementWorkspace extends SysAppWorkspace
{
    public SysAppWorkspaceMetadata getWorkspaceMetadata()
    {
        SysAppWorkspaceMetadata workspace;

        workspace = super();

        this.addConfig(workspace);

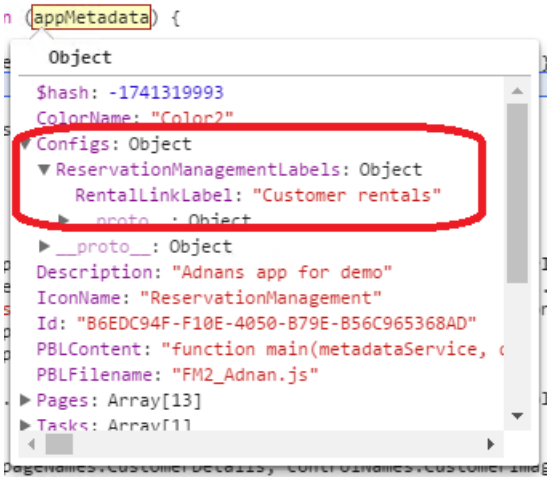
        return workspace;
    }

    private void addConfig(SysAppWorkspaceMetadata _workspace)
    {
        ReservationManagementWorkspaceLabels labelContract = new ReservationManagementWorkspaceLabels();
        labelContract.rentalLinkLabel("@FMMob:FMMob_CustomerRentals");

        _workspace.addConfig("ReservationManagementLabels", labelContract);
    }

}
```

The following illustration shows the config object in the **appInit** call in the mobile app.

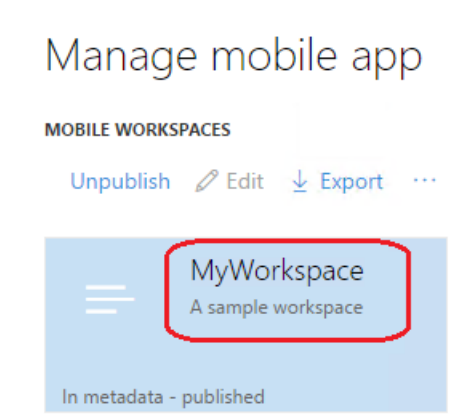


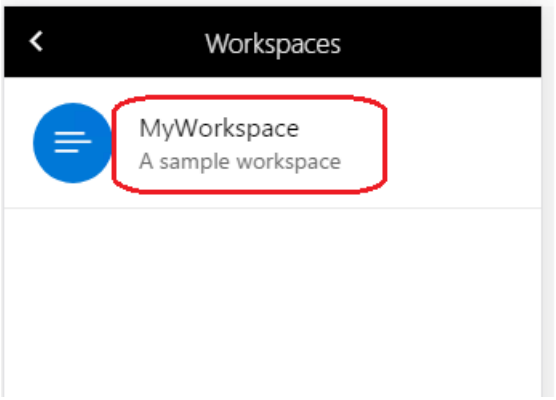
3. The config object can now be used and passed to the **addLink** method instead of the hard-coded label.

```
metadataService.addLink(pageNames.CustomerDetails,
    pageNames.CustomerRentals,
    controlNames.CustomerRentals,
    appMetadata.Configs.ReservationManagementLabels.RentalLinkLabel,
    true);
```

Use a workspace class to update the workspace title and description

A workspace class can be used to provide localized strings for the workspace title and description. If you don't localize the title and description, the fields will be in the language that you implemented them in. In this example, we will localize a workspace where **MyWorkspace** is the title and **A sample workspace** is the description.





1. If you don't have a workspace class for your workspace, create a workspace class.
2. Override the **getWorkspaceMetadata** method to get the workspace metadata. You must have the workspace metadata to provide labels for the workspace title and description fields.
3. Use the **workspaceTitle** and **workspaceDescription** properties to set the workspace title and description from a label. In the following illustration, placeholders are assigned to the **workspaceTitle** and **workspaceDescription** properties.

```
[SysAppWorkspaceAttribute("16DF07A7-B822-4D36-9D61-B388837C8140", "WorkspaceInAOT")]
class WorkspaceInAOT extends SysAppWorkspace
{
    public SysAppWorkspaceMetadata getWorkspaceMetadata()
    {
        SysAppWorkspaceMetadata workspaceMetadata;

        workspaceMetadata = super();

        this.updateWorkspacetitleAndDescription(workspaceMetadata);

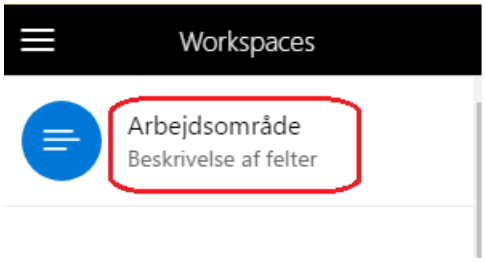
        return workspaceMetadata;
    }

    private void updateWorkspacetitleAndDescription(SysAppWorkspaceMetadata _workspaceMetadata)
    {
        _workspaceMetadata.workspaceTitle("@Client:Workspace");
        _workspaceMetadata.workspaceDescription("@ApplicationFoundation:FieldDescriptionsCaption");
    }

}
```

4. Build the workspace class.
5. Update the app list on the mobile client.

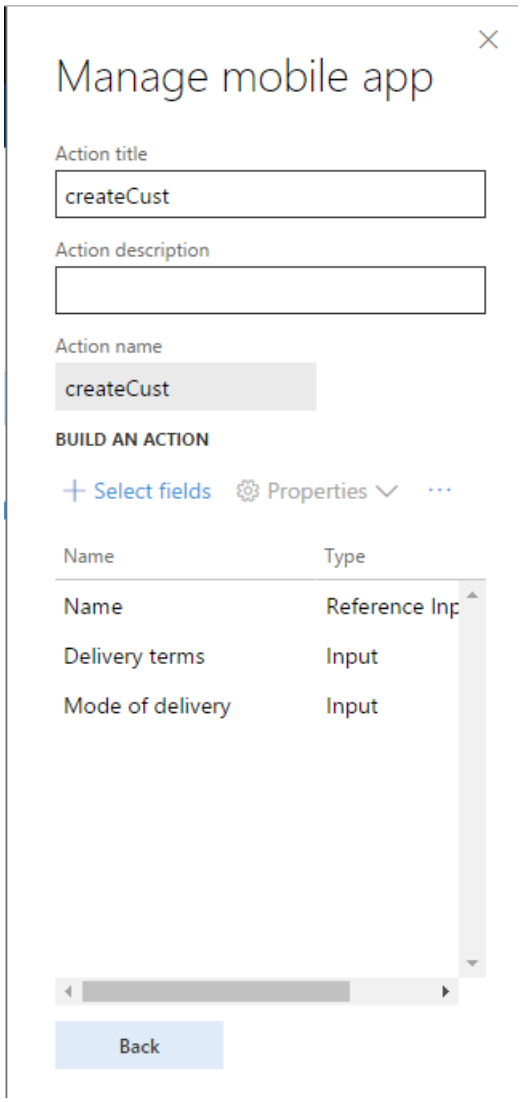
The following illustration shows the title and description on a phone that uses English and Danish.

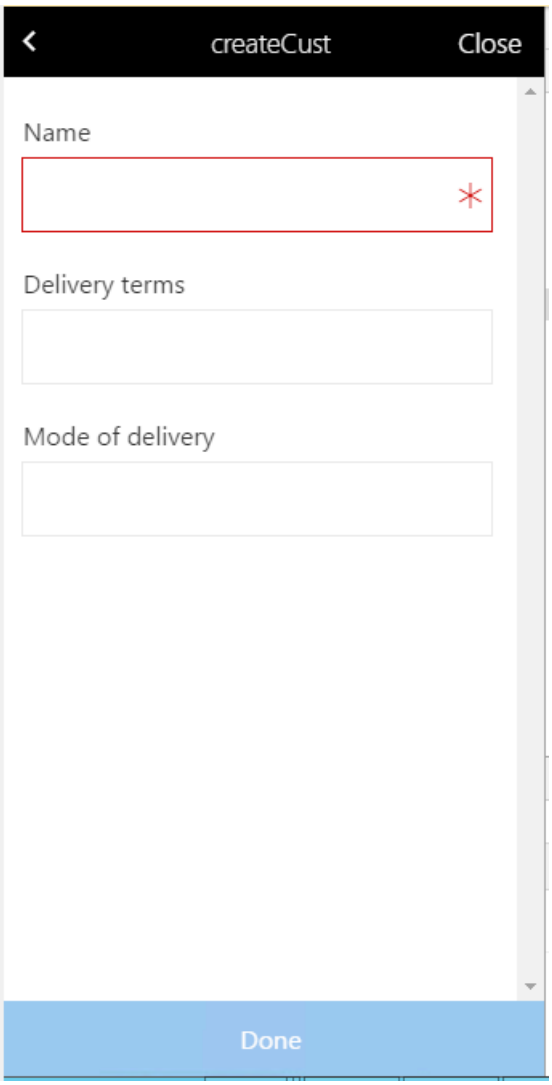


Make fields mandatory by using workspace classes

2/18/2021 • 2 minutes to read • Edit Online

When you use the mobile app designer to select fields for actions, some properties can be inferred. These properties include the field length, the type, and whether the field is mandatory. The workspace classes can be used to update these properties. For example, you might want to specify that the **Name** field is mandatory when a customer record is created, as shown in the following images.





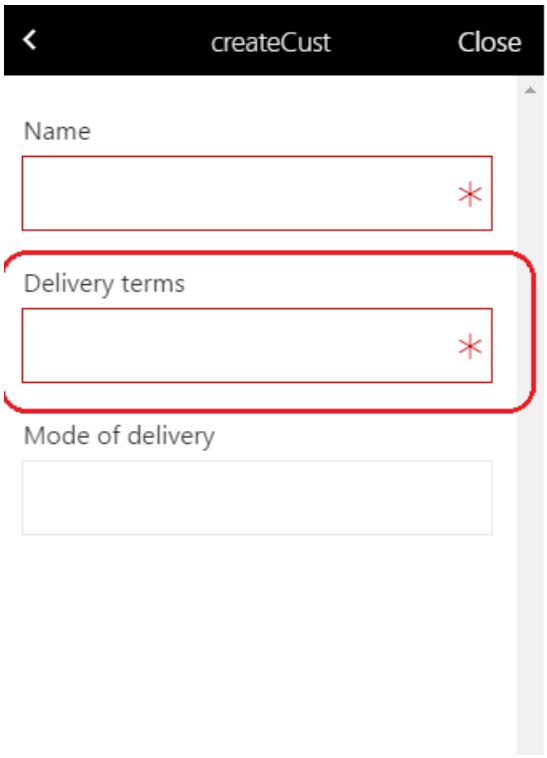
Follow these steps to make the **Delivery terms** field mandatory by using the workspace class.

1. Get the control name by using the app designer. In this example, the control name is **DynamicDetail_DlvTerm**.
2. Add the following code to set the **Mandatory** property for the control. This code uses the reflection-based **setProperty** method to set the **Mandatory** property.

```
public SysAppWorkspaceMetadata getWorkspaceMetadata()
{
    SysAppWorkspaceMetadata appMetadata;
    appMetadata = super();
    var createCustAction = appMetadata.getAction("createCust");
    this.setCustAccountMandatory(createCustAction);
    return appMetadata;
}
private void setCustAccountMandatory(SysAPpActionMetadata _createCustAction)
{
    var custAccount = +createCustAction.getControl("DynamicDetail_DlvTerm");
    custAccount.setProperty("Mandatory", true);
}
```

3. Build the solution, and then update the app metadata on the mobile app.

The **Delivery terms** field is now marked as **Mandatory**, as shown in the following illustration.



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Server-side development (workspace X++ APIs)

2/18/2021 • 23 minutes to read • Edit Online

Class SysAppActionAttribute

SysAppActionAttribute used for decorating methods defining actions of workspace

Methods

METHOD NAME	RETURNS	DESCRIPTION
new	void	Creates a new instance of SysAppActionAttribute class
pageMethodName	str	Gets the get method name that forms the page under which this task resides
actionTitle	str	Gets the Action Title
actionDescription	str	Gets the Action Description
crudOperationType	SysAppCRUDOperation	Gets the Crud Operation Type like Create, Update, Delete

Method new

Creates a new instance of SysAppActionAttribute class

```
public void new ([str _actionTitle], [str _actionDescription], [SysAppCRUDOperation _crudOperationType],
[str _pageMethodName])
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_actionTitle	str	True	Action title
_actionDescription	str	True	Action description
_crudOperationType	SysAppCRUDOperation	True	CRUD operation like Create, Update, Delete
_pageMethodName	str	True	Name of the method constructing parent page

Method pageMethodName

Gets the get method name that forms the page under which this task resides

```
public str pageMethodName ()
```

Return Value

The page method name that forms the page under which this task resides

Method actionTitle

Gets the Action Title

```
public str actionTitle ()
```

Return Value

The action title

Method actionDescription

Gets the Action Description

```
public str actionDescription ()
```

Return Value

The page description

Method crudOperationType

Gets the Crud Operation Type like Create, Update, Delete

```
public SysAppCRUDOperation crudOperationType ()
```

Return Value

The Crud Operation Type like Create, Update, Delete

Class SysAppActionMetadata

This class can be used to access and update AX mobile workspace action metadata

Methods

METHOD NAME	RETURNS	DESCRIPTION
new	void	
getActionName	str	Returns the action name
actionTitle	str	Gets or sets the action title
actionDescription	str	Gets or sets the action description
actionHidden	boolean	Gets or sets whether action is hidden
actionOrder	int	Gets or sets the action order
getControl	SysAppControlMetadata	Returns the control on the current action having the provided control name
getControlEnumerator	MapEnumerator	Returns a map enumerator that can be used to enumerate action controls. Where Key is control name and value is of type SysAppControlMetadata

Method new

```
public void new (Microsoft.Dynamics.Client.ServerForm.App.TaskMetadata _taskmetadata)
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_taskmetadata	Microsoft.Dynamics.Client.ServerForm.App.TaskMetadata	False	

Method getActionName

Returns the action name

```
public str getActionName ()
```

Return Value

The action name

Method actionTitle

Gets or sets the action title

```
public str actionTitle ([str _actionTitle])
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_actionTitle	str	True	The action title

Return Value

The action title

Method actionDescription

Gets or sets the action description

```
public str actionDescription ([str _actionDescription])
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_actionDescription	str	True	The action description

Return Value

The action description

Method actionHidden

Gets or sets whether action is hidden

```
public boolean actionHidden ([boolean _actionHidden])
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_actionHidden	boolean	True	Action hidden value

Return Value

True if the action is hidden; otherwise false

Method actionOrder

Gets or sets the action order

```
public int actionOrder ([int _actionOrder])
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_actionOrder	int	True	The action order

Return Value

The action order

Method getControl

Returns the control on the current action having the provided control name

```
public SysAppControlMetadata getControl (str _controlName)
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_controlName	str	False	The control name that will be used to search for control

Return Value

An object of SysAppControlMetadata is returned if a control with the provided control name exist on the action; otherwise null

Method getControlEnumerator

Returns a map enumerator that can be used to enumerate action controls. Where Key is control name and value is of type SysAppControlMetadata

```
public MapEnumerator getControlEnumerator ()
```

Return Value

A map enumerator

Class SysAppAttributeHelper

SysAppAttributeHelper class for fetching attributes from all the extended class

Methods

METHOD NAME	RETURNS	DESCRIPTION
getAttributeFromClass	SysAttribute	gets attribute from class

Method getAttributeFromClass

gets attribute from class

```
public SysAttribute getAttributeFromClass (SysDictClass _sysClass, SysAppAttributeType _attributeType)
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_sysClass	SysDictClass	False	class for which attributes is required
_attributeType	SysAppAttributeType	False	Type of attribute like SysAppEntityAttribute

Class SysAppCollectionAttribute

SysAppCollectionAttribute used for decorating methods forming list control

Methods

METHOD NAME	RETURNS	DESCRIPTION
new	void	Constructor

Method new

Constructor

```
public void new (str _itemContractName, [str _label], [str _relationshipName])
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_itemContractName	str	False	Data contract name of list item
_label	str	True	List control label
_relationshipName	str	True	Relationship name. By default the entity name of the list item is used as relationship name

Class SysAppControlMetadata

Represents an X++ wrapper over the managed ControlMetadata object to facilitate. passing around the object as X++ object

Methods

METHOD NAME	RETURNS	DESCRIPTION
new	void	Creates a new instance of the control metadata
getBaseLanguageId	str	Returns the base language ID for the app
getControlName	str	Returns the control name
controlLabel	str	Gets and sets the control label
controlHidden	boolean	Gets and sets whether the control is hidden
controlOrder	int	Gets or sets the control order
controlMandatory	boolean	Gets or sets the control mandatory
controlAllowNegative	boolean	Gets or sets the control allow negative
controlMaxLength	int	Gets or sets the control max length
getProperty	`anytype`	Gets the control property referenced by the key
setProperty	void	Sets the control property referenced by the key

Method new

Creates a new instance of the control metadata

```
public void new (Microsoft.Dynamics.Client.ServerForm.App.ControlMetadata _controlMetadata, [str
_baseLanguageId])
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_controlMetadata	Microsoft.Dynamics.Client.ServerForm.App.ControlMetadata	False	The controlMetadata object
_baseLanguageId	str	True	The base language

Method getBaseLanguageId

Returns the base language ID for the app

```
public str getBaseLanguageId ()
```

Return Value

The base language ID

Method getControlName

Returns the control name

```
public str getControlName ()
```

Return Value

The control name

Method controlLabel

Gets and sets the control label

```
public str controlLabel ([str _controlLabel])
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_controlLabel	str	True	The control label

Return Value

The control label

Method controlHidden

Gets and sets whether the control is hidden

```
public boolean controlHidden ([boolean _controlHidden])
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_controlHidden	boolean	True	Control hidden value

Return Value

True if the control is hidden; otherwise false

Method controlOrder

Gets or sets the control order

```
public int controlOrder ([int _controlOrder])
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_controlOrder	int	True	The control order

Return Value

The control order

Method controlMandatory

Gets or sets the control mandatory

```
public boolean controlMandatory ([boolean _controlMandatory])
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_controlMandatory	boolean	True	The control mandatory

Return Value

The control mandatory

Method controlAllowNegative

Gets or sets the control allow negative

```
public boolean controlAllowNegative ([boolean _controlAllowNegative])
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_controlAllowNegative	boolean	True	The control allows negative

Return Value

The control allows negative

Method controlMaxLength

Gets or sets the control max length

```
public int controlMaxLength ([int _controlMaxLength])
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_controlMaxLength	int	True	The control max length

Return Value

The control max length

Method getProperty

Gets the control property referenced by the key

```
public anytype getProperty (str _key)
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_key	str	False	The name of the control property

Return Value

The property value

Method setProperty

Sets the control property referenced by the key

```
public void setProperty (str _key, anytype _value)
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_key	str	False	The name of the control property
_value	anytype	False	The value of the control property

Class SysAppEntityAttribute

SysAppEntityAttribute used for decorating data contract entities

Methods

METHOD NAME	RETURNS	DESCRIPTION
new	void	Constructor
name	str	Gets the name of the entity
entityKey	str	Gets the entity key

Method new

Constructor

```
public void new (str _name, str _entityKey)
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_name	str	False	Entity name
_entityKey	str	False	Name of the entity's key

Method name

Gets the name of the entity

```
public str name ()
```

Return Value

Name of the entity

Method entityKey

Gets the entity key

```
public str entityKey ()
```

Return Value

Entity key

Class SysAppEntityContext

SysAppEntityContext used for defining entity context

Methods

METHOD NAME	RETURNS	DESCRIPTION
constructFromParams	SysAppEntityContext	Constructs SysAppEntityContext from entityName and entityId
constructFromBuffer	SysAppEntityContext	Constructs SysAppEntityContext from table buffer
entityName	str	Entity name on which filter applies
entityId	str	Field value on which filter applies

Method constructFromParams

Constructs SysAppEntityContext from entityName and entityId

```
public SysAppEntityContext constructFromParams (str _entityName, str _entityId)
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_entityName	str	False	Entity name
_entityId	str	False	Entity value

Return Value

Instance of SysAppEntityContext

Method constructFromBuffer

Constructs SysAppEntityContext from table buffer

```
public SysAppEntityContext constructFromBuffer (Common _tableBuffer)
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_tableBuffer	Common	False	table buffer forming the entity

Return Value

Instance of SysAppEntityContext

Method entityName

Entity name on which filter applies

```
public str entityName ([str _entityName])
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_entityName	str	True	Entity name

Return Value

Entity name

Method entityId

Field value on which filter applies

```
public str entityId ([str _entityId])
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_entityId	str	True	Entity value

Return Value

Entity value

Class SysAppFieldAttribute

SysAppFieldAttribute used for decorating methods forming bound fields

Methods

METHOD NAME	RETURNS	DESCRIPTION
new	void	Creates a new instance of SysAppFieldAttribute class

Method new

Creates a new instance of SysAppFieldAttribute class

```
public void new (str _fieldName, str _label)
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_fieldName	str	False	Entity field name with which control is bound
_label	str	False	Control label

Class SysAppFieldMultiSelectHelper

A helper class to provide helper methods for multi select scenarios used with D365 mobile app

Methods

METHOD NAME	RETURNS	DESCRIPTION
new	void	Creates a new instance of SysAppFieldMultiSelectHelper class
getSelectedRecIds	container	Returns a container of recIds of the records that were selected
getSelectedValues	container	Returns a container of selected values
getSelectedRecords	Common	Returns a buffer that contains all the selected records.. The buffer is marked as temp.. Later a while-Select can be used to iterate through all the records
setControlValue	void	A setter to set the multiselect control value

Method new

Creates a new instance of SysAppFieldMultiSelectHelper class

```
public void new (TableId _multiSelectTableId, FieldId _valueFieldId, FormStringControl _multiSelectControl)
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_multiSelectTableId	TableId	False	The backing tableId for the field
_valueFieldId	FieldId	False	The fieldId for the field that will be shown in the multi select control
_multiSelectControl	FormStringControl	False	The string control that will be the multi select control

Method getSelectedRecIds

Returns a container of recIds of the records that were selected

```
public container getSelectedRecIds ()
```

Return Value

A container of recOds for the records that were selected

Method getSelectedValues

Returns a container of selected values

```
public container getSelectedValues ()
```

Return Value

A container of selected values

Method getSelectedRecords

Returns a buffer that contains all the selected records.. The buffer is marked as temp.. Later a while-Select can be used to iterate through all the records

```
public Common getSelectedRecords ()
```

Return Value

A buffer containing all the selected records

Method setControlValue

A setter to set the multi select control value

```
public void setControlValue (str _value)
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_value	str	False	A colon separated value that will be used by SysAppFieldMultiSelectHelper

Class SysAppFilterContext

SysAppFilterContext class that holds context values

Methods

METHOD NAME	RETURNS	DESCRIPTION
entityName	str	Entity name on which filter applies
filterFieldName	str	Field name on which filter applies
filterFieldValueList	List	Gets the list of filter field values based on which filter happens
operator	str	Operator based on which result will be fetched
addFilterFieldValue	void	Adds filter field value

Method entityName

Entity name on which filter applies

```
public str entityName ([str _entityName])
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_entityName	str	True	Entity name on which filter applies

Return Value

Entity name on which filter applies

Method filterFieldName

Field name on which filter applies

```
public str filterFieldName ([str _filterFieldName])
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_filterFieldName	str	True	Field name on which filter applies

Return Value

Field name on which filter applies

Method filterFieldValueList

Gets the list of filter field values based on which filter happens

```
public List filterFieldValueList ()
```

Return Value

List of filter field values based on which filter happens

Method operator

Operator based on which result will be fetched

```
public str operator ([str _operator])
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_operator	str	True	Operator based on which result will be fetched

Return Value

Operator based on which result will be fetched

Method addFilterFieldValue

Adds filter field value

```
public void addFilterFieldValue ( _filterFieldValueList)
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_filterFieldValueList		False	Filter field values based on which filter happens

Class SysAppLookUpAttribute

SysAppPageAttribute used for decorating pages that are also lookup pages

Methods

METHOD NAME	RETURNS	DESCRIPTION
new	void	Creates a new instance of SysAppLookUpAttribute class
displayFieldName	str	Gets the display field name of lookup control
valueFieldName	str	Gets the value field name of lookup control

Method new

Creates a new instance of SysAppLookUpAttribute class

```
public void new (str _displayFieldName, str _valueFieldName)
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_displayFieldName	str	False	Lookup display field. Name of any control of lookup page
_valueFieldName	str	False	Lookup value field. Name of control formed by root data contract constructing lookup page

Method displayFieldName

Gets the display field name of lookup control

```
public str displayFieldName ()
```

Return Value

The display field name of lookup control

Method valueFieldName

Gets the value field name of lookup control

```
public str valueFieldName ()
```

Return Value

The value field name of lookup control

Class SysAppLookupFieldAttribute

SysAppLookupFieldAttribute used for decorating look up fields of action

Methods

METHOD NAME	RETURNS	DESCRIPTION
new	void	Creates a new instance of SysAppLookupFieldAttribute class
entityName	str	Gets the name of the entity with which lookup page is related

Method new

Creates a new instance of SysAppLookupFieldAttribute class

```
public void new ( _name)
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_name		False	Name of the entity with which lookup page is related

Method entityName

Gets the name of the entity with which lookup page is related

```
public str entityName ()
```

Return Value

Name of the entity

Class SysAppPageAttribute

SysAppPageAttribute used for decorating methods defining page of workspace

Methods

METHOD NAME	RETURNS	DESCRIPTION
new	void	Creates a new instance of SysAppPageAttribute class
pageTitle	str	Gets the Page Title of the page
pageDescription	str	Gets the Page Description

Method new

Creates a new instance of SysAppPageAttribute class

```
public void new ([str _pageTitle], [str _pageDescription])
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_pageTitle	str	True	The page title
_pageDescription	str	True	The page description

Method pageTitle

Gets the Page Title of the page

```
public str pageTitle ()
```

Return Value

The page title

Method pageDescription

Gets the Page Description

```
public str pageDescription ()
```

Return Value

The page description

Class SysAppPageMetadata

This class can be used to access and update AX mobile workspace page metadata

Methods

METHOD NAME	RETURNS	DESCRIPTION
new	void	
getPageName	str	Returns the page name
pageTitle	str	Gets and sets the page title
pageDescription	str	Gets or sets the page description
pageHidden	boolean	Gets and sets whether the page is hidden in the workspace
pageOrder	int	Gets or sets the page order
getControl	SysAppControlMetadata	Returns the control on the current page having the provided control name

METHOD NAME	RETURNS	DESCRIPTION
getControlEnumerator	MapEnumerator	Returns a map enumerator that can be used to enumerate page controls. Where Key is control name and value is of type SysAppControlMetadata

Method new

```
public void new (Microsoft.Dynamics.Client.ServerForm.App.PageMetadata _pageMetadata)
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_pageMetadata	Microsoft.Dynamics.Client.ServerForm.App.PageMetadata	False	

Method getPageName

Returns the page name

```
public str getPageName ()
```

Return Value

The page name

Method pageTitle

Gets and sets the page title

```
public str pageTitle ([str _pageTitle])
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_pageTitle	str	True	The page title

Return Value

The page title

Method pageDescription

Gets or sets the page description

```
public str pageDescription ([str _pageDescription])
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_pageDescription	str	True	The page description

Return Value

The page description>

Method pageHidden

Gets and sets whether the page is hidden in the workspace

```
public boolean pageHidden ([boolean _pageHidden])
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_pageHidden	boolean	True	page hidden value

Return Value

True if the current page is hidden in workspace; otherwise false

Method pageOrder

Gets or sets the page order

```
public int pageOrder ([int _pageOrder])
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_pageOrder	int	True	The page order

Return Value

The page order

Method getControl

Returns the control on the current page having the provided control name

```
public SysAppControlMetadata getControl (str _controlName)
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_controlName	str	False	The control name that will be used to search for control

Return Value

An object of SysAppControlMetadata is returned if a control with the provided control name exist on the page; otherwise null

Method getControlEnumerator

Returns a map enumerator that can be used to enumerate page controls. Where Key is control name and value is of type SysAppControlMetadata

```
public MapEnumerator getControlEnumerator ()
```

Return Value

A map enumerator

Class SysAppProjectionAttribute

SysAppProjectionAttribute used for decorating methods forming unbound fields

Methods

METHOD NAME	RETURNS	DESCRIPTION
new	void	Creates a new instance of SysAppControlMetadataAttributes class

Method new

Creates a new instance of SysAppControlMetadataAttributes class

```
public void new (str _label)
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_label	str	False	Control label

Class SysAppRelationalAttribute

SysAppRelationalAttribute used for decorating reference controls

Methods

METHOD NAME	RETURNS	DESCRIPTION
new	void	Constructor

Method new

Constructor

```
public void new ([str _name])
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_name	str	True	Property name of the referenced entity

Class SysAppRequestParams

Request class for X++ methods generating details and action pages

Methods

METHOD NAME	RETURNS	DESCRIPTION
entityContext	SysAppEntityContext	Entity context of the request

METHOD NAME	RETURNS	DESCRIPTION
filterContext	List	List of SysAppFilterContext for filter contexts

Method entityContext

Entity context of the request

```
public SysAppEntityContext entityContext ([SysAppEntityContext _entityContext])
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_entityContext	SysAppEntityContext	True	Entity context of the request

Return Value

Entity context of the request

Method filterContext

List of SysAppFilterContext for filter contexts

```
public List filterContext ([List _filterContext])
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_filterContext	List	True	List of SysAppFilterContext for filter contexts of page

Return Value

List of SysAppFilterContext for filter contexts of page

Class SysAppResponse

SysAppResponse class. This class holds the response object for generated pages and actions

Methods

METHOD NAME	RETURNS	DESCRIPTION
new	void	
jobId	str	Job ID of the request
data	Microsoft.Dynamics.Client.ServerForm.App.CompositeData	Data of the page
failedInAppCall	boolean	Data of the page
commits	List	Commits after task is completed

METHOD NAME	RETURNS	DESCRIPTION
messages	List	Job ID of the request
addMessage	void	Adds message
addCommit	void	Adds commits

Method new

```
public void new ()
```

Method jobId

Job ID of the request

```
public str jobId ()
```

Return Value

jobid of the request

Method data

Data of the page

```
public Microsoft.Dynamics.Client.ServerForm.App.CompositeData data
([Microsoft.Dynamics.Client.ServerForm.App.CompositeData _data])
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_data	Microsoft.Dynamics.Client.ServerForm.App.CompositeData	True	Data of the page

Return Value

Data of the page

Method failedInAppCall

Data of the page

```
public boolean failedInAppCall ([boolean _failedInAppCall])
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_failedInAppCall	boolean	True	Sets to true if it fails in calling application code

Return Value

True when fails in calling application code

Method commits

Commits after task is completed

```
public List commits ()
```

Return Value

Commits after task is completed

Method messages

Job ID of the request

```
public List messages ()
```

Return Value

Messages after task is completed

Method addMessage

Adds message

```
public void addMessage (SysAppResponseMessage _message)
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_message	SysAppResponseMessage	False	message as SysAppResponseMessage object

Method addCommit

Adds commits

```
public void addCommit (SysAppEntityContext _entityContext)
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_entityContext	SysAppEntityContext	False	Entity context containing entity name and entity ID of the entity that is committed

Class SysAppResponseMessage

SysAppResponseMessage class for response messages

Methods

METHOD NAME	RETURNS	DESCRIPTION
new	void	Creates a new instance of SysAppResponseMessage class

METHOD NAME	RETURNS	DESCRIPTION
text	str	Gets the message text
type	SysAppMessageType	Gets the message type: info, error, warning

Method new

Creates a new instance of SysAppResponseMessage class

```
public void new (str _text, [SysAppMessageType _type])
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_text	str	False	The message text
_type	SysAppMessageType	True	Message Type: info, error, warning

Method text

Gets the message text

```
public str text ()
```

Return Value

The message text

Method type

Gets the message type: info, error, warning

```
public SysAppMessageType type ()
```

Return Value

The message type: info, error, warning

Class SysAppSecurityAttribute

SysAppSecurityAttribute used for decorating methods forming pages and actions. specifies security attribute of page or action

Methods

METHOD NAME	RETURNS	DESCRIPTION
new	void	Creates a new instance of SysAppSecurityAttribute class. This will help in checking if the user logged in has access to the specified menu item and menu item type
menuItemType	MenuItemType	Gets the Menu Item Type of the page

METHOD NAME	RETURNS	DESCRIPTION
menuItemName	MenuItemName	Gets the Menu Item Name of the page

Method new

Creates a new instance of SysAppSecurityAttribute class. This will help in checking if the user logged in has access to the specified menu item and menu item type

```
public void new ([MenuItemName _menuItemName], [MenuItemType _menuItemType])
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_menuItemName	MenuItemName	True	Menu Item Name of the page
_menuItemType	MenuItemType	True	Menu Item Type of the page like action, display, or output

Method menuItemType

Gets the Menu Item Type of the page

```
public MenuItemType menuItemType ()
```

Return Value

Menu Item Type of the page

Method menuItemName

Gets the Menu Item Name of the page

```
public MenuItemName menuItemName ()
```

Return Value

Menu Item Name of the page

Class SysAppWorkspace

This is the base class of mobile workspace. Mobile workspace classes need to extend from this class

Methods

METHOD NAME	RETURNS	DESCRIPTION
getEnumValues	List	Called during workspace initialization. Can be used to modify the enum values that are returned to AX mobile
getWorkspaceMetadata	SysAppWorkspaceMetadata	Called during workspace initialization. Can be used to modify the workspace metadata

METHOD NAME	RETURNS	DESCRIPTION
onBeginAppJob	void	Called before the start of execution of AX mobile job
onEndAppJob	void	Called after the end of execution of AX mobile job
workspaceHidden	boolean	Can be used to control whether the workspace is hidden or not. Checks that the current user has access menu item specified by SysAppWorkspaceSecurityAttribute on the workspace class. If the attribute is not specified on the class then it always returns false

Method getEnumValues

Called during workspace initialization. Can be used to modify the enum values that are returned to AX mobile

```
public List getEnumValues (EnumName _enumName)
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_enumName	EnumName	False	The enum name

Return Value

A list of enum value

Method getWorkspaceMetadata

Called during workspace initialization. Can be used to modify the workspace metadata

```
public SysAppWorkspaceMetadata getWorkspaceMetadata ()
```

Return Value

An object representing the workspace metadata

Method onBeginAppJob

Called before the start of execution of AX mobile job

```
public void onBeginAppJob (SysAppJobRequest _sysAppJobRequest)
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_sysAppJobRequest	SysAppJobRequest	False	A class containing job request parameters

Method onEndAppJob

Called after the end of execution of AX mobile job

```
public void onEndAppJob (SysAppJobResponse _sysAppJobResponse)
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_sysAppJobResponse	SysAppJobResponse	False	A class containing job response parameters

Method workspaceHidden

Can be used to control whether the workspace is hidden or not. Checks that the current user has access menu item specified by SysAppWorkspaceSecurityAttribute on the workspace class. If the attribute is not specified on the class then it always returns false

```
public boolean workspaceHidden ()
```

Return Value

Returns true if the workspace is hidden otherwise false

Class SysAppWorkspaceAttribute

Applied on classes that are extended from SysAppWorkspace

Methods

METHOD NAME	RETURNS	DESCRIPTION
new	void	Creates a new instance of SysAppWorkspaceAttribute class
AppId	str	Gets or sets the AppId of the workspace
AppResourceName	str	Gets or sets the AOT Resource name that contains the workspace
WorkspaceHidden	boolean	Gets or sets if the workspace is hidden from designer

Method new

Creates a new instance of SysAppWorkspaceAttribute class

```
public void new (str _appId, [str _appResourceName], [boolean _workspaceHidden])
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_appId	str	False	The appId of the workspace
_appResourceName	str	True	The AOT resource name that contains the workspace

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_workspaceHidden	boolean	True	The workspace is hidden from designer

Method AppId

Gets or sets the AppId of the workspace

```
public str AppId ([str _appId])
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_appId	str	True	The AppId of the workspace

Return Value

The AppId of the workspace

Method AppResourceName

Gets or sets the AOT Resource name that contains the workspace

```
public str AppResourceName ([str _appResourceName])
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_appResourceName	str	True	The AOT Resource name that contains the workspace

Return Value

The AOT Resource name that contains the workspace

Method WorkspaceHidden

Gets or sets if the workspace is hidden from designer

```
public boolean WorkspaceHidden ([boolean _workspaceHidden])
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_workspaceHidden	boolean	True	The workspace hidden

Return Value

Whether the workspace is hidden from designer or not

Class SysAppWorkspaceMetadata

This class can be used to access and update metadata of an AX mobile workspace

Methods

METHOD NAME	RETURNS	DESCRIPTION
new	void	
addConfig	void	Adds a custom config to the mobile workspace metadata
getPage	SysAppPageMetadata	Returns the page with the pageName provided
getPageEnumerator	MapEnumerator	Returns a map enumerator that can be used to enumerate workspace pages. Where key is page name and value is of type SysAppPageMetadata
getAction	SysAppActionMetadata	Returns the action with the actionName provided
getActionEnumerator	MapEnumerator	Returns a map enumerator that can be used to enumerate workspace actions. Where key is action name and value is of type SysAppActionMetadata
getPageNameForRecordingId	str	Returns a pageName if the provided recordingId is used by a workspace page
getActionNameForRecordingId	str	Returns a actionName if the provided recordingId is used by a workspace action
workspaceTitle	str	Gets and sets the workspace title
workspaceDescription	str	Gets and sets the workspace description

Method new

```
public void new (str _appId, [SysAppWorkspaceAttribute _attribute])
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_appId	str	False	
_attribute	SysAppWorkspaceAttribute	True	

Method addConfig

Adds a custom config to the mobile workspace metadata

```
public void addConfig (str _configName, object _configValue)
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_configName	str	False	A config name
_configValue	object	False	An object of an X++ data contract class

Method getPage

Returns the page with the pageName provided

```
public SysAppPageMetadata getPage (str _pageName)
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_pageName	str	False	A page name

Return Value

Returns the pageMetadata if a page with the provided name exists; otherwise null

Method getPageEnumerator

Returns a map enumerator that can be used to enumerate workspace pages. Where key is page name and value is of type SysAppPageMetadata

```
public MapEnumerator getPageEnumerator ()
```

Return Value

A map enumerator

Method getAction

Returns the action with the actionName provided

```
public SysAppActionMetadata getAction (str _actionName)
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_actionName	str	False	An action name

Return Value

Returns the ActionMetadata if an action with the provided name exists; otherwise null

Method getActionEnumerator

Returns a map enumerator that can be used to enumerate workspace actions. Where key is action name and value is of type SysAppActionMetadata

```
public MapEnumerator getActionEnumerator ()
```

Return Value

A map enumerator

Method getPageNameForRecordingId

Returns a pageName if the provided recordingId is used by a workspace page

```
public str getPageNameForRecordingId (str _recordingId)
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_recordingId	str	False	A recordingId

Return Value

A page name if the supplied recordingId is used by a workspace page; otherwise empty string

Method getActionNameForRecordingId

Returns a actionName if the provided recordingId is used by a workspace action

```
public str getActionNameForRecordingId (str _recordingId)
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_recordingId	str	False	A recordingId

Return Value

An action name if the supplied recordingId is used by a workspace action; otherwise empty string

Method workspaceTitle

Gets and sets the workspace title

```
public str workspaceTitle ([str _workspaceTitle])
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_workspaceTitle	str	True	The workspace title

Return Value

The workspace title

Method workspaceDescription

Gets and sets the workspace description

```
public str workspaceDescription ([str _workspaceDescription])
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_workspaceDescription	str	True	The workspace description

Return Value

The workspace description

Class SysAppWorkspaceSecurityAttribute

Controls the visibility based of workspace based on the menu item tied to this attribute

Methods

METHOD NAME	RETURNS	DESCRIPTION
new	void	Creates a new instance of attribute
WorkspaceMenuItemName	MenuItemName	Gets or sets the workspace menuItem for the workspace security attribute
WorkspaceMenuItemType	MenuItemType	Gets or sets the workspace menu item type for the workspace security attribute

Method new

Creates a new instance of attribute

```
public void new (MenuItemName _workspaceMenuItemName, [MenuItemType _workspaceMenuItemType])
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_workspaceMenuItemName	MenuItemName	False	The menu item name to which the workspace needs to be tied
_workspaceMenuItemType	MenuItemType	True	The menu item type

Method WorkspaceMenuItemName

Gets or sets the workspace menuItem for the workspace security attribute

```
public MenuItemName WorkspaceMenuItemName ([MenuItemName _workspaceMenuItemName])
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_workspaceMenuItemName	MenuItemName	True	The workspace menu item for the workspace security attribute

Return Value

The workspace menu item for the workspace security attribute

Method WorkspaceMenuItemType

Gets or sets the workspace menu item type for the workspace security attribute

```
public MenuItemType WorkspaceMenuItemType ([MenuItemType _workspaceMenuItemType])
```

Parameters

PARAMETER NAME	PARAMETER TYPE	OPTIONAL	DESCRIPTION
_workspaceMenuItemType	MenuItemType	True	The workspace menu item type for the `workspacesecurity` attribute

Return Value

The workspace menu item type for the workspace security attribute

Client-side design APIs

2/18/2021 • 3 minutes to read • Edit Online

This topic provides an overview of the application programming interfaces (APIs) for client-side design and includes recommendations for using them.

Terminology

The following list includes some frequently used terms that apply to the client-side design APIs.

- **Design** – A property that can optionally be specified on a Page, Action, or other component object to override its default design.
- **Component** – A component can be one of four types:
  - **Block container** (default) – A container that has CSS block behaviors. (In other words, the container is equivalent to an element that has a CSS **display: block** style declaration.)
  - **Flex container** – A container that has CSS flex behaviors. (In other words, the container is equivalent to an element that has a CSS **display: flex** style declaration.)
  - **Control reference** – A component that refers to a control that exists in the static metadata (XML) of the Page or Action.
  - **New control** – A component that instantiates a new control. (In other words, the control doesn't already exist in the static metadata [XML] of the Page or Action.)

    A component is represented in the design by a JavaScript Object Notation (JSON) object, and the properties of this JSON object represent the properties of the component. Almost every JSON object in the design property hierarchy is a component.
- **Item** – A component that is nested in a container.
- **Property** – Several types of properties can be set on a component:
  - Container-specific
  - Item-specific
  - Control-specific
  - List-specific
  - Generic (non-specific)

    Properties are specified as key-value pairs on the component's JSON object. The properties that are applicable depend on the type of component that the property is applied to.

    For properties that have a predefined list of possible values, the first value that is shown in the documentation is the property's default value. In most cases, if you don't specify a property at all (that is, if you omit the property from the JSON object), the property behaves as if you had set its default value.

    Generic properties can be applied to all component types.

    When you specify properties, follow these guidelines:

- You should not enclose property names in quotation marks.
- You must enclose all property values in double quotation marks, unless the documentation specifies otherwise.

- **Inheritance** – If a color, font size, or font weight is applied to a control, all descendant controls inherit the same property, unless they are reassigned. If padding is applied to a control, it's inherited by the item (non-container) descendants of the control. No other properties are inherited.

Using design APIs

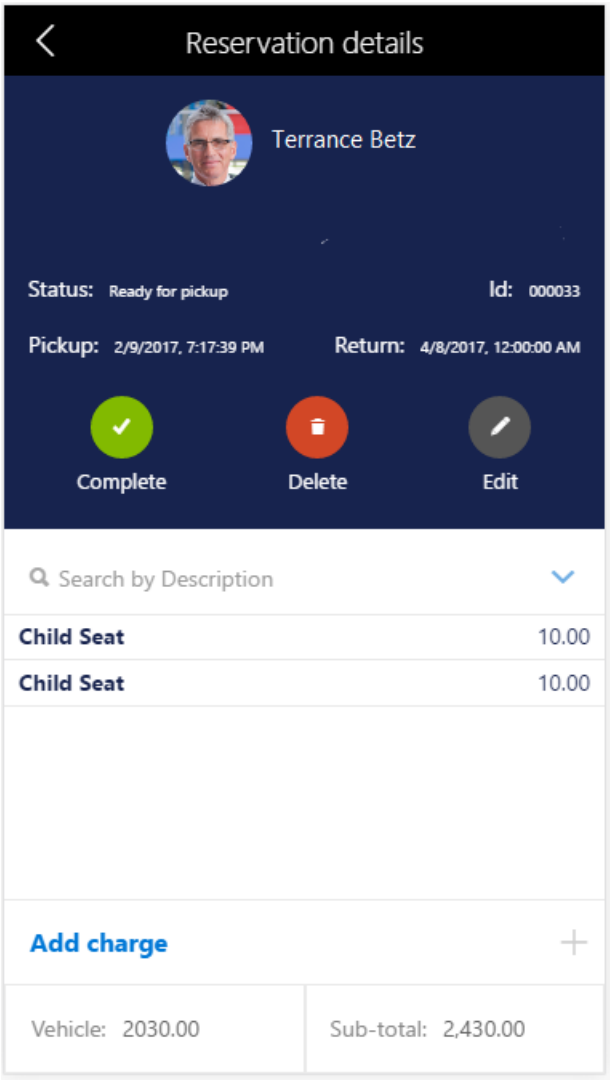
The following code is a modified segment of business logic code from a Reservation Management example. Specifically, this code is from a variable that specifies the design for a reservation details page. Comments are included in the code to highlight a few possibilities.

> **NOTE**
>
> Any color, font size, or font weight that is applied to a control is also applied to all children of that control. Padding is inherited by non-container children. No other properties are inherited. Containers include lists, pages, groups, and parts.

After a control is created that doesn't have any children or items, the control name just has to be written in quotation marks (see **FMCustomer_FullName** in the following code). However, if any customization will be applied to that control, the code must be blocked, and the **name** label must be used (see **FMCustomer_Image** in the following code).

```
// Page root container
"flexFlow":"column nowrap",
"items":[
    // Upper third of page, contains 4 rows
    {
        "flexFlow":"column nowrap",
        "background":"theme", // set background color to the theme color
        "color":"light",      // set the foreground (e.g. font) color to light
        "fontSize":"small",
        "border": "none",
        "padding":"small",
        "items":[
            // Row 1/4 with customer image and name
            {
                "flexFlow":"row nowrap",
                "alignItems":"center",
                "justifyItems":"center",
                "labelStyle":"hidden", // don't show label for field
                "fontSize":"large",
                "fontWeight":"bold",
                "items":[
                    {
                    // Customer image – since we're modifying the imageStyle, etc., this
                    // code must be blocked and the "FMCustomer_Image" must be labeled
                    // with "name".
                        "name":"FMCustomer_Image",
                        "imageStyle":"circular",
                        height:3,
                        width:3
                    },
                    // don't need to create a new object or use the "name" label if
                    // there is no customization
                    "FMCustomer_FullName",
                ]
            },
            // Row 2/4 with vehicle description
            . . .
        }
    }
```

The following illustration shows the customer image, customer name, font, background color, and so on, that preceding code produces.



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Client APIs home page

2/18/2021 • 2 minutes to read • Edit Online

- Application
  - Application
  - ApplicationMetadata
  - main
- Control
  - Container
    - ContainerControl
    - ContainerControlDesign
    - ContainerControlMetadata
  - Field
    - Field
    - FieldDesign
    - FieldMetadata
  - File Uploader
    - FileUploader
    - FileUploaderDesign
    - FileUploaderMetadata
  - Group
    - Group
    - GroupDesign
    - GroupMetadata
  - Hyperlink
    - HyperLink
    - HyperLinkDesign
    - HyperLinkMetadata
  - Image
    - Image
    - ImageDesign
    - ImageMetadata

- ControlMetadata
- ControlType

- Defer
  - all
  - defer
  - Deferred
  - reject
  - resolve

- Event
  - EventHook
  - IEventListener

- Page
  - CompleteEventArgs
  - Design
  - NavigationArgs
  - Page
  - PageMetadata
  - PageOptions
  - PageState
  - PageSubmitArgs
  - PageTarget

- Services
  - AsyncService
  - CacheService
  - DataService
  - ExpressionOperator
  - MetadataService
  - PageData

PageState enumeration

2/18/2021 • 2 minutes to read • Edit Online

Represents the various high-level states the a page can be in.

Index

Enumeration members

- error
- loaded
- loading
- offline
- refreshing

Enumeration members

error

error: 10 The page is currently in the error state, but can be refreshed in an attempt to get out of this state.

loaded

loaded: 3 The page is fully loaded and can be refreshed and, if possible, submitted.

loading

loading: 2 The page is currently being loaded.

offline

offline: 1 The page was loaded in the offline mode, thus not refreshable.

refreshing

refreshing: 4 The page is currently refreshing its data.

Application module

2/18/2021 • 2 minutes to read • Edit Online

An application is a unit of runtime execution with sandboxing around concepts and data used inside of it. Each application consists of pages, actions, data queries, and logic that glue them together. An application is primarily described with a declarative metadata system, and can have an accompanying imperative extension model.

The imperative extension of the application is typically defined in a script module with a designated entry point, the main function, which allows the imperative logic to integrate with the application life cycle.

Index

Types

- Application
- ApplicationMetadata

Functions

- main

Types

Application

Hierarchy

Application

Properties

NAME	SIGNATURE	DESCRIPTION
minVersion	minVersion: string (optional)	An optional marker to indicate the minimum platform version required by this component. When this value is specified and the component tries to load in an older version of the platform, the corresponding workspace is not loaded and user is directed to install a newer version of the platform.

Methods

NAME	SIGNATURE	DESCRIPTION
appInit	appInit(metadata: ApplicationMetadata): any	This method is invoked at the point the application is about to run, with the instance of the application metadata loaded. The metadata passed in can be still modified to change behaviors before this method returns.

ApplicationMetadata

Hierarchy

ApplicationMetadata

Properties

NAME	SIGNATURE	DESCRIPTION
ColorName	ColorName: string (optional)	The theme color of the application
Configs	Configs: [name: string]: any (optional)	An application can have a set of named config supplied by the author or the resource provider
Description	Description: string (optional)	The description of the application
IconName	IconName: string (optional)	The representative icon of the application
ID	ID: string	The unique identifier of the application
Title	Title: string	The title of the application

Functions

main

main(metadataService: MetadataService, dataService: DataService, cacheService: CacheService, asyncService: AsyncService): Application

The main method of a business logic module. Each business logic module (as JavaScript file) must contain one main method. The method is invoked when the module is loaded and is being initialized. The method must return the component to run from this module.

Parameters

NAME	TYPE	DESCRIPTION
metadataService	MetadataService	
dataService	DataService	
cacheService	CacheService	
asyncService	AsyncService	

Returns Application

Defer module

2/18/2021 • 2 minutes to read • Edit Online

Index

Types

- Deferred

Functions

- all
- defer
- reject
- resolve

Types

Deferred

Hierarchy

Deferred

Properties

NAME	SIGNATURE	DESCRIPTION
promise	promise: Promise <T>	

Methods

NAME	SIGNATURE	DESCRIPTION
reject	reject(error?: any): void	
resolve	resolve(value?: T | PromiseLike <T>): void	

Functions

all

all(...args: any [ ]): Promise <any [ ]>

Parameters

NAME	TYPE	DESCRIPTION
...args	any [ ]	

Returns Promise <any [ ]>

defer

defer <T>(): Deferred <T>

Returns Deferred <T>

reject

reject(error?: any): Promise <any>

Parameters

NAME	TYPE	DESCRIPTION
error?	any	

Returns Promise <any>

resolve

resolve <T>(value?: T | PromiseLike <T>): Promise <T>

Parameters

NAME	TYPE	DESCRIPTION
value?	T	PromiseLike <T>	

Returns Promise <T>

Event module

2/18/2021 • 2 minutes to read • Edit Online

Index

Types

- EventHook

Type aliases

- IEventListener

Types

EventHook

Hierarchy

EventHook

Methods

NAME	SIGNATURE	DESCRIPTION
subscribe	subscribe(listener: IEventListener <T>): void	Subscribe a listener to this event.
unsubscribe	unsubscribe(listener: IEventListener <T>): void	Unsubscribe a listener from this event.
unsubscribeAll	unsubscribeAll(): void	Remove all listeners from this event.

Type aliases

IEventListener

IEventListener: function

Control module

2/18/2021 • 2 minutes to read • Edit Online

Controls are what make up the content of a page.

Index

Types

- Control
- ControlMetadata

Type aliases

- ControlType

Types

Control

Hierarchy

Control
  └ PageLink
  └ ContainerControl
  └ InputControl
  └ Image

Properties

NAME	SIGNATURE	DESCRIPTION
container	container: boolean (optional)	True if the control is a container.
generic	generic: boolean (optional)	
getDataSource	getDataSource: function(): any	
hidden	hidden: boolean	True if the control is hidden.

Methods

NAME	SIGNATURE	DESCRIPTION
applyDesign	applyDesign(design: Design): void	Applies given design to the design on the control.
dataContext	dataContext(): any	
getDesign	getDesign(): Design	Returns the design object of this control.
isEditable	isEditable(): boolean	Boolean indicating if the control is editable.

NAME	SIGNATURE	DESCRIPTION
metadata	metadata(): ControlMetadata	Returns the metadata object of this control.
parent	parent(): Control | Page	Returns the parent (control or page) of this control.
root	root(): Page	Returns the root form instance (page) of this control.

ControlMetadata

Hierarchy

ControlMetadata

- PageLinkMetadata
- ContainerControlMetadata
- InputControlMetadata
- ImageMetadata

Properties

NAME	SIGNATURE	DESCRIPTION
BoundEntity	BoundEntity: string (optional)	The entity to which the control is bound.
BoundField	BoundField: string (optional)	
Description	Description: string (optional)	Description of the control.
Editable	Editable: boolean (optional)	Boolean indicating if the control is editable.
ExtType	ExtType: ControlType (optional)	The extended control type. For example, a control of type Input might have an extended type of Barcode.
HelpText	HelpText: string (optional)	The keyboard shortcut for a command. For example, "(Shift+F5)"
Hidden	Hidden: boolean (optional)	Boolean indicating if the control is hidden or not.
Id	Id: string (optional)	Identification string for a control.
Label	Label: string (optional)	Label for a control. For example, a control representing a person's first name might have a label "First Name".
Name	Name: string (optional)	Name of a control.
Order	Order: number (optional)	Number indicating the order in which a control will appear on a page.
Type	Type: ControlType (optional)	String indicating the control type.

Type aliases

ControlType

ControlType: "FileUpload" | "Barcode" | "Input" | "MultilineInput" | "Navigation" | "Integer" | "Int64" | "Date" | "DateTime" | "ComboBox" | "Real" | "List" | "Lookup" | "MultiLookup" | "Navigation" | "Image" | "Group" | "Part" | "Calendar" | "HyperLink" | "Timer"

Controls must be assigned any of the types listed in ControlType.

Container module (Client APIs)

2/18/2021 • 4 minutes to read • Edit Online

A container control can contain any number of controls.

Index

Types

- ContainerControl
- ContainerControlDesign
- ContainerControlMetadata

Types

ContainerControl

Hierarchy

Control
└ ContainerControl
└ Group
└ List
└ Part

Properties

NAME	SIGNATURE	DESCRIPTION
container	container: boolean	True if the control is a container. Overrides Control.container
generic	generic: boolean (optional)	Inherited from Control.generic
getDataSource	getDataSource: function(): any	Inherited from Control.getDataSource
hidden	hidden: boolean	True if the control is hidden. Inherited from Control.hidden

Methods

NAME	SIGNATURE	DESCRIPTION
applyDesign	applyDesign(design: Design): void	Applies given design to the design on the control. Inherited from Control.applyDesign
dataContext	dataContext(): any	Inherited from Control.dataContext
getControl	getControl(controlName: string): Control	Given the name of a control, returns the control instance.
getControlById	getControlById(id: string): Control	Given the ID of a control, returns the control instance.

NAME	SIGNATURE	DESCRIPTION
getDesign	getDesign(): Design	Returns the design object of this control. Inherited from Control.getDesign
isEditable	isEditable(): boolean	Boolean indicating if the control is editable. Inherited from Control.isEditable
metadata	metadata(): ContainerControlMetadata	Returns the metadata object of this control. Overrides Control.metadata
parent	parent(): Control | Page	Returns the parent (control or page) of this control. Inherited from Control.parent
root	root(): Page	Returns the root form instance (page) of this control. Inherited from Control.root

ContainerControlDesign

Hierarchy

Design
└ ContainerControlDesign
└ GroupDesign
└ ListDesign
└ PartDesign

Properties

NAME	SIGNATURE	DESCRIPTION
alignItems	alignItems: string (optional)	This property is an alias for the CSS property "align-items". Inherited from Design.alignItems
alignSelf	alignSelf: string (optional)	Inherited from Design.alignSelf
allowScroll	allowScroll: string (optional)	True if the container will allow scrolling when its items do not fit into the container's available space. If a container has an item which may scroll, then set this property to false to prevent nested scrolling areas.
background	background: string (optional)	The background color of the container.
bindings	bindings: any (optional)	Inherited from Design.bindings
border	border: "none" | "solid" | "left" | "right" | "top" | "bottom" (optional)	The border behavior of a control. This property will not be inherited by the children. Inherited from Design.border

NAME	SIGNATURE	DESCRIPTION
color	color: string (optional)	The foreground color of the container. Inherited from Design.color
flexFlow	flexFlow: string (optional)	Specifying this property makes the component a flex container component. Inherited from Design.flexFlow
flexSize	flexSize: string (optional)	One number or two numbers written as a string. For example, "(size to grow) [(size-to-shrink)]" to accommodate available space in the immediate flex container. Inherited from Design.flexSize
fontSize	fontSize: "medium" | "xx-small" | "x-small" | "small" | "large" | "x-large" | "xx-large" (optional)	The proportional text size Inherited from Design.fontSize
fontWeight	fontWeight: "normal" | "bold" (optional)	Normal or bold text. Inherited from Design.fontWeight
itemBorder	itemBorder: "solid" | "none" (optional)	If true, a border will appear around each row in the list.
items	items: string | Design [ ] (optional)	An array containing the components to place inside of the container.
justifyItems	justifyItems: "flex-start" | "flex-end" | "center" | "space-between" (optional)	This property is an alias for the CSS property "justify-content". Inherited from Design.justifyItems
label	label: string (optional)	Inherited from Design.label
labelPosition	labelPosition: "stacked" | "hidden" | "inline" (optional)	Determines how a label is positioned, if at all. By default, labelPosition is set to stacked. Inherited from Design.labelPosition
name	name: string (optional)	Inherited from Design.name
padding	padding: "none" | "small" | "std" (optional)	Allows specifying the component's padding behavior. Inherited from Design.padding
type	type: ControlType (optional)	The type of the control as a string. Inherited from Design.type

ContainerControlMetadata

Hierarchy

ControlMetadata
- ContainerControlMetadata
  - GroupMetadata
  - ListMetadata

└ PartMetadata

Properties

NAME	SIGNATURE	DESCRIPTION
BoundEntity	BoundEntity: string (optional)	The entity to which the control is bound. Inherited from ControlMetadata.BoundEntity
BoundField	BoundField: string (optional)	Inherited from ControlMetadata.BoundField
Description	Description: string (optional)	Description of the control. Inherited from ControlMetadata.Description
Editable	Editable: boolean (optional)	Boolean indicating if the control is editable. Inherited from ControlMetadata.Editable
ExtType	ExtType: ControlType (optional)	The extended control type. For example, a control of type Input might have an extended type of Barcode. Inherited from ControlMetadata.ExtType
HelpText	HelpText: string (optional)	The keyboard shortcut for a command. For example, "(Shift+F5)" Inherited from ControlMetadata.HelpText
Hidden	Hidden: boolean (optional)	Boolean indicating if the control is hidden or not. Inherited from ControlMetadata.Hidden
Id	Id: string (optional)	Identification string for a control. Inherited from ControlMetadata.Id
Label	Label: string (optional)	Label for a control. For example, a control representing a person's first name might have a label "First Name". Inherited from ControlMetadata.Label
Name	Name: string (optional)	Name of a control. Inherited from ControlMetadata.Name
Order	Order: number (optional)	Number indicating the order in which a control will appear on a page. Inherited from ControlMetadata.Order
Type	Type: ControlType (optional)	String indicating the control type. Inherited from ControlMetadata.Type

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Field module

2/18/2021 • 5 minutes to read • Edit Online

Represents the run-time instance of a field.

Index

Types

- Field
- FieldDesign
- FieldMetadata

Types

Field

Hierarchy

InputControl
└ Field

Properties

NAME	SIGNATURE	DESCRIPTION
container	container: boolean (optional)	True if the control is a container. Inherited from Control.container
generic	generic: boolean (optional)	Inherited from Control.generic
getDataSource	getDataSource: function(): any	Inherited from Control.getDataSource
hidden	hidden: boolean	True if the control is hidden. Inherited from Control.hidden

Methods

NAME	SIGNATURE	DESCRIPTION
applyDesign	applyDesign(IDesign: FieldDesign): void	Applies given design to the design on the control. Overrides Control.applyDesign
dataContext	dataContext(): any	Inherited from Control.dataContext
getDesign	getDesign(): Design	Returns the design object of this control. Inherited from Control.getDesign
getEditableFormattedValue	getEditableFormattedValue(): string | number | Date	Gets a formatted decimal string value of an editable field control.

NAME	SIGNATURE	DESCRIPTION
getEditableValue	getEditableValue(): string | number | Date	Gets the value for an editable field control.
getEntityRef	getEntityRef(): any	Gets value of entityRef binding to control.
getFormattedValue	getFormattedValue(): string	Gets a formatted decimal string value.
getRefLink	getRefLink(): NavigationArgs	Gets the navigation object for a reference link.
getValue	getValue(): any	Gets the value for a field control.
hasRefLink	hasRefLink(): boolean	Returns true if the field has a refLink, otherwise false.
hasUnWrapText	hasUnWrapText(): boolean	Gets wrap text property of control.
isEditable	isEditable(): boolean	Boolean indicating if the control is editable. Inherited from Control.isEditable
metadata	metadata(): FieldMetadata	Returns the metadata object of this control. Overrides InputControl.metadata
parent	parent(): Control | Page	Returns the parent (control or page) of this control. Inherited from Control.parent
root	root(): Page	Returns the root form instance (page) of this control. Inherited from Control.root
setEditableValue	setEditableValue(value: string | number | Date): void	Sets the value for an editable field control.

Events

NAME	SIGNATURE	DESCRIPTION
onDataChanged	onDataChanged: EventHook <null>	An event that is triggered when the input control's data changes. Inherited from InputControl.onDataChanged

FieldDesign

Hierarchy

InputControlDesign

└ FieldDesign

Properties

NAME	SIGNATURE	DESCRIPTION
alignItems	alignItems: string (optional)	This property is an alias for the CSS property "align-items". Inherited from Design.alignItems
alignSelf	alignSelf: string (optional)	Inherited from Design.alignSelf
bindings	bindings: any (optional)	Inherited from Design.bindings
border	border: "none" | "solid" | "left" | "right" | "top" | "bottom" (optional)	The border behavior of a control. This property will not be inherited by the children. Inherited from Design.border
color	color: string (optional)	The foreground color of the container. Inherited from Design.color
flexFlow	flexFlow: string (optional)	Specifying this property makes the component a flex container component. Inherited from Design.flexFlow
flexSize	flexSize: string (optional)	One number or two numbers written as a string. For example, "(size to grow) [(size-to-shrink)]" to accommodate available space in the immediate flex container. Inherited from Design.flexSize
fontSize	fontSize: "medium" | "xx-small" | "x-small" | "small" | "large" | "x-large" | "xx-large" (optional)	The proportional text size Inherited from Design.fontSize
fontWeight	fontWeight: "normal" | "bold" (optional)	Normal or bold text. Inherited from Design.fontWeight
justifyItems	justifyItems: "flex-start" | "flex-end" | "center" | "space-between" (optional)	This property is an alias for the CSS property "justify-content". Inherited from Design.justifyItems
label	label: string (optional)	Inherited from Design.label
labelPosition	labelPosition: "stacked" | "hidden" | "inline" (optional)	Determines how a label is positioned, if at all. By default, labelPosition is set to stacked. Inherited from Design.labelPosition
name	name: string (optional)	Inherited from Design.name
padding	padding: "none" | "small" | "std" (optional)	Allows specifying the component's padding behavior. Inherited from Design.padding
type	type: ControlType (optional)	The type of the control as a string. Inherited from Design.type

FieldMetadata

Hierarchy

InputControlMetadata

└ FieldMetadata

Properties

NAME	SIGNATURE	DESCRIPTION
BoundEntity	BoundEntity: string (optional)	The entity to which the control is bound. Inherited from ControlMetadata.BoundEntity
BoundField	BoundField: string (optional)	Inherited from ControlMetadata.BoundField
DecimalPlaces	DecimalPlaces: number (optional)	The number of decimals that appear on a field of type "Real".
Description	Description: string (optional)	Description of the control. Inherited from ControlMetadata.Description
Editable	Editable: boolean (optional)	Boolean indicating if the control is editable. Inherited from ControlMetadata.Editable
ExtType	ExtType: ControlType (optional)	The extended control type. For example, a control of type Input might have an extended type of Barcode. Inherited from ControlMetadata.ExtType
Formatting	Formatting: any (optional)	Formats a field of type "DateTime" or "Date".
HelpText	HelpText: string (optional)	The keyboard shortcut for a command. For example, "(Shift+F5)" Inherited from ControlMetadata.HelpText
Hidden	Hidden: boolean (optional)	Boolean indicating if the control is hidden or not. Inherited from ControlMetadata.Hidden
Id	Id: string (optional)	Identification string for a control. Inherited from ControlMetadata.Id
Label	Label: string (optional)	Label for a control. For example, a control representing a person's first name might have a label "First Name". Inherited from ControlMetadata.Label

NAME	SIGNATURE	DESCRIPTION
LinkType	LinkType: "Telephone" | "Email" | "Url" (optional)	Assigning the link type of a field allows for the appropriate mobile application to be opened when the link is selected.
Mandatory	Mandatory: boolean (optional)	If set to true then input for the control is required for the task to be completed. Mandatory controls will have a red outline. Inherited from InputControlMetadata.Mandatory
Name	Name: string (optional)	Name of a control. Inherited from ControlMetadata.Name
NumSequence	NumSequence: NumberSequenceConfig (optional)	Used for auto detecting and changing visibility of the number sequence controls in the task or page, based on AX number sequence configuration, through extended business logic. Inherited from InputControlMetadata.NumSequence
Order	Order: number (optional)	Number indicating the order in which a control will appear on a page. Inherited from ControlMetadata.Order
ReferenceAppId	ReferenceAppId: string (optional)	The ID of the app that the field control lives in.
ReferencePageId	ReferencePageId: string (optional)	The ID of the page that the field control lives in.
Style	Style: string (optional)	Styles a field of type "DateTime" or "Date".
Type	Type: ControlType (optional)	String indicating the control type. Inherited from ControlMetadata.Type
UnWrapText	UnWrapText: boolean (optional)	False by default -- text of the page will be wrapped.
WrapText	WrapText: boolean (optional)	If true then the text of the field control will wrap to the next line.

File Uploader module

2/18/2021 • 4 minutes to read • Edit Online

A control for uploading images.

Index

Types

- FileUploader
- FileUploaderDesign
- FileUploaderMetadata

Types

FileUploader

Hierarchy

Value
└ FileUploader

Properties

NAME	SIGNATURE	DESCRIPTION
container	container: boolean (optional)	True if the control is a container. Inherited from Control.container
generic	generic: boolean (optional)	Inherited from Control.generic
getDataSource	getDataSource: function(): any	Inherited from Control.getDataSource
hidden	hidden: boolean	True if the control is hidden. Inherited from Control.hidden
image	image: Image	

Methods

NAME	SIGNATURE	DESCRIPTION
applyDesign	applyDesign(IDesign: FileUploaderDesign): void	Applies given design to the design on the control. Overrides Control.applyDesign
canLoadFromDevice	canLoadFromDevice(): boolean	Returns true if the mobile phone has camera plugin.
dataContext	dataContext(): any	Inherited from Control.dataContext
getDesign	getDesign(): Design	Returns the design object of this control. Inherited from Control.getDesign

NAME	SIGNATURE	DESCRIPTION
getImage	getImage(options: any): Promise <string>	Returns a promise of an object with image data.
getValue	getValue(): any	Gets the value of the entity that is bound to the control. Overrides Value.getValue
isEditable	isEditable(): boolean	Boolean indicating if the control is editable. Inherited from Control.isEditable
loadFromFileSystem	loadFromFileSystem(file: Blob): Promise <any>	
metadata	metadata(): FileUploaderMetadata	Returns the metadata object of this control. Overrides Value.metadata
parent	parent(): Control | Page	Returns the parent (control or page) of this control. Inherited from Control.parent
root	root(): Page	Returns the root form instance (page) of this control. Inherited from Control.root
setCamera	setCamera(camera: any): void	Set the camera object on the control.
setValue	setValue(value: string): void	Sets the value of the control. Inherited from Value.setValue

Events

NAME	SIGNATURE	DESCRIPTION
onDataChanged	onDataChanged: EventHook <null>	An event that is triggered when the input control's data changes. Inherited from InputControl.onDataChanged

FileUploaderDesign

Hierarchy

ValueDesign

└ FileUploaderDesign

Properties

NAME	SIGNATURE	DESCRIPTION
alignItems	alignItems: string (optional)	This property is an alias for the CSS property "align-items". Inherited from Design.alignItems
alignSelf	alignSelf: string (optional)	Inherited from Design.alignSelf

NAME	SIGNATURE	DESCRIPTION
bindings	bindings: any (optional)	Inherited from Design.bindings
border	border: "none" | "solid" | "left" | "right" | "top" | "bottom" (optional)	The border behavior of a control. This property will not be inherited by the children. Inherited from Design.border
color	color: string (optional)	The foreground color of the container. Inherited from Design.color
flexFlow	flexFlow: string (optional)	Specifying this property makes the component a flex container component. Inherited from Design.flexFlow
flexSize	flexSize: string (optional)	One number or two numbers written as a string. For example, "(size to grow) [(size-to-shrink)]" to accommodate available space in the immediate flex container. Inherited from Design.flexSize
fontSize	fontSize: "medium" | "xx-small" | "x-small" | "small" | "large" | "x-large" | "xx-large" (optional)	The proportional text size Inherited from Design.fontSize
fontWeight	fontWeight: "normal" | "bold" (optional)	Normal or bold text. Inherited from Design.fontWeight
justifyItems	justifyItems: "flex-start" | "flex-end" | "center" | "space-between" (optional)	This property is an alias for the CSS property "justify-content". Inherited from Design.justifyItems
label	label: string (optional)	Inherited from Design.label
labelPosition	labelPosition: "stacked" | "hidden" | "inline" (optional)	Determines how a label is positioned, if at all. By default, labelPosition is set to stacked. Inherited from Design.labelPosition
name	name: string (optional)	Inherited from Design.name
padding	padding: "none" | "small" | "std" (optional)	Allows specifying the component's padding behavior. Inherited from Design.padding
type	type: ControlType (optional)	The type of the control as a string. Inherited from Design.type

FileUploaderMetadata

Hierarchy

ValueMetadata

└ FileUploaderMetadata

Properties

NAME	SIGNATURE	DESCRIPTION
BoundEntity	BoundEntity: string (optional)	The entity to which the control is bound. Inherited from ControlMetadata.BoundEntity
BoundField	BoundField: string (optional)	Inherited from ControlMetadata.BoundField
Description	Description: string (optional)	Description of the control. Inherited from ControlMetadata.Description
Editable	Editable: boolean (optional)	Boolean indicating if the control is editable. Inherited from ControlMetadata.Editable
ExtType	ExtType: ControlType (optional)	The extended control type. For example, a control of type Input might have an extended type of Barcode. Inherited from ControlMetadata.ExtType
HelpText	HelpText: string (optional)	The keyboard shortcut for a command. For example, "(Shift+F5)" Inherited from ControlMetadata.HelpText
Hidden	Hidden: boolean (optional)	Boolean indicating if the control is hidden or not. Inherited from ControlMetadata.Hidden
Id	Id: string (optional)	Identification string for a control. Inherited from ControlMetadata.Id
Label	Label: string (optional)	Label for a control. For example, a control representing a person's first name might have a label "First Name". Inherited from ControlMetadata.Label
Mandatory	Mandatory: boolean (optional)	If set to true then input for the control is required for the task to be completed. Mandatory controls will have a red outline. Inherited from InputControlMetadata.Mandatory
Name	Name: string (optional)	Name of a control. Inherited from ControlMetadata.Name

NAME	SIGNATURE	DESCRIPTION
NumSequence	NumSequence: NumberSequenceConfig (optional)	Used for auto detecting and changing visibility of the number sequence controls in the task or page, based on AX number sequence configuration, through extended business logic. Inherited from InputControlMetadata.NumSequence
Order	Order: number (optional)	Number indicating the order in which a control will appear on a page. Inherited from ControlMetadata.Order
Type	Type: ControlType (optional)	String indicating the control type. Inherited from ControlMetadata.Type

Group module

2/18/2021 • 4 minutes to read • Edit Online

A group control is a container control that has any number of controls as children.

Index

Types

- Group
- GroupDesign
- GroupMetadata

Types

Group

Hierarchy

ContainerControl
└ Group

Properties

NAME	SIGNATURE	DESCRIPTION
container	container: boolean	True if the control is a container. Inherited from ContainerControl.container Overrides Control.container
generic	generic: boolean (optional)	Inherited from Control.generic
getDataSource	getDataSource: function(): any	Inherited from Control.getDataSource
hidden	hidden: boolean	True if the control is hidden. Inherited from Control.hidden

Methods

NAME	SIGNATURE	DESCRIPTION
applyDesign	applyDesign(IDesign: GroupDesign): void	Applies given design to the design on the control. Overrides Control.applyDesign
dataContext	dataContext(): any	Inherited from Control.dataContext
getChildren	getChildren(): Control [ ]	Returns the list of children associated with this group control.
getControl	getControl(controlName: string): Control	Given the name of a control, returns the control instance. Inherited from ContainerControl.getControl

NAME	SIGNATURE	DESCRIPTION
getControlById	getControlById(id: string): Control	Given the ID of a control, returns the control instance. Inherited from ContainerControl.getControlById
getDesign	getDesign(): Design	Returns the design object of this control. Inherited from Control.getDesign
isEditable	isEditable(): boolean	Boolean indicating if the control is editable. Inherited from Control.isEditable
metadata	metadata(): GroupMetadata	Returns the metadata object of this control. Overrides ContainerControl.metadata
parent	parent(): Control | Page	Returns the parent (control or page) of this control. Inherited from Control.parent
root	root(): Page	Returns the root form instance (page) of this control. Inherited from Control.root

GroupDesign

Hierarchy

ContainerControlDesign
└ GroupDesign

Properties

NAME	SIGNATURE	DESCRIPTION
alignItems	alignItems: string (optional)	This property is an alias for the CSS property "align-items". Inherited from Design.alignItems
alignSelf	alignSelf: string (optional)	Inherited from Design.alignSelf
allowScroll	allowScroll: string (optional)	True if the container will allow scrolling when its items do not fit into the container's available space. If a container has an item which may scroll, then set this property to false to prevent nested scrolling areas. Inherited from ContainerControlDesign.allowScroll
background	background: string (optional)	The background color of the container. Inherited from ContainerControlDesign.background
bindings	bindings: any (optional)	Inherited from Design.bindings

NAME	SIGNATURE	DESCRIPTION
border	border: "none" | "solid" | "left" | "right" | "top" | "bottom" (optional)	The border behavior of a control. This property will not be inherited by the children. Inherited from Design.border
color	color: string (optional)	The foreground color of the container. Inherited from Design.color
flexFlow	flexFlow: string (optional)	Specifying this property makes the component a flex container component. Inherited from Design.flexFlow
flexSize	flexSize: string (optional)	One number or two numbers written as a string. For example, "(size to grow) [(size-to-shrink)]" to accommodate available space in the immediate flex container. Inherited from Design.flexSize
fontSize	fontSize: "medium" | "xx-small" | "x-small" | "small" | "large" | "x-large" | "xx-large" (optional)	The proportional text size Inherited from Design.fontSize
fontWeight	fontWeight: "normal" | "bold" (optional)	Normal or bold text. Inherited from Design.fontWeight
itemBorder	itemBorder: "solid" | "none" (optional)	If true, a border will appear around each row in the list. Inherited from ContainerControlDesign.itemBorder
items	items: string | Design [ ] (optional)	An array containing the components to place inside of the container. Inherited from ContainerControlDesign.items
justifyItems	justifyItems: "flex-start" | "flex-end" | "center" | "space-between" (optional)	This property is an alias for the CSS property "justify-content". Inherited from Design.justifyItems
label	label: string (optional)	Inherited from Design.label
labelPosition	labelPosition: "stacked" | "hidden" | "inline" (optional)	Determines how a label is positioned, if at all. By default, labelPosition is set to stacked. Inherited from Design.labelPosition
name	name: string (optional)	Inherited from Design.name
padding	padding: "none" | "small" | "std" (optional)	Allows specifying the component's padding behavior. Inherited from Design.padding

NAME	SIGNATURE	DESCRIPTION
type	type: ControlType (optional)	The type of the control as a string. Inherited from Design.type

GroupMetadata

Hierarchy

ContainerControlMetadata

└ GroupMetadata

Properties

NAME	SIGNATURE	DESCRIPTION
BoundEntity	BoundEntity: string (optional)	The entity to which the control is bound. Inherited from ControlMetadata.BoundEntity
BoundField	BoundField: string (optional)	Inherited from ControlMetadata.BoundField
Children	Children: ControlMetadata [ ] (optional)	List of control metadata for each child control.
Description	Description: string (optional)	Description of the control. Inherited from ControlMetadata.Description
Editable	Editable: boolean (optional)	Boolean indicating if the control is editable. Inherited from ControlMetadata.Editable
ExtType	ExtType: ControlType (optional)	The extended control type. For example, a control of type Input might have an extended type of Barcode. Inherited from ControlMetadata.ExtType
HelpText	HelpText: string (optional)	The keyboard shortcut for a command. For example, "(Shift+F5)" Inherited from ControlMetadata.HelpText
Hidden	Hidden: boolean (optional)	Boolean indicating if the control is hidden or not. Inherited from ControlMetadata.Hidden
Id	Id: string (optional)	Identification string for a control. Inherited from ControlMetadata.Id
Label	Label: string (optional)	Label for a control. For example, a control representing a person's first name might have a label "First Name". Inherited from ControlMetadata.Label

NAME	SIGNATURE	DESCRIPTION
Name	Name: string (optional)	Name of a control. Inherited from ControlMetadata.Name
Order	Order: number (optional)	Number indicating the order in which a control will appear on a page. Inherited from ControlMetadata.Order
Type	Type: ControlType (optional)	String indicating the control type. Inherited from ControlMetadata.Type

Hyperlink module

2/18/2021 • 4 minutes to read • Edit Online

Hyperlink control is a control to represent hyperlinks. Pagelinks can also be used in most cases.

Index

Types

- HyperLink
- HyperLinkDesign
- HyperLinkMetadata

Types

HyperLink

Hierarchy

Value
└ HyperLink

Properties

NAME	SIGNATURE	DESCRIPTION
container	container: boolean (optional)	True if the control is a container. Inherited from Control.container
generic	generic: boolean (optional)	Inherited from Control.generic
getDataSource	getDataSource: function(): any	Inherited from Control.getDataSource
hidden	hidden: boolean	True if the control is hidden. Inherited from Control.hidden

Methods

NAME	SIGNATURE	DESCRIPTION
applyDesign	applyDesign(IDesign: HyperLinkDesign): void	Applies given design to the design on the control. Overrides Control.applyDesign
dataContext	dataContext(): any	Inherited from Control.dataContext
getDesign	getDesign(): Design	Returns the design object of this control. Inherited from Control.getDesign
getHyperLinkValue	getHyperLinkValue(): string	
getValue	getValue(): string	Returns the value of the control. Inherited from Value.getValue

NAME	SIGNATURE	DESCRIPTION
isEditable	isEditable(): boolean	Boolean indicating if the control is editable. Inherited from Control.isEditable
isHyperLinkURLPresent	isHyperLinkURLPresent(): boolean	
metadata	metadata(): HyperLinkMetadata	Returns the metadata object of this control. Overrides Value.metadata
parent	parent(): Control | Page	Returns the parent (control or page) of this control. Inherited from Control.parent
root	root(): Page	Returns the root form instance (page) of this control. Inherited from Control.root
setBaseURL	setBaseURL(url: string): any	
setValue	setValue(value: string): void	Sets the value of the control. Inherited from Value.setValue

Events

NAME	SIGNATURE	DESCRIPTION
onDataChanged	onDataChanged: EventHook <null>	An event that is triggered when the input control's data changes. Inherited from InputControl.onDataChanged

HyperLinkDesign

Hierarchy

ValueDesign

└ HyperLinkDesign

Properties

NAME	SIGNATURE	DESCRIPTION
alignItems	alignItems: string (optional)	This property is an alias for the CSS property "align-items". Inherited from Design.alignItems
alignSelf	alignSelf: string (optional)	Inherited from Design.alignSelf
bindings	bindings: any (optional)	Inherited from Design.bindings
border	border: "none" | "solid" | "left" | "right" | "top" | "bottom" (optional)	The border behavior of a control. This property will not be inherited by the children. Inherited from Design.border

NAME	SIGNATURE	DESCRIPTION
color	color: string (optional)	The foreground color of the container. Inherited from Design.color
flexFlow	flexFlow: string (optional)	Specifying this property makes the component a flex container component. Inherited from Design.flexFlow
flexSize	flexSize: string (optional)	One number or two numbers written as a string. For example, "(size to grow) [(size-to-shrink)]" to accommodate available space in the immediate flex container. Inherited from Design.flexSize
fontSize	fontSize: "medium" | "xx-small" | "x-small" | "small" | "large" | "x-large" | "xx-large" (optional)	The proportional text size Inherited from Design.fontSize
fontWeight	fontWeight: "normal" | "bold" (optional)	Normal or bold text. Inherited from Design.fontWeight
justifyItems	justifyItems: "flex-start" | "flex-end" | "center" | "space-between" (optional)	This property is an alias for the CSS property "justify-content". Inherited from Design.justifyItems
label	label: string (optional)	Inherited from Design.label
labelPosition	labelPosition: "stacked" | "hidden" | "inline" (optional)	Determines how a label is positioned, if at all. By default, labelPosition is set to stacked. Inherited from Design.labelPosition
name	name: string (optional)	Inherited from Design.name
padding	padding: "none" | "small" | "std" (optional)	Allows specifying the component's padding behavior. Inherited from Design.padding
type	type: ControlType (optional)	The type of the control as a string. Inherited from Design.type

HyperLinkMetadata

Hierarchy

ValueMetadata

└ HyperLinkMetadata

Properties

NAME	SIGNATURE	DESCRIPTION
BoundEntity	BoundEntity: string (optional)	The entity to which the control is bound. Inherited from ControlMetadata.BoundEntity

NAME	SIGNATURE	DESCRIPTION
BoundField	BoundField: string (optional)	Inherited from ControlMetadata.BoundField
Description	Description: string (optional)	Description of the control. Inherited from ControlMetadata.Description
Editable	Editable: boolean (optional)	Boolean indicating if the control is editable. Inherited from ControlMetadata.Editable
ExtType	ExtType: ControlType (optional)	The extended control type. For example, a control of type Input might have an extended type of Barcode. Inherited from ControlMetadata.ExtType
HelpText	HelpText: string (optional)	The keyboard shortcut for a command. For example, "(Shift+F5)" Inherited from ControlMetadata.HelpText
Hidden	Hidden: boolean (optional)	Boolean indicating if the control is hidden or not. Inherited from ControlMetadata.Hidden
Id	Id: string (optional)	Identification string for a control. Inherited from ControlMetadata.Id
Label	Label: string (optional)	Label for a control. For example, a control representing a person's first name might have a label "First Name". Inherited from ControlMetadata.Label
Mandatory	Mandatory: boolean (optional)	If set to true then input for the control is required for the task to be completed. Mandatory controls will have a red outline. Inherited from InputControlMetadata.Mandatory
Name	Name: string (optional)	Name of a control. Inherited from ControlMetadata.Name
NumSequence	NumSequence: NumberSequenceConfig (optional)	Used for auto detecting and changing visibility of the number sequence controls in the task or page, based on AX number sequence configuration, through extended business logic. Inherited from InputControlMetadata.NumSequence

NAME	SIGNATURE	DESCRIPTION
Order	Order: number (optional)	Number indicating the order in which a control will appear on a page. Inherited from ControlMetadata.Order
Type	Type: ControlType (optional)	String indicating the control type. Inherited from ControlMetadata.Type

Image module

2/18/2021 • 4 minutes to read • Edit Online

Image control for representing images in the mobile app. Images can be of any of the following types: DataUri, Base64, URL, AOTResource, or Symbol.

Index

Types

- Image
- ImageDesign
- ImageMetadata

Type aliases

- ImageStyleType

Types

Image

Hierarchy

Control
└ Image

Properties

NAME	SIGNATURE	DESCRIPTION
container	container: boolean (optional)	True if the control is a container. Inherited from Control.container
generic	generic: boolean (optional)	Inherited from Control.generic
getDataSource	getDataSource: function(): any	Inherited from Control.getDataSource
hidden	hidden: boolean	True if the control is hidden. Inherited from Control.hidden
imageSource	imageSource: string	Defines the imageSource.
imageView	imageView: string	Dictates the style of the image.
placeholderClass	placeholderClass: string	
symbol	symbol: string	Defines the symbol if the image is of type symbol.

Methods

NAME	SIGNATURE	DESCRIPTION
applyDesign	applyDesign(design: ImageDesign): void	Applies given design to the design on the control. Overrides Control.applyDesign
dataContext	dataContext(): any	Inherited from Control.dataContext
getDesign	getDesign(): Design	Returns the design object of this control. Inherited from Control.getDesign
isEditable	isEditable(): boolean	Boolean indicating if the control is editable. Inherited from Control.isEditable
metadata	metadata(): ImageMetadata	Returns the metadata object of this control. Overrides Control.metadata
parent	parent(): Control | Page	Returns the parent (control or page) of this control. Inherited from Control.parent
root	root(): Page	Returns the root form instance (page) of this control. Inherited from Control.root

ImageDesign

Hierarchy

Design

└ ImageDesign

Properties

NAME	SIGNATURE	DESCRIPTION
alignItems	alignItems: string (optional)	This property is an alias for the CSS property "align-items". Inherited from Design.alignItems
alignSelf	alignSelf: string (optional)	Inherited from Design.alignSelf
bindings	bindings: any (optional)	Inherited from Design.bindings
border	border: "none" | "solid" | "left" | "right" | "top" | "bottom" (optional)	The border behavior of a control. This property will not be inherited by the children. Inherited from Design.border
color	color: string (optional)	The foreground color of the container. Inherited from Design.color
flexFlow	flexFlow: string (optional)	Specifying this property makes the component a flex container component. Inherited from Design.flexFlow

NAME	SIGNATURE	DESCRIPTION
flexSize	flexSize: string (optional)	One number or two numbers written as a string. For example, "(size to grow) [(size-to-shrink)]" to accommodate available space in the immediate flex container. Inherited from Design.flexSize
fontSize	fontSize: "medium" | "xx-small" | "x-small" | "small" | "large" | "x-large" | "xx-large" (optional)	The proportional text size Inherited from Design.fontSize
fontWeight	fontWeight: "normal" | "bold" (optional)	Normal or bold text. Inherited from Design.fontWeight
height	height: string (optional)	The relative vertical size of the image.
imageStyle	imageStyle: ImageStyleType (optional)	The style of the image.
justifyItems	justifyItems: "flex-start" | "flex-end" | "center" | "space-between" (optional)	This property is an alias for the CSS property "justify-content". Inherited from Design.justifyItems
label	label: string (optional)	Inherited from Design.label
labelPosition	labelPosition: "stacked" | "hidden" | "inline" (optional)	Determines how a label is positioned, if at all. By default, labelPosition is set to stacked. Inherited from Design.labelPosition
name	name: string (optional)	Inherited from Design.name
padding	padding: "none" | "small" | "std" (optional)	Allows specifying the component's padding behavior. Inherited from Design.padding
type	type: ControlType (optional)	The type of the control as a string. Inherited from Design.type
width	width: string (optional)	The relative horizontal size of the image.

ImageMetadata

Hierarchy

ControlMetadata

└ ImageMetadata

Properties

NAME	SIGNATURE	DESCRIPTION
BaseUrl	BaseUrl: string (optional)	Base URL for AOTResource type image.

NAME	SIGNATURE	DESCRIPTION
BoundEntity	BoundEntity: string (optional)	The entity to which the control is bound. Inherited from ControlMetadata.BoundEntity
BoundField	BoundField: string (optional)	Inherited from ControlMetadata.BoundField
Description	Description: string (optional)	Description of the control. Inherited from ControlMetadata.Description
Editable	Editable: boolean (optional)	Boolean indicating if the control is editable. Inherited from ControlMetadata.Editable
ExtType	ExtType: ControlType (optional)	The extended control type. For example, a control of type Input might have an extended type of Barcode. Inherited from ControlMetadata.ExtType
Height	Height: number (optional)	The relative vertical size of the image.
HelpText	HelpText: string (optional)	The keyboard shortcut for a command. For example, "(Shift+F5)" Inherited from ControlMetadata.HelpText
Hidden	Hidden: boolean (optional)	Boolean indicating if the control is hidden or not. Inherited from ControlMetadata.Hidden
Id	Id: string (optional)	Identification string for a control. Inherited from ControlMetadata.Id
ImageStyle	ImageStyle: ImageStyleType (optional)	The style of the image.
Label	Label: string (optional)	Label for a control. For example, a control representing a person's first name might have a label "First Name". Inherited from ControlMetadata.Label
Name	Name: string (optional)	Name of a control. Inherited from ControlMetadata.Name
Order	Order: number (optional)	Number indicating the order in which a control will appear on a page. Inherited from ControlMetadata.Order
Type	Type: ControlType (optional)	String indicating the control type. Inherited from ControlMetadata.Type

NAME	SIGNATURE	DESCRIPTION
Width	Width: number (optional)	The relative horizontal size of the image.

Type aliases

ImageStyleType

ImageStyleType: "square" | "symbol" | "wide" | "circular" | "zoomable"

Input module

2/18/2021 • 4 minutes to read • Edit Online

Input controls are typically used on task pages for collecting user input, for example, for a new control.

Index

Types

- InputControl
- InputControlDesign
- InputControlMetadata
- NumberSequenceConfig

Types

InputControl

Hierarchy

Control
└ InputControl
  └ Field
  └ Lookup
  └ MultiLookup
  └ Value

Properties

NAME	SIGNATURE	DESCRIPTION
container	container: boolean (optional)	True if the control is a container. Inherited from Control.container
generic	generic: boolean (optional)	Inherited from Control.generic
getDataSource	getDataSource: function(): any	Inherited from Control.getDataSource
hidden	hidden: boolean	True if the control is hidden. Inherited from Control.hidden

Methods

NAME	SIGNATURE	DESCRIPTION
applyDesign	applyDesign(design: Design): void	Applies given design to the design on the control. Inherited from Control.applyDesign
dataContext	dataContext(): any	Inherited from Control.dataContext
getDesign	getDesign(): Design	Returns the design object of this control. Inherited from Control.getDesign

NAME	SIGNATURE	DESCRIPTION
isEditable	isEditable(): boolean	Boolean indicating if the control is editable. Inherited from Control.isEditable
metadata	metadata(): InputControlMetadata	Returns the metadata object of this control. Overrides Control.metadata
parent	parent(): Control | Page	Returns the parent (control or page) of this control. Inherited from Control.parent
root	root(): Page	Returns the root form instance (page) of this control. Inherited from Control.root

Events

NAME	SIGNATURE	DESCRIPTION
onDataChanged	onDataChanged: EventHook <null>	An event that is triggered when the input control's data changes.

InputControlDesign

Hierarchy

Design
- InputControlDesign
  - FieldDesign
  - LookupDesign
  - MultiLookupDesign
  - ValueDesign

Properties

NAME	SIGNATURE	DESCRIPTION
alignItems	alignItems: string (optional)	This property is an alias for the CSS property "align-items". Inherited from Design.alignItems
alignSelf	alignSelf: string (optional)	Inherited from Design.alignSelf
bindings	bindings: any (optional)	Inherited from Design.bindings
border	border: "none" | "solid" | "left" | "right" | "top" | "bottom" (optional)	The border behavior of a control. This property will not be inherited by the children. Inherited from Design.border
color	color: string (optional)	The foreground color of the container. Inherited from Design.color

NAME	SIGNATURE	DESCRIPTION
flexFlow	flexFlow: string (optional)	Specifying this property makes the component a flex container component. Inherited from Design.flexFlow
flexSize	flexSize: string (optional)	One number or two numbers written as a string. For example, "(size to grow) [(size-to-shrink)]" to accommodate available space in the immediate flex container. Inherited from Design.flexSize
fontSize	fontSize: "medium" | "xx-small" | "x-small" | "small" | "large" | "x-large" | "xx-large" (optional)	The proportional text size Inherited from Design.fontSize
fontWeight	fontWeight: "normal" | "bold" (optional)	Normal or bold text. Inherited from Design.fontWeight
justifyItems	justifyItems: "flex-start" | "flex-end" | "center" | "space-between" (optional)	This property is an alias for the CSS property "justify-content". Inherited from Design.justifyItems
label	label: string (optional)	Inherited from Design.label
labelPosition	labelPosition: "stacked" | "hidden" | "inline" (optional)	Determines how a label is positioned, if at all. By default, labelPosition is set to stacked. Inherited from Design.labelPosition
name	name: string (optional)	Inherited from Design.name
padding	padding: "none" | "small" | "std" (optional)	Allows specifying the component's padding behavior. Inherited from Design.padding
type	type: ControlType (optional)	The type of the control as a string. Inherited from Design.type

InputControlMetadata

Hierarchy

ControlMetadata
- InputControlMetadata
  - FieldMetadata
  - LookupMetadata
  - MultiLookupMetadata
  - ValueMetadata

Properties

NAME	SIGNATURE	DESCRIPTION

NAME	SIGNATURE	DESCRIPTION
BoundEntity	BoundEntity: string (optional)	The entity to which the control is bound. Inherited from ControlMetadata.BoundEntity
BoundField	BoundField: string (optional)	Inherited from ControlMetadata.BoundField
Description	Description: string (optional)	Description of the control. Inherited from ControlMetadata.Description
Editable	Editable: boolean (optional)	Boolean indicating if the control is editable. Inherited from ControlMetadata.Editable
ExtType	ExtType: ControlType (optional)	The extended control type. For example, a control of type Input might have an extended type of Barcode. Inherited from ControlMetadata.ExtType
HelpText	HelpText: string (optional)	The keyboard shortcut for a command. For example, "(Shift+F5)" Inherited from ControlMetadata.HelpText
Hidden	Hidden: boolean (optional)	Boolean indicating if the control is hidden or not. Inherited from ControlMetadata.Hidden
Id	Id: string (optional)	Identification string for a control. Inherited from ControlMetadata.Id
Label	Label: string (optional)	Label for a control. For example, a control representing a person's first name might have a label "First Name". Inherited from ControlMetadata.Label
Mandatory	Mandatory: boolean (optional)	If set to true then input for the control is required for the task to be completed. Mandatory controls will have a red outline.
Name	Name: string (optional)	Name of a control. Inherited from ControlMetadata.Name
NumSequence	NumSequence: NumberSequenceConfig (optional)	Used for auto detecting and changing visibility of the number sequence controls in the task or page, based on AX number sequence configuration, through extended business logic.

NAME	SIGNATURE	DESCRIPTION
Order	Order: number (optional)	Number indicating the order in which a control will appear on a page. Inherited from ControlMetadata.Order
Type	Type: ControlType (optional)	String indicating the control type. Inherited from ControlMetadata.Type

NumberSequenceConfig

Hierarchy

NumberSequenceConfig

Properties

NAME	SIGNATURE	DESCRIPTION
dataType	dataType: string	The data type is used to lookup whether the number sequence is editable or not on the reference page.
referencePageName	referencePageName: string	Page name of the page that defines if the num sequence is editable.

List module

2/18/2021 • 6 minutes to read • Edit Online

A list is a control that contains any numbers of rows. Each row follows a template for the layout of any number of controls. Lists come in two styles: simple and card.

Index

Types

Types

List

Hierarchy

ContainerControl
└ List

Properties

NAME	SIGNATURE	DESCRIPTION
$accessibility	$accessibility: any	
DefaultSearchColumn	DefaultSearchColumn: string	
container	container: boolean	True if the control is a container. Inherited from ContainerControl.container Overrides Control.container
emptyListMessage	emptyListMessage: string	Settable property to override default empty list message.
enableMultiSelect	enableMultiSelect: boolean	
generic	generic: boolean (optional)	Inherited from Control.generic
getDataSource	getDataSource: function(): any	Inherited from Control.getDataSource
hidden	hidden: boolean	True if the control is hidden. Inherited from Control.hidden
hideEmptyListMessage	hideEmptyListMessage: boolean	If true, no message is shown if the list is empty. To set this property, update the corresponding metadata property via configureControl.

NAME	SIGNATURE	DESCRIPTION
imageFields	imageFields: any [ ]	
performingRemoteSearch	performingRemoteSearch: boolean	
searchQuery	searchQuery: [value: string]: any	

Methods

NAME	SIGNATURE	DESCRIPTION
allowsNavigation	allowsNavigation(): boolean	
applyDesign	applyDesign(IDesign: ListDesign): void	Applies given design to the design on the control. Overrides Control.applyDesign
applySearch	applySearch(): void	
canPerformRemoteSearch	canPerformRemoteSearch(): boolean	
clearSearch	clearSearch(): void	
dataContext	dataContext(): any	Inherited from Control.dataContext
getColumnLabel	getColumnLabel(id: string): string	
getControl	getControl(controlName: string): Control	Given the name of a control, returns the control instance. Inherited from ContainerControl.getControl
getControlById	getControlById(id: string): Control	Given the ID of a control, returns the control instance. Inherited from ContainerControl.getControlById
getControlMetadata	getControlMetadata(controlName: string): Control	
getControlMetadataById	getControlMetadataById(id: string): Control	
getData	getData(): any [ ]	
getDesign	getDesign(): Design	Returns the design object of this control. Inherited from Control.getDesign
getListData	getListData(): any	
getRenderedRows	getRenderedRows(): Row [ ]	

NAME	SIGNATURE	DESCRIPTION
getRowNavigation	getRowNavigation(row: Row): Promise <any> | any	
getRowSelectionCount	getRowSelectionCount(): number	
getRowSelections	getRowSelections(): string [ ]	
getRowTracking	getRowTracking(row: any, index: string): string	
getSearchColumn	getSearchColumn(): string	
getSearchColumnLabel	getSearchColumnLabel(): string	
getSearchableColumns	getSearchableColumns(): any [ ]	
hideSearchBar	hideSearchBar(): boolean	
isEditable	isEditable(): boolean	Boolean indicating if the control is editable. Inherited from Control.isEditable
loadMetaData	loadMetaData(): void	
loadMore	loadMore(): void	
metadata	metadata(): ListMetadata	Returns the metadata object of this control. Overrides ContainerControl.metadata
parent	parent(): Control | Page	Returns the parent (control or page) of this control. Inherited from Control.parent
performRemoteSearch	performRemoteSearch(): void	
root	root(): Page	Returns the root form instance (page) of this control. Inherited from Control.root
selectSearchColumn	selectSearchColumn(column: string): void	
setRowSections	setRowSections(selections: string [ ]): void	

Events

NAME	SIGNATURE	DESCRIPTION
onRowCreate	onRowCreate: EventHook <Row>	
onRowSelect	onRowSelect: EventHook <Row>	

ListDesign

Hierarchy

ContainerControlDesign

└ ListDesign

Properties

NAME	SIGNATURE	DESCRIPTION
alignItems	alignItems: string (optional)	This property is an alias for the CSS property "align-items". Inherited from Design.alignItems
alignSelf	alignSelf: string (optional)	Inherited from Design.alignSelf
allowScroll	allowScroll: string (optional)	True if the container will allow scrolling when its items do not fit into the container's available space. If a container has an item which may scroll, then set this property to false to prevent nested scrolling areas. Inherited from ContainerControlDesign.allowScroll
background	background: string (optional)	The background color of the container. Inherited from ContainerControlDesign.background
bindings	bindings: any (optional)	Inherited from Design.bindings
border	border: "none" | "solid" | "left" | "right" | "top" | "bottom" (optional)	The border behavior of a control. This property will not be inherited by the children. Inherited from Design.border
color	color: string (optional)	The foreground color of the container. Inherited from Design.color
design	design: GroupDesign (optional)	The design object that will be applied to each row.
flexFlow	flexFlow: string (optional)	Specifying this property makes the component a flex container component. Inherited from Design.flexFlow
flexSize	flexSize: string (optional)	One number or two numbers written as a string. For example, "(size to grow) [(size-to-shrink)]" to accommodate available space in the immediate flex container. Inherited from Design.flexSize
fontSize	fontSize: "medium" | "xx-small" | "x-small" | "small" | "large" | "x-large" | "xx-large" (optional)	The proportional text size Inherited from Design.fontSize

NAME	SIGNATURE	DESCRIPTION
fontWeight	fontWeight: "normal" | "bold" (optional)	Normal or bold text. Inherited from Design.fontWeight
hideArrow	hideArrow: boolean (optional)	Allows an arrow (>) on a default styled navigation control to be hidden.
hideSearchBar	hideSearchBar: boolean (optional)	If true, the search bar will be hidden.
itemBorder	itemBorder: "solid" | "none" (optional)	If true, a border will appear around each row in the list. Inherited from ContainerControlDesign.itemBorder
items	items: string | Design [ ] (optional)	An array containing the components to place inside of the container. Inherited from ContainerControlDesign.items
justifyItems	justifyItems: "flex-start" | "flex-end" | "center" | "space-between" (optional)	This property is an alias for the CSS property "justify-content". Inherited from Design.justifyItems
label	label: string (optional)	Inherited from Design.label
labelPosition	labelPosition: "stacked" | "hidden" | "inline" (optional)	Determines how a label is positioned, if at all. By default, labelPosition is set to stacked. Inherited from Design.labelPosition
name	name: string (optional)	Inherited from Design.name
padding	padding: "none" | "small" | "std" (optional)	Allows specifying the component's padding behavior. Inherited from Design.padding
type	type: ControlType (optional)	The type of the control as a string. Inherited from Design.type

ListMetadata

Hierarchy

ContainerControlMetadata

└ ListMetadata

Properties

NAME	SIGNATURE	DESCRIPTION
BoundEntity	BoundEntity: string (optional)	The entity to which the control is bound. Inherited from ControlMetadata.BoundEntity
BoundField	BoundField: string (optional)	Inherited from ControlMetadata.BoundField

NAME	SIGNATURE	DESCRIPTION
Children	Children: ControlMetadata [ ] (optional)	List of metadata for controls that will appear in each row of the list.
Description	Description: string (optional)	Description of the control. Inherited from ControlMetadata.Description
DetailsPageAppId	DetailsPageAppId: string (optional)	App ID of the page that each row in the list will navigate to.
DetailsPageId	DetailsPageId: string (optional)	The ID of the page to which each row will navigate.
Editable	Editable: boolean (optional)	Boolean indicating if the control is editable. Inherited from ControlMetadata.Editable
EmptyListMessage	EmptyListMessage: string (optional)	If set, overrides the default message for empty lists.
ExtType	ExtType: ControlType (optional)	The extended control type. For example, a control of type Input might have an extended type of Barcode. Inherited from ControlMetadata.ExtType
HelpText	HelpText: string (optional)	The keyboard shortcut for a command. For example, "(Shift+F5)" Inherited from ControlMetadata.HelpText
Hidden	Hidden: boolean (optional)	Boolean indicating if the control is hidden or not. Inherited from ControlMetadata.Hidden
HideEmptyListMessage	HideEmptyListMessage: boolean (optional)	If true, the empty list message will be hidden.
HideSearchBar	HideSearchBar: boolean (optional)	If true, the search bar will be hidden.
Id	Id: string (optional)	Identification string for a control. Inherited from ControlMetadata.Id
InfiniteScroll	InfiniteScroll: boolean (optional)	If set to true then the list will allow infinite scroll.
InfiniteScrollPageSize	InfiniteScrollPageSize: number (optional)	Number of rows to load initially and the number of rows to load after the user reaches the end of the currently displayed rows.

NAME	SIGNATURE	DESCRIPTION
Label	Label: string (optional)	Label for a control. For example, a control representing a person's first name might have a label "First Name". Inherited from ControlMetadata.Label
ListStyle	ListStyle: string (optional)	Dictates the list template type.
MultiSelect	MultiSelect: boolean (optional)	If true, then the list will be a multi-select list.
Name	Name: string (optional)	Name of a control. Inherited from ControlMetadata.Name
NonEntityProjection	NonEntityProjection: boolean (optional)	
Order	Order: number (optional)	Number indicating the order in which a control will appear on a page. Inherited from ControlMetadata.Order
Type	Type: ControlType (optional)	String indicating the control type. Inherited from ControlMetadata.Type

Methods

NAME	SIGNATURE	DESCRIPTION
navigationHandler	Optional navigationHandler(row: Row): Promise <any> | NavigationArgs	A function that determines the navigation for a given row.

Events

NAME	SIGNATURE	DESCRIPTION
OnNavigate	OnNavigate: function(navigation: NavigationArgs): any (optional)	An event that is triggered when a pagelink control is selected.
OnRowSelect	OnRowSelect: function(row: Row): void (optional)	An event that is triggered when a row is selected.

Row

Hierarchy

Row

Properties

NAME	SIGNATURE	DESCRIPTION
fieldList	fieldList: Control [ ]	
headerField	headerField: Control	
hidden	hidden: boolean	If true then the row will be hidden.

NAME	SIGNATURE	DESCRIPTION
imageFields	imageFields: Control [ ]	
isSelected	isSelected: boolean	
item	item: any	A container of rendered data.
template	template: Group	Group control that represents the template for a row.

Methods

NAME	SIGNATURE	DESCRIPTION
getControl	getControl(controlName: string): Control	
getControlById	getControlById(id: string): Control	
getControlValueById	getControlValueById(id: string): string	
getRowHeader	getRowHeader(): Control	
getRowId	getRowId(): string	
hasImageField	hasImageField(): boolean	Returns true if the row has an image field.
isEntityCreatedNew	isEntityCreatedNew(): boolean	
isEntityDeleted	isEntityDeleted(): boolean	
isEntityModified	isEntityModified(): boolean	
isEntitySyncPending	isEntitySyncPending(): boolean	
select	select(): any	

Lookup module

2/18/2021 • 4 minutes to read • Edit Online

A lookup is an input control that is used to select an input from a list of options. For example, a lookup could be used to lookup a customer when linking a customer to a new sales order.

Index

Types

- Lookup
- LookupDesign
- LookupMetadata

Types

Lookup

Hierarchy

InputControl
└ Lookup

Properties

NAME	SIGNATURE	DESCRIPTION
container	container: boolean (optional)	True if the control is a container. Inherited from Control.container
generic	generic: boolean (optional)	Inherited from Control.generic
getDataSource	getDataSource: function(): any	Inherited from Control.getDataSource
hidden	hidden: boolean	True if the control is hidden. Inherited from Control.hidden

Methods

NAME	SIGNATURE	DESCRIPTION
applyDesign	applyDesign(IDesign: LookupDesign): void	Applies given design to the design on the control. Overrides Control.applyDesign
dataContext	dataContext(): any	Inherited from Control.dataContext
getDesign	getDesign(): Design	Returns the design object of this control. Inherited from Control.getDesign
getDisplayValue	getDisplayValue(): string	
getLookupPage	getLookupPage(): Page	

NAME	SIGNATURE	DESCRIPTION
getValue	getValue(): string | number	
isEditable	isEditable(): boolean	Boolean indicating if the control is editable. Inherited from Control.isEditable
metadata	metadata(): LookupMetadata	Returns the metadata object of this control. Overrides InputControl.metadata
parent	parent(): Control | Page	Returns the parent (control or page) of this control. Inherited from Control.parent
root	root(): Page	Returns the root form instance (page) of this control. Inherited from Control.root
setEntityRef	setEntityRef(newValue: string | number): Promise <any>	

Events

NAME	SIGNATURE	DESCRIPTION
onDataChanged	onDataChanged: EventHook <null>	An event that is triggered when the input control's data changes. Inherited from InputControl.onDataChanged

LookupDesign

Hierarchy

InputControlDesign
└ LookupDesign

Properties

NAME	SIGNATURE	DESCRIPTION
alignItems	alignItems: string (optional)	This property is an alias for the CSS property "align-items". Inherited from Design.alignItems
alignSelf	alignSelf: string (optional)	Inherited from Design.alignSelf
bindings	bindings: any (optional)	Inherited from Design.bindings
border	border: "none" | "solid" | "left" | "right" | "top" | "bottom" (optional)	The border behavior of a control. This property will not be inherited by the children. Inherited from Design.border
color	color: string (optional)	The foreground color of the container. Inherited from Design.color

NAME	SIGNATURE	DESCRIPTION
flexFlow	flexFlow: string (optional)	Specifying this property makes the component a flex container component. Inherited from Design.flexFlow
flexSize	flexSize: string (optional)	One number or two numbers written as a string. For example, "(size to grow) [(size-to-shrink)]" to accommodate available space in the immediate flex container. Inherited from Design.flexSize
fontSize	fontSize: "medium" | "xx-small" | "x-small" | "small" | "large" | "x-large" | "xx-large" (optional)	The proportional text size Inherited from Design.fontSize
fontWeight	fontWeight: "normal" | "bold" (optional)	Normal or bold text. Inherited from Design.fontWeight
justifyItems	justifyItems: "flex-start" | "flex-end" | "center" | "space-between" (optional)	This property is an alias for the CSS property "justify-content". Inherited from Design.justifyItems
label	label: string (optional)	Inherited from Design.label
labelPosition	labelPosition: "stacked" | "hidden" | "inline" (optional)	Determines how a label is positioned, if at all. By default, labelPosition is set to stacked. Inherited from Design.labelPosition
name	name: string (optional)	Inherited from Design.name
padding	padding: "none" | "small" | "std" (optional)	Allows specifying the component's padding behavior. Inherited from Design.padding
type	type: ControlType (optional)	The type of the control as a string. Inherited from Design.type

LookupMetadata

Hierarchy

InputControlMetadata

└ LookupMetadata

Properties

NAME	SIGNATURE	DESCRIPTION
BoundEntity	BoundEntity: string (optional)	The entity to which the control is bound. Inherited from ControlMetadata.BoundEntity
BoundField	BoundField: string (optional)	Inherited from ControlMetadata.BoundField

NAME	SIGNATURE	DESCRIPTION
Description	Description: string (optional)	Description of the control. Inherited from ControlMetadata.Description
DisplayField	DisplayField: string (optional)	The name of a control on the page, whose value should be displayed to the user. Usually, this value is user-friendly user-readable text.
DisplayKey	DisplayKey: string (optional)	
Editable	Editable: boolean (optional)	Boolean indicating if the control is editable. Inherited from ControlMetadata.Editable
ExtType	ExtType: ControlType (optional)	The extended control type. For example, a control of type Input might have an extended type of Barcode. Inherited from ControlMetadata.ExtType
FilterContext	FilterContext: DataFilter (optional)	
HelpText	HelpText: string (optional)	The keyboard shortcut for a command. For example, "(Shift+F5)" Inherited from ControlMetadata.HelpText
Hidden	Hidden: boolean (optional)	Boolean indicating if the control is hidden or not. Inherited from ControlMetadata.Hidden
Id	Id: string (optional)	Identification string for a control. Inherited from ControlMetadata.Id
Label	Label: string (optional)	Label for a control. For example, a control representing a person's first name might have a label "First Name". Inherited from ControlMetadata.Label
LookupEntity	LookupEntity: any (optional)	The entity that is being looked up in the lookup.
LookupPage	LookupPage: string (optional)	
LookupPageId	LookupPageId: string (optional)	
Mandatory	Mandatory: boolean (optional)	If set to true then input for the control is required for the task to be completed. Mandatory controls will have a red outline. Inherited from InputControlMetadata.Mandatory

NAME	SIGNATURE	DESCRIPTION
MultiSelect	MultiSelect: boolean (optional)	If true, lookup will be configured as a multi-select.
Name	Name: string (optional)	Name of a control. Inherited from ControlMetadata.Name
NumSequence	NumSequence: NumberSequenceConfig (optional)	Used for auto detecting and changing visibility of the number sequence controls in the task or page, based on AX number sequence configuration, through extended business logic. Inherited from InputControlMetadata.NumSequence
Order	Order: number (optional)	Number indicating the order in which a control will appear on a page. Inherited from ControlMetadata.Order
ReferenceAppId	ReferenceAppId: string (optional)	
ShowLookupPage	ShowLookupPage: boolean (optional)	
Type	Type: ControlType (optional)	String indicating the control type. Inherited from ControlMetadata.Type
ValueField	ValueField: string (optional)	The name of a control on the page, whose value should be used when committing the data. Usually, this value is a unique key.
ValueKey	ValueKey: string (optional)	

Events

NAME	SIGNATURE	DESCRIPTION
OnOptionSelected	OnOptionSelected: function(lookup: any, lookupEntityData: any): void (optional)	An event that is triggered by an option being selected.
OnValueChanged	OnValueChanged: function(value: any): void (optional)	An event that is triggered by a value being changed.

Multi-Lookup module

2/18/2021 • 4 minutes to read • Edit Online

Multi-Lookup controls are similar to regular lookups except they allow multiple selections at once.

Index

Types

- MultiLookup
- MultiLookupDesign
- MultiLookupMetadata

Types

MultiLookup

Hierarchy

InputControl
└ MultiLookup

Properties

NAME	SIGNATURE	DESCRIPTION
container	container: boolean (optional)	True if the control is a container. Inherited from Control.container
generic	generic: boolean (optional)	Inherited from Control.generic
getDataSource	getDataSource: function(): any	Inherited from Control.getDataSource
getEntityRefs	getEntityRefs: function(): string [ ] | number [ ]	
hidden	hidden: boolean	True if the control is hidden. Inherited from Control.hidden
setEntityRefs	setEntityRefs: function(ids: string [ ] | number [ ]): Promise <any>	

Methods

NAME	SIGNATURE	DESCRIPTION
applyDesign	applyDesign(IDesign: MultiLookupDesign): void	Applies given design to the design on the control. Overrides Control.applyDesign
dataContext	dataContext(): any	Inherited from Control.dataContext

NAME	SIGNATURE	DESCRIPTION
getDesign	getDesign(): Design	Returns the design object of this control. Inherited from Control.getDesign
getLookupPage	getLookupPage(): Page	
isEditable	isEditable(): boolean	Boolean indicating if the control is editable. Inherited from Control.isEditable
metadata	metadata(): MultiLookupMetadata	Returns the metadata object of this control. Overrides InputControl.metadata
parent	parent(): Control | Page	Returns the parent (control or page) of this control. Inherited from Control.parent
root	root(): Page	Returns the root form instance (page) of this control. Inherited from Control.root

Events

NAME	SIGNATURE	DESCRIPTION
onDataChanged	onDataChanged: EventHook <null>	An event that is triggered when the input control's data changes. Inherited from InputControl.onDataChanged

MultiLookupDesign

Hierarchy

InputControlDesign
└ MultiLookupDesign

Properties

NAME	SIGNATURE	DESCRIPTION
alignItems	alignItems: string (optional)	This property is an alias for the CSS property "align-items". Inherited from Design.alignItems
alignSelf	alignSelf: string (optional)	Inherited from Design.alignSelf
bindings	bindings: any (optional)	Inherited from Design.bindings
border	border: "none" | "solid" | "left" | "right" | "top" | "bottom" (optional)	The border behavior of a control. This property will not be inherited by the children. Inherited from Design.border
color	color: string (optional)	The foreground color of the container. Inherited from Design.color

NAME	SIGNATURE	DESCRIPTION
flexFlow	flexFlow: string (optional)	Specifying this property makes the component a flex container component. Inherited from Design.flexFlow
flexSize	flexSize: string (optional)	One number or two numbers written as a string. For example, "(size to grow) [(size-to-shrink)]" to accommodate available space in the immediate flex container. Inherited from Design.flexSize
fontSize	fontSize: "medium" | "xx-small" | "x-small" | "small" | "large" | "x-large" | "xx-large" (optional)	The proportional text size Inherited from Design.fontSize
fontWeight	fontWeight: "normal" | "bold" (optional)	Normal or bold text. Inherited from Design.fontWeight
justifyItems	justifyItems: "flex-start" | "flex-end" | "center" | "space-between" (optional)	This property is an alias for the CSS property "justify-content". Inherited from Design.justifyItems
label	label: string (optional)	Inherited from Design.label
labelPosition	labelPosition: "stacked" | "hidden" | "inline" (optional)	Determines how a label is positioned, if at all. By default, labelPosition is set to stacked. Inherited from Design.labelPosition
name	name: string (optional)	Inherited from Design.name
padding	padding: "none" | "small" | "std" (optional)	Allows specifying the component's padding behavior. Inherited from Design.padding
type	type: ControlType (optional)	The type of the control as a string. Inherited from Design.type

MultiLookupMetadata

Hierarchy

InputControlMetadata

└ MultiLookupMetadata

Properties

NAME	SIGNATURE	DESCRIPTION
BoundEntity	BoundEntity: string (optional)	The entity to which the control is bound. Inherited from ControlMetadata.BoundEntity
BoundField	BoundField: string (optional)	Inherited from ControlMetadata.BoundField

NAME	SIGNATURE	DESCRIPTION
Description	Description: string (optional)	Description of the control. Inherited from ControlMetadata.Description
Design	Design: Design (optional)	Design object for the lookup page that is referenced by the LookupPageId.
Editable	Editable: boolean (optional)	Boolean indicating if the control is editable. Inherited from ControlMetadata.Editable
ExtType	ExtType: ControlType (optional)	The extended control type. For example, a control of type Input might have an extended type of Barcode. Inherited from ControlMetadata.ExtType
FilterContext	FilterContext: DataFilter (optional)	
FilterLocalOnly	FilterLocalOnly: boolean (optional)	
HelpText	HelpText: string (optional)	The keyboard shortcut for a command. For example, "(Shift+F5)" Inherited from ControlMetadata.HelpText
Hidden	Hidden: boolean (optional)	Boolean indicating if the control is hidden or not. Inherited from ControlMetadata.Hidden
Id	Id: string (optional)	Identification string for a control. Inherited from ControlMetadata.Id
Label	Label: string (optional)	Label for a control. For example, a control representing a person's first name might have a label "First Name". Inherited from ControlMetadata.Label
LookupPageId	LookupPageId: string (optional)	Page that is hosted within the multi-lookup.
Mandatory	Mandatory: boolean (optional)	If set to true then input for the control is required for the task to be completed. Mandatory controls will have a red outline. Inherited from InputControlMetadata.Mandatory
Name	Name: string (optional)	Name of a control. Inherited from ControlMetadata.Name

NAME	SIGNATURE	DESCRIPTION
NumSequence	NumSequence: NumberSequenceConfig (optional)	Used for auto detecting and changing visibility of the number sequence controls in the task or page, based on AX number sequence configuration, through extended business logic. Inherited from InputControlMetadata.NumSequence
Order	Order: number (optional)	Number indicating the order in which a control will appear on a page. Inherited from ControlMetadata.Order
ReferenceAppId	ReferenceAppId: string (optional)	
ReverseLookupRelation	ReverseLookupRelation: boolean (optional)	
ShowPending	ShowPending: boolean (optional)	
Type	Type: ControlType (optional)	String indicating the control type. Inherited from ControlMetadata.Type

Events

NAME	SIGNATURE	DESCRIPTION
OnLookupPageCreate	OnLookupPageCreate: function(args: any, multiLookup: any): void (optional)	
OnLookupPageCreated	OnLookupPageCreated: function(args: any, multiLookup: any): void (optional)	

Page module

2/18/2021 • 6 minutes to read • Edit Online

The IPage interface encapsulates the various properties, life cycle and event hooks associated with a page in a workspace.

Page data Synchronization

Pages that can submit changes (also referred to as actions) go through various stages before they're completely in sync with the server. As soon as a page is submitted, there are three possible consequences:

- It fails client-side validation: The client logic might prevent the page from being submitted.
- The client is online: The submission is processed as soon as the application-wide sync queue is cleared.
- The client is offline: The submission is added to the application-wide sync queue and stays there as long as the client is offline.

While a submission is waiting to be synchronized, it can be in one of these states:

- Pending: The submission is still pending and can still be edited.
- Processing: The submission is currently being synchronized. In this state, any further edits are not allowed on the page.

After a submission goes through to the server, it can be in one of these states:

- Synchronized: The submission is accepted by the server and is synchronized.
- Error: The server rejects the submission and the page enters an error state.

Multiple pending submissions for a given page and context might be clubbed (and sent) together if the page is deemed to be idempotent. This implies that the server need not process all submissions in any order and depends on the design of the page on the server.

Index

Enumerations

- PageState

Types

- CompleteEventArgs
- Design
- NavigationArgs
- Page
- PageMetadata
- PageOptions
- PageSubmitArgs
- PageTarget

Enumerations

PageState

Enumeration members

NAME	SIGNATURE	DESCRIPTION
error	error: 10	The page is currently in the error state, but can be refreshed in an attempt to get out of this state.
loaded	loaded: 3	The page is fully loaded and can be refreshed and, if possible, submitted.
loading	loading: 2	The page is currently being loaded.
offline	offline: 1	The page was loaded in the offline mode, thus not refreshable.
refreshing	refreshing: 4	The page is currently refreshing its data.

Types

CompleteEventArgs

Hierarchy

CompleteEventArgs

Properties

NAME	SIGNATURE	DESCRIPTION
error	error: boolean (optional)	
navigation	navigation: NavigationArgs (optional)	
processed	processed: boolean (optional)	

Design

Hierarchy

Design
- PageLinkDesign
- ContainerControlDesign
- InputControlDesign
- ImageDesign

Properties

NAME	SIGNATURE	DESCRIPTION
alignItems	alignItems: string (optional)	This property is an alias for the CSS property "align-items".
alignSelf	alignSelf: string (optional)	
bindings	bindings: any (optional)	

NAME	SIGNATURE	DESCRIPTION
border	border: "none" | "solid" | "left" | "right" | "top" | "bottom" (optional)	The border behavior of a control. This property will not be inherited by the children.
color	color: string (optional)	The foreground color of the container.
flexFlow	flexFlow: string (optional)	Specifying this property makes the component a flex container component.
flexSize	flexSize: string (optional)	One number or two numbers written as a string. For example, "(size to grow) [(size-to-shrink)]" to accommodate available space in the immediate flex container.
fontSize	fontSize: "medium" | "xx-small" | "x-small" | "small" | "large" | "x-large" | "xx-large" (optional)	The proportional text size
fontWeight	fontWeight: "normal" | "bold" (optional)	Normal or bold text.
justifyItems	justifyItems: "flex-start" | "flex-end" | "center" | "space-between" (optional)	This property is an alias for the CSS property "justify-content".
label	label: string (optional)	
labelPosition	labelPosition: "stacked" | "hidden" | "inline" (optional)	Determines how a label is positioned, if at all. By default, labelPosition is set to stacked.
name	name: string (optional)	
padding	padding: "none" | "small" | "std" (optional)	Allows specifying the component's padding behavior.
type	type: ControlType (optional)	The type of the control as a string.

NavigationArgs

Hierarchy

PageTarget

└ NavigationArgs

Properties

NAME	SIGNATURE	DESCRIPTION
label	label: string (optional)	
options	options: any (optional)	
params	params: PageOptions (optional)	Inherited from PageTarget.params

NAME	SIGNATURE	DESCRIPTION
replace	replace: boolean (optional)	If set to true, removes current view firing navigation from navigation history stack.
to	to: string (optional)	Inherited from PageTarget.to
url	url: string (optional)	If provided, this link is directly opened.

Page

Hierarchy

Page

Properties

NAME	SIGNATURE	DESCRIPTION
children	children: Control [ ]	The list of all direct children controls of the page.
dataLoadedInitially	dataLoadedInitially: Promise <void>	A promise which resolves when the data has loaded for the first time.
initialized	initialized: boolean	True if the page instance has been initialized.
metadata	metadata: PageMetadata	The page metadata.
metadataLoaded	metadataLoaded: Promise <void>	A promise which resolves when the metadata has finished loading.
pageContext	pageContext: string	The current page context.
pageFilter	pageFilter: DataFilter	The current filter applied on the page.
state	state: PageState	The current state of the page.
syncError	syncError: boolean	True if the page's submission is in error state. This normally happens when the server rejects submissions due to validation errors.
syncPending	syncPending: boolean	True if the page's submission is waiting to be synced.
syncProcessing	syncProcessing: boolean	True if the page instance is currently syncing its submission.
syncUnitEditable	syncUnitEditable: boolean	True if it's possible to edit a submission while it's waiting to be synchronized.
title	title: string	The title of the page.

Methods

NAME	SIGNATURE	DESCRIPTION
canSubmit	canSubmit(): boolean	Returns true if action page can be submitted and there are no validation error messages.
close	close(): void	Dispose the page instance and all its lifecycle events.
getAction	getAction(actionName: string): PageLink	Get a page action by name.
getActions	getActions(): PageLink [ ]	Get all page actions.
getControl	getControl(controlName: string): Control	Get a page control by name.
getDesign	getDesign(): Design	Get the design object associated with the page.
getEntityContext	getEntityContext(): EntityRef	Get current entity context.
isEditable	isEditable(): boolean	Returns true if the page is an Action Page
refreshData	refreshData(): Promise <void>	Force refresh page data.
resume	resume(): Promise <void>	Resume a temporarily suspended page.
submit	submit(): Promise <CompleteEventArgs>	Submit an Action.
suspend	suspend(): void	Temporarily suspend a page.

Events

NAME	SIGNATURE	DESCRIPTION
onClose	onClose: EventHook <null>	Event that is raised when a page is closed.
onComplete	onComplete: EventHook <any>	Event that is raised when an action is completed.
onDataLoaded	onDataLoaded: EventHook <any>	Event that fires when the page data has loaded.
onInit	onInit: EventHook <any>	Event that fires when a page instance has been initialized, and the metadata has been loaded.
onPreInit	onPreInit: EventHook <any>	Event that fires when a page instance has been initialized.

NAME	SIGNATURE	DESCRIPTION
onRefresh	onRefresh: EventHook <null>	Event that fires on forced page refresh, before new data has been loaded.
onStateChange	onStateChange: EventHook <null>	Event that fires when the page state changes.
onSubmit	onSubmit: EventHook <PageSubmitArgs>	Event that fires before an action is submitted. It can be intercepted for action validation deferring
onSyncStatusChange	onSyncStatusChange: EventHook <null>	Event that fires when the page sync status changes.

PageMetadata

Hierarchy

PageMetadata

Properties

NAME	SIGNATURE	DESCRIPTION
Controls	Controls: ControlMetadata [ ] (optional)	
Design	Design: Design (optional)	
Id	Id: string (optional)	
QuickSubmit	QuickSubmit: boolean (optional)	
SourcePageId	SourcePageId: string (optional)	
SubmitButtonDesign	SubmitButtonDesign: Design (optional)	
Tasks	Tasks: PageMetadata [ ] (optional)	
Title	Title: string (optional)	

Events

NAME	SIGNATURE	DESCRIPTION
OnDataLoaded	OnDataLoaded: function(sender: Page, dataWrapper: any): void (optional)	
OnInit	OnInit: function(sender: Page): void (optional)	
OnPreInit	OnPreInit: function(sender: Page): void (optional)	
OnSubmit	OnSubmit: function(dataValues: any, args: any): void (optional)	

NAME	SIGNATURE	DESCRIPTION
OnTaskSubmitted	OnTaskSubmitted: function(taskHandle: any, taskOptions: any): any (optional)	
OnTaskSubmitting	OnTaskSubmitting: function(taskOptions: any): any (optional)	

PageOptions

Hierarchy

PageOptions

Properties

NAME	SIGNATURE	DESCRIPTION
appId	appId: string (optional)	
design	design: Design (optional)	
excludeContext	excludeContext: boolean (optional)	
filter	filter: DataFilter (optional)	
filterLocalOnly	filterLocalOnly: boolean (optional)	
pageContext	pageContext: string (optional)	
pageId	pageId: string (optional)	
readOptions	readOptions: IReadOptions (optional)	

PageSubmitArgs

Hierarchy

PageSubmitArgs

Properties

NAME	SIGNATURE	DESCRIPTION
dataValues	dataValues: any	Get the payload of the submit action.
sender	sender: Page	Get the sender page instance of the submit action.

Methods

NAME	SIGNATURE	DESCRIPTION
addMessage	addMessage(message: string, type: any): any	Add a validation error message to be displayed.
cancel	cancel(): any	Prevent the action from submitting.

NAME	SIGNATURE	DESCRIPTION
getMessages	getMessages(): string [ ]	Get all previously added messages
isCancelled	isCancelled(): boolean	Check if the submit action is cancelled.
wait	wait(promise: Promise <any>): any	Wait on a given promise before continuing with the submission.

PageTarget

Hierarchy

PageTarget
└ NavigationArgs

Properties

NAME	SIGNATURE	DESCRIPTION
params	params: PageOptions (optional)	
to	to: string (optional)	

Pagelink module

2/18/2021 • 4 minutes to read • Edit Online

A pagelink is a control that navigates to another page.

Index

Types

- PageLink
- PageLinkDesign
- PageLinkMetadata

Types

PageLink

Hierarchy

Control
└ PageLink

Properties

NAME	SIGNATURE	DESCRIPTION
container	container: boolean (optional)	True if the control is a container. Inherited from Control.container
generic	generic: boolean (optional)	Inherited from Control.generic
getDataSource	getDataSource: function(): any	Inherited from Control.getDataSource
hidden	hidden: boolean	True if the control is hidden. Inherited from Control.hidden

Methods

NAME	SIGNATURE	DESCRIPTION
allowsNavigation	allowsNavigation(): boolean	
applyDesign	applyDesign(design: PageLinkDesign): void	Applies given design to the design on the control. Overrides Control.applyDesign
dataContext	dataContext(): any	Inherited from Control.dataContext
getCount	getCount(): number | string	
getDesign	getDesign(): Design	Returns the design object of this control. Inherited from Control.getDesign

NAME	SIGNATURE	DESCRIPTION
getNavigationHandler	getNavigationHandler(): NavigationArgs	
isEditable	isEditable(): boolean	Boolean indicating if the control is editable. Inherited from Control.isEditable
metadata	metadata(): PageLinkMetadata	Returns the metadata object of this control. Overrides Control.metadata
parent	parent(): Control | Page	Returns the parent (control or page) of this control. Inherited from Control.parent
root	root(): Page	Returns the root form instance (page) of this control. Inherited from Control.root
showCount	showCount(): boolean	

PageLinkDesign

Hierarchy

Design

└ PageLinkDesign

Properties

NAME	SIGNATURE	DESCRIPTION
alignItems	alignItems: string (optional)	This property is an alias for the CSS property "align-items". Inherited from Design.alignItems
alignSelf	alignSelf: string (optional)	Inherited from Design.alignSelf
background	background: string (optional)	Sets the background color.
bindings	bindings: any (optional)	Inherited from Design.bindings
border	border: "none" | "solid" | "left" | "right" | "top" | "bottom" (optional)	The border behavior of a control. This property will not be inherited by the children. Inherited from Design.border
color	color: string (optional)	The foreground color of the container. Inherited from Design.color
excludeContext	excludeContext: boolean (optional)	
flexFlow	flexFlow: string (optional)	Specifying this property makes the component a flex container component. Inherited from Design.flexFlow

NAME	SIGNATURE	DESCRIPTION
flexSize	flexSize: string (optional)	One number or two numbers written as a string. For example, "(size to grow) [(size-to-shrink)]" to accommodate available space in the immediate flex container. Inherited from Design.flexSize
fontSize	fontSize: "medium" | "xx-small" | "x-small" | "small" | "large" | "x-large" | "xx-large" (optional)	The proportional text size Inherited from Design.fontSize
fontWeight	fontWeight: "normal" | "bold" (optional)	Normal or bold text. Inherited from Design.fontWeight
hideArrow	hideArrow: boolean (optional)	Allows an arrow (>) on a default styled navigation control to be hidden.
icon	icon: string (optional)	Name of the icon that is displayed in the pagelink control.
justifyItems	justifyItems: "flex-start" | "flex-end" | "center" | "space-between" (optional)	This property is an alias for the CSS property "justify-content". Inherited from Design.justifyItems
label	label: string (optional)	Inherited from Design.label
labelPosition	labelPosition: "stacked" | "hidden" | "inline" (optional)	Determines how a label is positioned, if at all. By default, labelPosition is set to stacked. Inherited from Design.labelPosition
name	name: string (optional)	Inherited from Design.name
navigation	navigation: NavigationArgs (optional)	Navigation object of the pagelink.
padding	padding: "none" | "small" | "std" (optional)	Allows specifying the component's padding behavior. Inherited from Design.padding
showCount	showCount: boolean (optional)	If true, shows a count of the records present in the list on the target page.
style	style: string (optional)	Determines the visual style of the pagelink control. Options: * "inline": takes up the full width its container, with the label in-line with the icon * "button": takes up only as much width as needed by the label, with the label below the icon
type	type: ControlType (optional)	The type of the control as a string. Inherited from Design.type

PageLinkMetadata

Hierarchy

ControlMetadata
└ PageLinkMetadata

Properties

NAME	SIGNATURE	DESCRIPTION
BoundEntity	BoundEntity: string (optional)	The entity to which the control is bound. Inherited from ControlMetadata.BoundEntity
BoundField	BoundField: string (optional)	Inherited from ControlMetadata.BoundField
Description	Description: string (optional)	Description of the control. Inherited from ControlMetadata.Description
Editable	Editable: boolean (optional)	Boolean indicating if the control is editable. Inherited from ControlMetadata.Editable
ExcludeContext	ExcludeContext: boolean (optional)	
ExtType	ExtType: ControlType (optional)	The extended control type. For example, a control of type Input might have an extended type of Barcode. Inherited from ControlMetadata.ExtType
HelpText	HelpText: string (optional)	The keyboard shortcut for a command. For example, "(Shift+F5)" Inherited from ControlMetadata.HelpText
Hidden	Hidden: boolean (optional)	Boolean indicating if the control is hidden or not. Inherited from ControlMetadata.Hidden
Icon	Icon: string (optional)	Name of the icon that is displayed in the pagelink control.
IconSize	IconSize: number (optional)	Determines the size of the icon that is displayed in the pagelink control.
Id	Id: string (optional)	Identification string for a control. Inherited from ControlMetadata.Id
Label	Label: string (optional)	Label for a control. For example, a control representing a person's first name might have a label "First Name". Inherited from ControlMetadata.Label
Name	Name: string (optional)	Name of a control. Inherited from ControlMetadata.Name

NAME	SIGNATURE	DESCRIPTION
Navigation	Navigation: NavigationArgs (optional)	Navigation object of the pagelink.
Order	Order: number (optional)	Number indicating the order in which a control will appear on a page. Inherited from ControlMetadata.Order
ShowCount	ShowCount: boolean (optional)	If true, shows a count of the records present in the list on the target page.
Style	Style: string (optional)	Determines the visual style of the pagelink control.
Target	Target: string (optional)	Name of the target action or page to navigate to when the pagelink is selected.
Type	Type: ControlType (optional)	String indicating the control type. Inherited from ControlMetadata.Type
UseDataContext	UseDataContext: boolean (optional)	

Events

NAME	SIGNATURE	DESCRIPTION
OnNavigate	OnNavigate: function(navigation: NavigationArgs | string): any (optional)	An event that is triggered when the navigation is triggered.

Part module

2/18/2021 • 4 minutes to read • Edit Online

A part is a container control that contains only a page, allowing for a page to be embedded within a page.

Index

Types

- Part
- PartDesign
- PartMetadata

Types

Part

Hierarchy

ContainerControl
└ Part

Properties

NAME	SIGNATURE	DESCRIPTION
container	container: boolean	True if the control is a container. Inherited from ContainerControl.container Overrides Control.container
generic	generic: boolean (optional)	Inherited from Control.generic
getDataSource	getDataSource: function(): any	Inherited from Control.getDataSource
hidden	hidden: boolean	True if the control is hidden. Inherited from Control.hidden

Methods

NAME	SIGNATURE	DESCRIPTION
applyDesign	applyDesign(IDesign: PartDesign): void	Applies given design to the design on the control. Overrides Control.applyDesign
dataContext	dataContext(): any	Inherited from Control.dataContext
getControl	getControl(controlName: string): Control	Given the name of a control, returns the control instance. Inherited from ContainerControl.getControl

NAME	SIGNATURE	DESCRIPTION
getControlById	getControlById(id: string): Control	Given the ID of a control, returns the control instance. Inherited from ContainerControl.getControlById
getDesign	getDesign(): Design	Returns the design object of this control. Inherited from Control.getDesign
getEntityRef	getEntityRef(): string	Gets value of entityRef binding to control.
getPartPage	getPartPage(): Page	Gets the page of the part.
hasTarget	hasTarget(): boolean	Returns true if the part has a target page.
isEditable	isEditable(): boolean	Boolean indicating if the control is editable. Inherited from Control.isEditable
metadata	metadata(): PartMetadata	Returns the metadata object of this control. Overrides ContainerControl.metadata
parent	parent(): Control | Page	Returns the parent (control or page) of this control. Inherited from Control.parent
root	root(): Page	Returns the root form instance (page) of this control. Inherited from Control.root

PartDesign

Hierarchy

ContainerControlDesign

└ PartDesign

Properties

NAME	SIGNATURE	DESCRIPTION
alignItems	alignItems: string (optional)	This property is an alias for the CSS property "align-items". Inherited from Design.alignItems
alignSelf	alignSelf: string (optional)	Inherited from Design.alignSelf

NAME	SIGNATURE	DESCRIPTION
allowScroll	allowScroll: string (optional)	True if the container will allow scrolling when its items do not fit into the container's available space. If a container has an item which may scroll, then set this property to false to prevent nested scrolling areas. Inherited from ContainerControlDesign.allowScroll
background	background: string (optional)	The background color of the container. Inherited from ContainerControlDesign.background
bindings	bindings: any (optional)	Inherited from Design.bindings
border	border: "none" | "solid" | "left" | "right" | "top" | "bottom" (optional)	The border behavior of a control. This property will not be inherited by the children. Inherited from Design.border
color	color: string (optional)	The foreground color of the container. Inherited from Design.color
design	design: PartDesign (optional)	Design for the target page.
flexFlow	flexFlow: string (optional)	Specifying this property makes the component a flex container component. Inherited from Design.flexFlow
flexSize	flexSize: string (optional)	One number or two numbers written as a string. For example, "(size to grow) [(size-to-shrink)]" to accommodate available space in the immediate flex container. Inherited from Design.flexSize
fontSize	fontSize: "medium" | "xx-small" | "x-small" | "small" | "large" | "x-large" | "xx-large" (optional)	The proportional text size Inherited from Design.fontSize
fontWeight	fontWeight: "normal" | "bold" (optional)	Normal or bold text. Inherited from Design.fontWeight
itemBorder	itemBorder: "solid" | "none" (optional)	If true, a border will appear around each row in the list. Inherited from ContainerControlDesign.itemBorder
items	items: string | Design [ ] (optional)	An array containing the components to place inside of the container. Inherited from ContainerControlDesign.items

NAME	SIGNATURE	DESCRIPTION
justifyItems	justifyItems: "flex-start" | "flex-end" | "center" | "space-between" (optional)	This property is an alias for the CSS property "justify-content". Inherited from Design.justifyItems
label	label: string (optional)	Inherited from Design.label
labelPosition	labelPosition: "stacked" | "hidden" | "inline" (optional)	Determines how a label is positioned, if at all. By default, labelPosition is set to stacked. Inherited from Design.labelPosition
name	name: string (optional)	Inherited from Design.name
padding	padding: "none" | "small" | "std" (optional)	Allows specifying the component's padding behavior. Inherited from Design.padding
target	target: PageTarget (optional)	Target page of the part.
type	type: ControlType (optional)	The type of the control as a string. Inherited from Design.type

PartMetadata

Hierarchy

ContainerControlMetadata

└ PartMetadata

Properties

NAME	SIGNATURE	DESCRIPTION
BoundEntity	BoundEntity: string (optional)	The entity to which the control is bound. Inherited from ControlMetadata.BoundEntity
BoundField	BoundField: string (optional)	Inherited from ControlMetadata.BoundField
Description	Description: string (optional)	Description of the control. Inherited from ControlMetadata.Description
Design	Design: PartDesign (optional)	Design for the target page.
Editable	Editable: boolean (optional)	Boolean indicating if the control is editable. Inherited from ControlMetadata.Editable
ExtType	ExtType: ControlType (optional)	The extended control type. For example, a control of type Input might have an extended type of Barcode. Inherited from ControlMetadata.ExtType

NAME	SIGNATURE	DESCRIPTION
HelpText	HelpText: string (optional)	The keyboard shortcut for a command. For example, "(Shift+F5)" Inherited from ControlMetadata.HelpText
Hidden	Hidden: boolean (optional)	Boolean indicating if the control is hidden or not. Inherited from ControlMetadata.Hidden
Id	Id: string (optional)	Identification string for a control. Inherited from ControlMetadata.Id
Label	Label: string (optional)	Label for a control. For example, a control representing a person's first name might have a label "First Name". Inherited from ControlMetadata.Label
Name	Name: string (optional)	Name of a control. Inherited from ControlMetadata.Name
Order	Order: number (optional)	Number indicating the order in which a control will appear on a page. Inherited from ControlMetadata.Order
Target	Target: PageTarget (optional)	Target page of the part.
Type	Type: ControlType (optional)	String indicating the control type. Inherited from ControlMetadata.Type

Services module

2/18/2021 • 2 minutes to read • Edit Online

Various services that are available to the application in client runtime.

Index

Types

- AsyncService
- CacheService
- DataService
- MetadataService
- PageData

Type aliases

- ExpressionOperator

Types

AsyncService

Hierarchy

AsyncService

Methods

NAME	SIGNATURE	DESCRIPTION
all	all(...args: any [ ]): Promise <any [ ]>	
defer	defer <T>(): Deferred <T>	Creates a deferred object which can be used to return a promise from event handlers (where applicable) and resolve reject them asynchronously.

CacheService

Hierarchy

CacheService

Methods

NAME	SIGNATURE	DESCRIPTION
getData	getData(cacheKey: string): any	
setData	setData(cacheKey: string, data: any): any	

DataService

Hierarchy

DataService

Methods

NAME	SIGNATURE	DESCRIPTION
findEntityData	findEntityData(entityType: any, propertyName: string, propertyValue: any, includeChanges?: boolean): any	
getEntityData	getEntityData(entityType: any, entityId: string): any	
getPageData	getPageData(pageId: string, context: any, filter: any, allowedStaleness: number): Promise <PageData>	

MetadataService

Hierarchy

MetadataService

Properties

NAME	SIGNATURE	DESCRIPTION
version	version: string	Gets the version of the platform currently running.

Methods

NAME	SIGNATURE	DESCRIPTION
addControl	addControl(componentName: string, controlName: string, controlType: ControlType, parentContainerName?: string, options?: ControlMetadata): any	
compareVersion	compareVersion(versionToCompare: string): 1 | -1	Compares the current platform version with a reference version.
configureAction	configureAction(actionName: string, options: PageMetadata): any	Configuring an action allows specifying or overriding certain behaviors specific to actions.
configureControl	configureControl(componentName: string, controlName: string, options: ControlMetadata): any	Configuring a control allows specifying or overriding certain behaviors specific to the control. Note that the available behaviors vary by control type.
configureEntity	configureEntity(entityName: string, options: any): any	Configuring an entity allows specifying or overriding certain behaviors specific to the entity.
configureLookup	configureLookup(taskName: string, lookupControlName: string, options: LookupMetadata): any	Configures a field on an action to behave as a lookup. Requires using an existing page which contains a list control.
configurePage	configurePage(pageName: string, options: PageMetadata): any	Configuring a Page allows specifying or overriding certain behaviors specific to the Page.

NAME	SIGNATURE	DESCRIPTION
configureWorkspace	configureWorkspace(options: PageMetadata): any	Configuring a workspace allows specifying or overriding certain behaviors specific to the workspace.
findAction	findAction(actionName: string): PageMetadata	Gets a copy of the current metadata instance of a specified Action, for the purpose of inspecting the metadata (not to be used for changing the metadata).
findControl	findControl(componentMetadata: any, controlName: string): ControlMetadata	Gets a copy of the current metadata instance of a specified control, for the purpose of inspecting the metadata (not to be used for changing the metadata).
findPage	findPage(pageName: string): PageMetadata	Gets a copy of the current metadata instance of a specified page, for the purpose of inspecting the metadata (not to be used for changing the metadata).
getFilterExpression	getFilterExpression(pageName: string, listControlName: string, controlName: string, operator: ExpressionOperator, value: string): DataFilter	Create a DataFilter object for a list control based on the provided options.
getFormReference	getFormReference(componentName: string, filterContext: DataFilter, excludeContext: boolean, filterLocalOnly?: boolean): NavigationArgs	Create an INavigationArgs object for a specific page action to be used with a navigation control.
hideNavigation	hideNavigation(pageNamesToHide: string [ ]): any	Hides the specified page(s) from the default landing page.

PageData

Hierarchy

PageData

Methods

NAME	SIGNATURE	DESCRIPTION
getControlValue	getControlValue(controlName: string): any	Gets the value of a control directly from the data set loaded in the page.
setControlValue	setControlValue(controlName: string, value: any): any	Sets the value of a control directly into the data set loaded in the page.

Type aliases

ExpressionOperator

ExpressionOperator: "Is" | "IsNot" | "Contains" | "BeginsWith" | "EndsWith" | "GreaterThan" | "LessThan" | "GreaterThanOrEqual" | "LessThanOrEqual"

Represents possible values for the expression operator used in defining filters and in other places

Value module

2/18/2021 • 5 minutes to read • Edit Online

This is the base class for single value controls.

Index

Types

- GenericValue
- Value
- ValueDesign
- ValueMetadata

Types

GenericValue

Hierarchy

Value
└ GenericValue

Properties

NAME	SIGNATURE	DESCRIPTION
container	container: boolean (optional)	True if the control is a container. Inherited from Control.container
generic	generic: boolean	True if the control is a generic. Overrides Control.generic
getDataSource	getDataSource: function(): any	Inherited from Control.getDataSource
hidden	hidden: boolean	True if the control is hidden. Inherited from Control.hidden

Methods

NAME	SIGNATURE	DESCRIPTION
applyDesign	applyDesign(design: Design): void	Applies given design to the design on the control. Inherited from Control.applyDesign
dataContext	dataContext(): any	Inherited from Control.dataContext
getDesign	getDesign(): Design	Returns the design object of this control. Inherited from Control.getDesign
getValue	getValue(): string	Returns the value of the control. Inherited from Value.getValue

NAME	SIGNATURE	DESCRIPTION
isEditable	isEditable(): boolean	Boolean indicating if the control is editable. Inherited from Control.isEditable
metadata	metadata(): ValueMetadata	Returns the metadata object of this control. Inherited from Value.metadata Overrides InputControl.metadata
parent	parent(): Control | Page	Returns the parent (control or page) of this control. Inherited from Control.parent
root	root(): Page	Returns the root form instance (page) of this control. Inherited from Control.root
setValue	setValue(value: string): void	Sets the value of the control. Inherited from Value.setValue

Events

NAME	SIGNATURE	DESCRIPTION
onDataChanged	onDataChanged: EventHook <null>	An event that is triggered when the input control's data changes. Inherited from InputControl.onDataChanged

Value

Hierarchy

InputControl
- Value
  - FileUploader
  - HyperLink
  - GenericValue

Properties

NAME	SIGNATURE	DESCRIPTION
container	container: boolean (optional)	True if the control is a container. Inherited from Control.container
generic	generic: boolean (optional)	Inherited from Control.generic
getDataSource	getDataSource: function(): any	Inherited from Control.getDataSource
hidden	hidden: boolean	True if the control is hidden. Inherited from Control.hidden

Methods

NAME	SIGNATURE	DESCRIPTION
applyDesign	applyDesign(design: Design): void	Applies given design to the design on the control. Inherited from Control.applyDesign
dataContext	dataContext(): any	Inherited from Control.dataContext
getDesign	getDesign(): Design	Returns the design object of this control. Inherited from Control.getDesign
getValue	getValue(): string	Returns the value of the control.
isEditable	isEditable(): boolean	Boolean indicating if the control is editable. Inherited from Control.isEditable
metadata	metadata(): ValueMetadata	Returns the metadata object of this control. Overrides InputControl.metadata
parent	parent(): Control | Page	Returns the parent (control or page) of this control. Inherited from Control.parent
root	root(): Page	Returns the root form instance (page) of this control. Inherited from Control.root
setValue	setValue(value: string): void	Sets the value of the control.

Events

NAME	SIGNATURE	DESCRIPTION
onDataChanged	onDataChanged: EventHook <null>	An event that is triggered when the input control's data changes. Inherited from InputControl.onDataChanged

ValueDesign

Hierarchy

InputControlDesign
  └ ValueDesign
    └ FileUploaderDesign
    └ HyperLinkDesign

Properties

NAME	SIGNATURE	DESCRIPTION
alignItems	alignItems: string (optional)	This property is an alias for the CSS property "align-items". Inherited from Design.alignItems
alignSelf	alignSelf: string (optional)	Inherited from Design.alignSelf

NAME	SIGNATURE	DESCRIPTION
bindings	bindings: any (optional)	Inherited from Design.bindings
border	border: "none" | "solid" | "left" | "right" | "top" | "bottom" (optional)	The border behavior of a control. This property will not be inherited by the children. Inherited from Design.border
color	color: string (optional)	The foreground color of the container. Inherited from Design.color
flexFlow	flexFlow: string (optional)	Specifying this property makes the component a flex container component. Inherited from Design.flexFlow
flexSize	flexSize: string (optional)	One number or two numbers written as a string. For example, "(size to grow) [(size-to-shrink)]" to accommodate available space in the immediate flex container. Inherited from Design.flexSize
fontSize	fontSize: "medium" | "xx-small" | "x-small" | "small" | "large" | "x-large" | "xx-large" (optional)	The proportional text size Inherited from Design.fontSize
fontWeight	fontWeight: "normal" | "bold" (optional)	Normal or bold text. Inherited from Design.fontWeight
justifyItems	justifyItems: "flex-start" | "flex-end" | "center" | "space-between" (optional)	This property is an alias for the CSS property "justify-content". Inherited from Design.justifyItems
label	label: string (optional)	Inherited from Design.label
labelPosition	labelPosition: "stacked" | "hidden" | "inline" (optional)	Determines how a label is positioned, if at all. By default, labelPosition is set to stacked. Inherited from Design.labelPosition
name	name: string (optional)	Inherited from Design.name
padding	padding: "none" | "small" | "std" (optional)	Allows specifying the component's padding behavior. Inherited from Design.padding
type	type: ControlType (optional)	The type of the control as a string. Inherited from Design.type

ValueMetadata

Hierarchy

InputControlMetadata

└ ValueMetadata

└ FileUploaderMetadata
└ HyperLinkMetadata

Properties

NAME	SIGNATURE	DESCRIPTION
BoundEntity	BoundEntity: string (optional)	The entity to which the control is bound. Inherited from ControlMetadata.BoundEntity
BoundField	BoundField: string (optional)	Inherited from ControlMetadata.BoundField
Description	Description: string (optional)	Description of the control. Inherited from ControlMetadata.Description
Editable	Editable: boolean (optional)	Boolean indicating if the control is editable. Inherited from ControlMetadata.Editable
ExtType	ExtType: ControlType (optional)	The extended control type. For example, a control of type Input might have an extended type of Barcode. Inherited from ControlMetadata.ExtType
HelpText	HelpText: string (optional)	The keyboard shortcut for a command. For example, "(Shift+F5)" Inherited from ControlMetadata.HelpText
Hidden	Hidden: boolean (optional)	Boolean indicating if the control is hidden or not. Inherited from ControlMetadata.Hidden
Id	Id: string (optional)	Identification string for a control. Inherited from ControlMetadata.Id
Label	Label: string (optional)	Label for a control. For example, a control representing a person's first name might have a label "First Name". Inherited from ControlMetadata.Label
Mandatory	Mandatory: boolean (optional)	If set to true then input for the control is required for the task to be completed. Mandatory controls will have a red outline. Inherited from InputControlMetadata.Mandatory
Name	Name: string (optional)	Name of a control. Inherited from ControlMetadata.Name

NAME	SIGNATURE	DESCRIPTION
NumSequence	NumSequence: NumberSequenceConfig (optional)	Used for auto detecting and changing visibility of the number sequence controls in the task or page, based on AX number sequence configuration, through extended business logic. Inherited from InputControlMetadata.NumSequence
Order	Order: number (optional)	Number indicating the order in which a control will appear on a page. Inherited from ControlMetadata.Order
Type	Type: ControlType (optional)	String indicating the control type. Inherited from ControlMetadata.Type

Application type

2/18/2021 • 2 minutes to read • Edit Online

Represents a runtime instance of an application.

Hierarchy

Application

Index

Properties

- minVersion

Methods

- appInit

Properties

minVersion

minVersion: string (optional)

An optional marker to indicate the minimum platform version required by this component. When this is specified and the component is attempted to be loaded in an older version of the platform, the corresponding workspace is not loaded and user is directed to install a newer version of the platform.

Methods

appInit

appInit(metadata: ApplicationMetadata): any

This method is invoked at the point the application is about to run, with the instance of the application metadata loaded. The metadata passed in can be still modified to change behaviors before this method returns.

Parameters

NAME	TYPE	DESCRIPTION
metadata	ApplicationMetadata	

Returns any

ApplicationMetadata type

2/18/2021 • 2 minutes to read • Edit Online

Represents the declarative metadata of an application

Hierarchy

ApplicationMetadata

Index

Properties

Properties

ColorName

ColorName: string (optional)

The theme color of the application

Configs

Configs: [name: string]: any (optional)

An application can have a set of named config supplied by the author or the resource provider

Description

Description: string (optional)

The description of the application

IconName

IconName: string (optional)

The representative icon of the application

Id

Id: string

The unique identifier of the application

Title

Title: string

The title of the application

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

AsyncService type

2/18/2021 • 2 minutes to read • Edit Online

Provides ability to perform async operations from business logic code.

Hierarchy

AsyncService

Index

Method list

- all
- defer

Methods

all

```
all(...args: any [ ]): Promise <any [ ]>
```

Parameters

NAME	TYPE	DESCRIPTION
...args	any [ ]	

Returns Promise <any [ ]>

defer

```
defer <>(): [Deferred](defer-ideferred.md) <>
```

Creates a deferred object that can be used to return a promise from event handlers (where applicable) and resolve/reject them asynchronously.

Returns Deferred <T>

CacheService type

2/18/2021 • 2 minutes to read • Edit Online

Provides ability to access data from the device cache and update data into the device cache.

Hierarchy

CacheService

Index

Methods

- getData
- setData

Methods

getData

getData(cacheKey: string): any

Parameters

NAME	TYPE	DESCRIPTION
cacheKey	string	

Returns any

setData

setData(cacheKey: string, data: any): any

Parameters

NAME	TYPE	DESCRIPTION
cacheKey	string	
data	any	

Returns any

CompleteEventArgs type

2/18/2021 • 2 minutes to read • Edit Online

Hierarchy

CompleteEventArgs

Index

Properties

- error
- navigation
- processed

Properties

error

error: boolean (optional)

navigation

navigation: NavigationArgs (optional)

processed

processed: boolean (optional)

ContainerControl type

2/18/2021 • 2 minutes to read • Edit Online

Container control interface with methods and attributes for all container controls. A container control can contain any number of controls.

Hierarchy

Control
  └─ ContainerControl
    └─ Group
    └─ List
    └─ Part

Index

Properties

- container
- generic
- getDataSource
- hidden

Methods

- applyDesign
- dataContext
- getControl
- getControlById
- getDesign
- isEditable
- metadata
- parent
- root

Properties

container

container: boolean

True if the control is a container.

> Overrides Control.container

generic

generic: boolean (optional)

> Inherited from Control.generic

getDataSource

getDataSource: function(): any

> Inherited from Control.getDataSource

hidden

hidden: boolean

True if the control is hidden.

> Inherited from Control.hidden

Methods

applyDesign

applyDesign(design: Design): void

Applies given design to the design on the control. If a design already exists, the prototype chain of the design will be preserved.

> Inherited from Control.applyDesign

Parameters

NAME	TYPE	DESCRIPTION
design	Design	object containing design properties as keys

Returns void

dataContext

dataContext(): any

> Inherited from Control.dataContext

Returns any

getControl

getControl(controlName: string): Control

Given the name of a control, returns the control instance.

Parameters

NAME	TYPE	DESCRIPTION
controlName	string	control name

Returns Control

getControlById

getControlById(id: string): Control

Given the ID of a control, returns the control instance.

Parameters

NAME	TYPE	DESCRIPTION
id	string	control ID

Returns Control

getDesign

getDesign(): Design

Returns the design object of this control.

> Inherited from Control.getDesign

Returns Design

isEditable

isEditable(): boolean

Boolean indicating if the control is editable. Returns false when either the control or its parent is not editable. Returns true when both the control and its parent are editable. Returns true when either the control or its parent is editable and the other is undefined. Returns undefined if both the control's edit-ability and its parent's edit-ability is undefined.

> Inherited from Control.isEditable

Returns boolean

metadata

metadata(): ContainerControlMetadata

Returns the metadata object of this control.

> Overrides Control.metadata

Returns ContainerControlMetadata

parent

parent(): Control | Page

Returns the parent (control or page) of this control.

> Inherited from Control.parent

Returns Control | Page

root

root(): Page

Returns the root form instance (page) of this control.

> Inherited from Control.root

Returns Page

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

ContainerControlDesign type

2/18/2021 • 3 minutes to read • Edit Online

Container control design object has properties specific to all container controls.

Hierarchy

Design
└ ContainerControlDesign
└ GroupDesign
└ ListDesign
└ PartDesign

Index

Properties

- alignItems
- alignSelf
- allowScroll
- background
- bindings
- border
- color
- flexFlow
- flexSize
- fontSize
- fontWeight
- itemBorder
- items
- justifyItems
- label
- labelPosition
- name
- padding
- type

Properties

alignItems

alignItems: string (optional)

This property is an alias for the CSS property "align-items". Please refer to this web page for documentation on the "align-items" property.

> Inherited from Design.alignItems

alignSelf

alignSelf: string (optional)

> Inherited from Design.alignSelf

allowScroll

allowScroll: string (optional)

True if the container will allow scrolling when its items do not fit into the container's available space. If a container has an item which may scroll, then set this property to false to prevent nested scrolling areas.

background

background: string (optional)

The background color of the container. Consider modifying the color attribute in the same container so that fonts overlaying the background color will appear appropriately. Note: if background is set to "theme", the theme color of the app will be used. The following colors are available:

```
blue:           #0078D7;
pomegranate:    #911844;
raspberry:      #8D398F;
darkOrange:     #D24726;
green:          #369F47;
blueberry:      #17234E;
grape:          #432158;
lightBlue:      #5DB2FF;
lightGreen:     #82BA00;
pink:           #DC4FAD;
teal:           #008299;
mediumDarkBlue: #004B8B;
cordovan:       #570000;
darkCordovan:   #380000;
black:          #000000;
lightGray:      #e8e8e8;
light:          #fff   ;
dark:           #333333;
```

bindings

bindings: any (optional)

> Inherited from Design.bindings

border

border: "none" | "solid" | "left" | "right" | "top" | "bottom" (optional)

The border behavior of a control. This property will not be inherited by the children.

> Inherited from Design.border

color

color: string (optional)

The foreground color of the container. This will modify the color of all headers, items, labels, and icons within the container.
Consider setting the background color at the same time as necessary when setting this attribute.
Note: if color is set to "theme", the theme color of the app will be used.
The following colors are available:

```
blue:           #0078D7;
pomegranate:    #911844;
raspberry:      #8D398F;
darkOrange:     #D24726;
green:          #369F47;
blueberry:      #17234E;
grape:          #432158;
lightBlue:      #5DB2FF;
lightGreen:     #82BA00;
pink:           #DC4FAD;
teal:           #008299;
mediumDarkBlue: #004B8B;
cordovan:       #570000;
darkCordovan:   #380000;
black:          #000000;
lightGray:      #e8e8e8;
light:          #fff   ;
dark:           #333333;
```

> Inherited from Design.color

flexFlow

flexFlow: string (optional)

Specifying this property makes the component a flex container component. This property is an alias for the CSS property "flex-flow". Please refer to this web page for documentation on the "flex-flow" property.

> Inherited from Design.flexFlow

flexSize

flexSize: string (optional)

One number or two numbers written as a string. For example, "(size to grow) [(size-to-shrink)]" to accommodate available space in the immediate flex container. This property is an alias for the CSS property "flex". Please refer to this web page for documentation on the "flex" property.

> Inherited from Design.flexSize

fontSize

fontSize: "medium" | "xx-small" | "x-small" | "small" | "large" | "x-large" | "xx-large" (optional)

The proportional text size

> Inherited from Design.fontSize

fontWeight

fontWeight: "normal" | "bold" (optional)

Normal or bold text.

> Inherited from Design.fontWeight

itemBorder

itemBorder: "solid" | "none" (optional)

If true, a border will appear around each row in the list. This property is equivalent to applying the border property individually to all items in the container.

items

items: string | Design [ ] (optional)

An array containing the components to place inside of the container.

justifyItems

justifyItems: "flex-start" | "flex-end" | "center" | "space-between" (optional)

This property is an alias for the CSS property "justify-content". Please refer to this web page for documentation on the "justify-content" property.

> Inherited from Design.justifyItems

label

label: string (optional)

> Inherited from Design.label

labelPosition

labelPosition: "stacked" | "hidden" | "inline" (optional)

Determines how a label is positioned, if at all. By default, labelPosition is set to stacked.

> Inherited from Design.labelPosition

name

name: string (optional)

> Inherited from Design.name

padding

padding: "none" | "small" | "std" (optional)

Allows specifying the component's padding behavior. A component will inherit the padding behavior specified by its parent container components.

> Inherited from Design.padding

type

type: ControlType (optional)

The type of the control as a string.

> Inherited from Design.type

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

ContainerControlMetadata type

2/18/2021 • 2 minutes to read • Edit Online

Container control metadata type.

Hierarchy

ControlMetadata
└ ContainerControlMetadata
  └ GroupMetadata
  └ ListMetadata
  └ PartMetadata

Index

Properties

- BoundEntity
- BoundField
- Description
- Editable
- ExtType
- HelpText
- Hidden
- Id
- Label
- Name
- Order
- Type

Properties

BoundEntity

BoundEntity: string (optional)

The entity to which the control is bound.

> Inherited from ControlMetadata.BoundEntity

BoundField

BoundField: string (optional)

> Inherited from ControlMetadata.BoundField

Description

Description: string (optional)

Description of the control.

> Inherited from ControlMetadata.Description

Editable

Editable: boolean (optional)

Boolean indicating if the control is editable. False when either the control or its parent is not editable. True when both the control and its parent are editable. True when either the control or its parent is editable and the other is undefined. Undefined if both the control's edit-ability and its parent's edit-ability is undefined.

> Inherited from ControlMetadata.Editable

ExtType

ExtType: ControlType (optional)

The extended control type. For example, a control of type Input might have an extended type of Barcode.

> Inherited from ControlMetadata.ExtType

HelpText

HelpText: string (optional)

The keyboard shortcut for a command. For example, "(Shift+F5)"

> Inherited from ControlMetadata.HelpText

Hidden

Hidden: boolean (optional)

Boolean indicating if the control is hidden or not.

> Inherited from ControlMetadata.Hidden

Id

Id: string (optional)

Identification string for a control.

> Inherited from ControlMetadata.Id

Label

Label: string (optional)

Label for a control. For example, a control representing a person's first name might have a label "First Name".

> Inherited from ControlMetadata.Label

Name

Name: string (optional)

Name of a control.

> Inherited from ControlMetadata.Name

Order

Order: number (optional)

Number indicating the order in which a control will appear on a page.

> Inherited from ControlMetadata.Order

Type

Type: ControlType (optional)

String indicating the control type.

> Inherited from ControlMetadata.Type

ControlMetadata type

2/18/2021 • 2 minutes to read • Edit Online

Interface for the metadata of a control. Overriding control metadata can modify a controls' look and behavior. Properties that can be modified vary by control but every control will have the base properties listed here.

Hierarchy

ControlMetadata
  └ PageLinkMetadata
  └ ContainerControlMetadata
  └ InputControlMetadata
  └ ImageMetadata

Index

Properties

- BoundEntity
- BoundField
- Description
- Editable
- ExtType
- HelpText
- Hidden
- Id
- Label
- Name
- Order
- Type

Properties

BoundEntity

BoundEntity: string (optional)

The entity to which the control is bound.

BoundField

BoundField: string (optional)

Description

Description: string (optional)

Description of the control.

Editable

Editable: boolean (optional)

Boolean indicating if the control is editable. False when either the control or its parent is not editable. True when both the control and its parent are editable. True when either the control or its parent is editable and the other is undefined. Undefined if both the control's edit-ability and its parent's edit-ability is undefined.

ExtType

ExtType: ControlType (optional)

The extended control type. For example, a control of type Input might have an extended type of Barcode.

HelpText

HelpText: string (optional)

The keyboard shortcut for a command. For example, "(Shift+F5)"

Hidden

Hidden: boolean (optional)

Boolean indicating if the control is hidden or not.

Id

Id: string (optional)

Identification string for a control.

Label

Label: string (optional)

Label for a control. For example, a control representing a person's first name might have a label "First Name".

Name

Name: string (optional)

Name of a control.

Order

Order: number (optional)

Number indicating the order in which a control will appear on a page.

Type

Type: ControlType (optional)

String indicating the control type.

DataService type

2/18/2021 • 2 minutes to read • Edit Online

Provides ability access data under the application workspace.

Hierarchy

DataService

Index

Methods

- findEntityData
- getEntityData
- getPageData

Methods

findEntityData

findEntityData(entityType: any, propertyName: string, propertyValue: any, includeChanges?: boolean): any

Parameters

NAME	TYPE	DESCRIPTION
entityType	any	
propertyName	string	
propertyValue	any	
includeChanges?	boolean	

Returns any

getEntityData

getEntityData(entityType: any, entityId: string): any

Parameters

NAME	TYPE	DESCRIPTION
entityType	any	
entityId	string	

Returns any

getPageData

getPageData(pageId: string, context: any, filter: any, allowedStaleness: number): Promise <PageData>

Parameters

NAME	TYPE	DESCRIPTION
pageId	string	
context	any	
filter	any	
allowedStaleness	number	

Returns Promise <PageData>

Deferred type<T>

2/18/2021 • 2 minutes to read • Edit Online

Hierarchy

Deferred

Index

Properties

- promise

Methods

- reject
- resolve

Properties

promise

promise: Promise <T>

Methods

reject

reject(error?: any): void

Parameters

NAME	TYPE	DESCRIPTION
error?	any	

Returns void

resolve

resolve(value?: T | PromiseLike <T>): void

Parameters

NAME	TYPE	DESCRIPTION
value?	T	PromiseLike <T>	

Returns void

Design type

2/18/2021 • 2 minutes to read • Edit Online

Design object type. For more information on design, please reference the Design Introduction.

Hierarchy

Design
└ PageLinkDesign
└ ContainerControlDesign
└ InputControlDesign
└ ImageDesign

Index

Properties

Properties

alignItems

alignItems: string (optional)

This property is an alias for the CSS property "align-items". Please refer to this web page for documen the "align-items" property.

alignSelf

alignSelf: string (optional)

bindings

bindings: any (optional)

border

border: "none" | "solid" | "left" | "right" | "top" | "bottom" (optional)

The border behavior of a control. This property will not be inherited by the children.

color

color: string (optional)

The foreground color of the container. This will modify the color of all headers, items, labels, and icons container.
Consider setting the background color at the same time as necessary when setting this attribute.
Note: if color is set to "theme", the theme color of the app will be used.
The following colors are available:

```
blue:           #0078D7;
pomegranate:    #911844;
raspberry:      #8D398F;
darkOrange:     #D24726;
green:          #369F47;
blueberry:      #17234E;
grape:          #432158;
lightBlue:      #5DB2FF;
lightGreen:     #82BA00;
pink:           #DC4FAD;
teal:           #008299;
mediumDarkBlue: #004B8B;
cordovan:       #570000;
darkCordovan:   #380000;
black:          #000000;
lightGray:      #e8e8e8;
light:          #fff   ;
dark:           #333333;
```

flexFlow

flexFlow: string (optional)

Specifying this property makes the component a flex container component. This property is an alias fo property "flex-flow". Please refer to this web page for documentation on the "flex-flow" property.

flexSize

flexSize: string (optional)

One number or two numbers written as a string. For example, "(size to grow) [(size-to-shrink)]" to acc available space in the immediate flex container. This property is an alias for the CSS property "flex". Ple to this web page for documentation on the "flex" property.

fontSize

fontSize: "medium" | "xx-small" | "x-small" | "small" | "large" | "x-large" | "xx-large" (optional)

The proportional text size

fontWeight

fontWeight: "normal" | "bold" (optional)

Normal or bold text.

justifyItems

justifyItems: "flex-start" | "flex-end" | "center" | "space-between" (optional)

This property is an alias for the CSS property "justify-content". Please refer to this web page for docun on the "justify-content" property.

label

label: string (optional)

labelPosition

labelPosition: "stacked" | "hidden" | "inline" (optional)

Determines how a label is positioned, if at all. By default, labelPosition is set to stacked.

name

name: string (optional)

padding

padding: "none" | "small" | "std" (optional)

Allows specifying the component's padding behavior. A component will inherit the padding behavior s by its parent container components.

type

type: ControlType (optional)

The type of the control as a string.

EventHook type<T>

2/18/2021 • 2 minutes to read • Edit Online

Hierarchy

EventHook

Index

Methods

- subscribe
- unsubscribe
- unsubscribeAll

Methods

subscribe

subscribe(listener: IEventListener <T>): void

Subscribe a listener to this event.

Parameters

NAME	TYPE	DESCRIPTION
listener	IEventListener <T>	

Returns void

unsubscribe

unsubscribe(listener: IEventListener <T>): void

Unsubscribe a listener from this event.

Parameters

NAME	TYPE	DESCRIPTION
listener	IEventListener <T>	

Returns void

unsubscribeAll

unsubscribeAll(): void

Remove all listeners from this event.

Returns void

Field type

2/18/2021 • 2 minutes to read • Edit Online

Field control type.

Hierarchy

InputControl
└ Field

Index

Properties

- container
- generic
- getDataSource
- hidden

Methods

- applyDesign
- dataContext
- getDesign
- getEditableFormattedValue
- getEditableValue
- getEntityRef
- getFormattedValue
- getRefLink
- getValue
- hasRefLink
- hasUnWrapText
- isEditable
- metadata
- parent
- root
- setEditableValue

Events

- onDataChanged

Properties

container

container: boolean (optional)

True if the control is a container.

> Inherited from Control.container

generic

generic: boolean (optional)

> Inherited from Control.generic

getDataSource

getDataSource: function(): any

> Inherited from Control.getDataSource

hidden

hidden: boolean

True if the control is hidden.

> Inherited from Control.hidden

Methods

applyDesign

applyDesign(IDesign: FieldDesign): void

Applies given design to the design on the control. If a design already exists, the prototype chain of the design will be preserved.

> Overrides Control.applyDesign

Parameters

NAME	TYPE	DESCRIPTION
IDesign	FieldDesign	object containing design properties as keys

Returns void

dataContext

dataContext(): any

> Inherited from Control.dataContext

Returns any

getDesign

getDesign(): Design

Returns the design object of this control.

> Inherited from Control.getDesign

Returns Design

getEditableFormattedValue

getEditableFormattedValue(): string | number | Date

Gets a formatted decimal string value of an editable field control.

Returns string | number | Date

getEditableValue

getEditableValue(): string | number | Date

Gets the value for an editable field control.

Returns string | number | Date

getEntityRef

getEntityRef(): any

Gets value of entityRef binding to control.

Returns any

value of entityRef binding to control

getFormattedValue

getFormattedValue(): string

Gets a formatted decimal string value.

Returns string

getRefLink

getRefLink(): NavigationArgs

Gets the navigation object for a reference link.

Returns NavigationArgs

getValue

getValue(): any

Gets the value for a field control.

Returns any

hasRefLink

hasRefLink(): boolean

Returns true if the field has a refLink, otherwise false.

Returns boolean

hasUnWrapText

hasUnWrapText(): boolean

Gets wrap text property of control.

Returns boolean

isEditable

isEditable(): boolean

Boolean indicating if the control is editable. Returns false when either the control or its parent is not editable. Returns true when both the control and its parent are editable. Returns true when either the control or its parent is editable and the other is undefined. Returns undefined if both the control's edit-ability and its parent's edit-ability is undefined.

> Inherited from Control.isEditable

Returns boolean

metadata

metadata(): FieldMetadata

Returns the metadata object of this control.

> Overrides InputControl.metadata

Returns FieldMetadata

parent

parent(): Control | Page

Returns the parent (control or page) of this control.

> Inherited from Control.parent

Returns Control | Page

root

root(): Page

Returns the root form instance (page) of this control.

> Inherited from Control.root

Returns Page

setEditableValue

setEditableValue(value: string | number | Date): void

Sets the value for an editable field control.

Parameters

NAME	TYPE	DESCRIPTION
value	string | number | Date	value

Returns void

Events

onDataChanged

onDataChanged: EventHook <null>

An event that is triggered when the input control's data changes.

> Inherited from InputControl.onDataChanged

FieldDesign type

2/18/2021 • 2 minutes to read • Edit Online

Design object interface for a field control.

Hierarchy

InputControlDesign
└ FieldDesign

Index

Properties

Properties

alignItems

alignItems: string (optional)

This property is an alias for the CSS property "align-items". Please refer to this web page for documentation on the "align-items" property.

> Inherited from Design.alignItems

alignSelf

alignSelf: string (optional)

> Inherited from Design.alignSelf

bindings

bindings: any (optional)

> Inherited from Design.bindings

border

border: "none" | "solid" | "left" | "right" | "top" | "bottom" (optional)

The border behavior of a control. This property will not be inherited by the children.

> Inherited from Design.border

color

color: string (optional)

The foreground color of the container. This will modify the color of all headers, items, labels, and icons within the container.
Consider setting the background color at the same time as necessary when setting this attribute.
Note: if color is set to "theme", the theme color of the app will be used.
The following colors are available:

```
blue:          #0078D7;
pomegranate:   #911844;
raspberry:     #8D398F;
darkOrange:    #D24726;
green:         #369F47;
blueberry:     #17234E;
grape:         #432158;
lightBlue:     #5DB2FF;
lightGreen:    #82BA00;
pink:          #DC4FAD;
teal:          #008299;
mediumDarkBlue: #004B8B;
cordovan:      #570000;
darkCordovan:  #380000;
black:         #000000;
lightGray:     #e8e8e8;
light:         #fff   ;
dark:          #333333;
```

> Inherited from Design.color

flexFlow

flexFlow: string (optional)

Specifying this property makes the component a flex container component. This property is an alias for the CSS property "flex-flow". Please refer to this web page for documentation on the "flex-flow" property.

> Inherited from Design.flexFlow

flexSize

flexSize: string (optional)

One number or two numbers written as a string. For example, "(size to grow) [(size-to-shrink)]" to accommodate available space in the immediate flex container. This property is an alias for the CSS property "flex". Please refer to this web page for documentation on the "flex" property.

> Inherited from Design.flexSize

fontSize

fontSize: "medium" | "xx-small" | "x-small" | "small" | "large" | "x-large" | "xx-large" (optional)

The proportional text size

> Inherited from Design.fontSize

fontWeight

fontWeight: "normal" | "bold" (optional)

Normal or bold text.

> Inherited from Design.fontWeight

justifyItems

justifyItems: "flex-start" | "flex-end" | "center" | "space-between" (optional)

This property is an alias for the CSS property "justify-content". Please refer to this web page for documentation on the "justify-content" property.

> Inherited from Design.justifyItems

label

label: string (optional)

> Inherited from Design.label

labelPosition

labelPosition: "stacked" | "hidden" | "inline" (optional)

Determines how a label is positioned, if at all. By default, labelPosition is set to stacked.

> Inherited from Design.labelPosition

name

name: string (optional)

> Inherited from Design.name

padding

padding: "none" | "small" | "std" (optional)

Allows specifying the component's padding behavior. A component will inherit the padding behavior specified by its parent container components.

> Inherited from Design.padding

type

type: ControlType (optional)

The type of the control as a string.

> Inherited from Design.type

FieldMetadata type

2/18/2021 • 3 minutes to read • Edit Online

Interface for field metadata.

Hierarchy

InputControlMetadata
└ FieldMetadata

Index

Properties

Properties

BoundEntity

BoundEntity: string (optional)

The entity to which the control is bound.

> Inherited from ControlMetadata.BoundEntity

BoundField

BoundField: string (optional)

> Inherited from ControlMetadata.BoundField

DecimalPlaces

DecimalPlaces: number (optional)

The number of decimals that appear on a field of type "Real". Default = 2; number must be in the range [0:20].

Description

Description: string (optional)

Description of the control.

> Inherited from ControlMetadata.Description

Editable

Editable: boolean (optional)

Boolean indicating if the control is editable. False when either the control or its parent is not editable. True when both the control and its parent are editable. True when either the control or its parent is editable and the other is undefined. Undefined if both the control's edit-ability and its parent's edit-ability is undefined.

> Inherited from ControlMetadata.Editable

ExtType

ExtType: ControlType (optional)

The extended control type. For example, a control of type Input might have an extended type of Barcode.

> Inherited from ControlMetadata.ExtType

Formatting

Formatting: any (optional)

Formats a field of type "DateTime" or "Date". **Note:** if browser does not support `toLocaleString` with options then it will show the entire value.
The options for Formatting depends on the Style that has been chosen: Style: "DateOnly" options, Style: "TimeOnly" options, and options for no style.

Example 1: `{ Style: "TimeOnly", Formatting: { timeZone: "UTC", timeZoneName: "short" } }`

Example 2: `{ Style: "DateOnly", Formatting: { month: "long", day: "numeric" } }` result: March 2

HelpText

HelpText: string (optional)

The keyboard shortcut for a command. For example, "(Shift+F5)"

> Inherited from ControlMetadata.HelpText

Hidden

Hidden: boolean (optional)

Boolean indicating if the control is hidden or not.

Inherited from ControlMetadata.Hidden

Id

Id: string (optional)

Identification string for a control.

Inherited from ControlMetadata.Id

Label

Label: string (optional)

Label for a control. For example, a control representing a person's first name might have a label "First Name".

Inherited from ControlMetadata.Label

LinkType

LinkType: "Telephone" | "Email" | "Url" (optional)

Assigning the link type of a field allows for the appropriate mobile application to be opened when the link is selected.

Mandatory

Mandatory: boolean (optional)

If set to true then input for the control is required for the task to be completed. Mandatory controls will have a red outline.

Inherited from InputControlMetadata.Mandatory

Name

Name: string (optional)

Name of a control.

Inherited from ControlMetadata.Name

NumSequence

NumSequence: NumberSequenceConfig (optional)

Used for auto detecting and changing visibility of the number sequence controls in the task or page, based on AX number sequence configuration, through extended business logic. Example:

```
// hide number sequence reference page from users
metadataService.hideNavigation('numSeqReferencePage');

// parameters to be passed to 'numSequence' flag in configureControl
var configParam = {
     referencePageName: 'numSeqReferencePage',
     dataType: 'HcmPersonnelNumberId'
};

// setup 'PersonnelNumber' control as number sequence in the task 'add-worker'
metadataService.configureControl('add-worker', 'PersonnelNumber', { numSequence: configParam });
```

> Inherited from InputControlMetadata.NumSequence

Order

Order: number (optional)

Number indicating the order in which a control will appear on a page.

> Inherited from ControlMetadata.Order

ReferenceAppId

ReferenceAppId: string (optional)

The ID of the app that the field control lives in.

ReferencePageId

ReferencePageId: string (optional)

The ID of the page that the field control lives in.

Style

Style: string (optional)

Styles a field of type "DateTime" or "Date". Example: `{ Style: TimeOnly }` result: 12:00:00 AM

Type

Type: ControlType (optional)

String indicating the control type.

> Inherited from ControlMetadata.Type

UnWrapText

UnWrapText: boolean (optional)

False by default -- text of the page will be wrapped.

WrapText

WrapText: boolean (optional)

If true then the text of the field control will wrap to the next line.

FileUploader type

2/18/2021 • 2 minutes to read • Edit Online

File uploader control type. A control for uploading files such as images.

Hierarchy

Value
└ FileUploader

Index

Properties

- container
- generic
- getDataSource
- hidden
- image

Methods

- applyDesign
- canLoadFromDevice
- dataContext
- getDesign
- getImage
- getValue
- isEditable
- loadFromFileSystem
- metadata
- parent
- root
- setCamera
- setValue

Events

- onDataChanged

Properties

container

container: boolean (optional)

True if the control is a container.

> Inherited from Control.container

generic

generic: boolean (optional)

> Inherited from Control.generic

getDataSource

getDataSource: function(): any

> Inherited from Control.getDataSource

hidden

hidden: boolean

True if the control is hidden.

> Inherited from Control.hidden

image

image: Image

Methods

applyDesign

applyDesign(IDesign: FileUploaderDesign): void

Applies given design to the design on the control. If a design already exists, the prototype chain of the design will be preserved.

> Overrides Control.applyDesign

Parameters

NAME	TYPE	DESCRIPTION
IDesign	FileUploaderDesign	object containing design properties as keys

Returns void

canLoadFromDevice

canLoadFromDevice(): boolean

Returns true if the mobile phone has camera plugin.

Returns boolean

dataContext

dataContext(): any

> Inherited from Control.dataContext

Returns any

getDesign

getDesign(): Design

Returns the design object of this control.

> Inherited from Control.getDesign

Returns Design

getImage

getImage(options: any): Promise <string>

Returns a promise of an object with image data.

Parameters

NAME	TYPE	DESCRIPTION
options	any	see camera options

Returns Promise <string>

getValue

getValue(): any

Gets the value of the entity that is bound to the control.

> Overrides Value.getValue

Returns any

isEditable

isEditable(): boolean

Boolean indicating if the control is editable. Returns false when either the control or its parent is not editable. Returns true when both the control and its parent are editable. Returns true when either the control or its parent is editable and the other is undefined. Returns undefined if both the control's edit-ability and its parent's edit-ability is undefined.

> Inherited from Control.isEditable

Returns boolean

loadFromFileSystem

loadFromFileSystem(file: Blob): Promise <any>

Parameters

NAME	TYPE	DESCRIPTION
file	Blob	

Returns Promise <any>

metadata

metadata(): FileUploaderMetadata

Returns the metadata object of this control.

> Overrides Value.metadata

Returns FileUploaderMetadata

parent

parent(): Control | Page

Returns the parent (control or page) of this control.

> Inherited from Control.parent

Returns Control | Page

root

root(): Page

Returns the root form instance (page) of this control.

> Inherited from Control.root

Returns Page

setCamera

setCamera(camera: any): void

Set the camera object on the control.

Parameters

NAME	TYPE	DESCRIPTION
camera	any	

Returns void

setValue

setValue(value: string): void

Sets the value of the control.

> Inherited from Value.setValue

Parameters

NAME	TYPE	DESCRIPTION
value	string	

Returns void

Events

onDataChanged

onDataChanged: EventHook <null>

An event that is triggered when the input control's data changes.

> Inherited from InputControl.onDataChanged

FileUploaderDesign type

2/18/2021 • 2 minutes to read • Edit Online

File uploader design object type.

Hierarchy

ValueDesign
└ FileUploaderDesign

Index

Properties

Properties

alignItems

alignItems: string (optional)

This property is an alias for the CSS property "align-items". Please refer to this web page for documentation on the "align-items" property.

> Inherited from Design.alignItems

alignSelf

alignSelf: string (optional)

> Inherited from Design.alignSelf

bindings

bindings: any (optional)

> Inherited from Design.bindings

border

border: "none" | "solid" | "left" | "right" | "top" | "bottom" (optional)

The border behavior of a control. This property will not be inherited by the children.

> Inherited from Design.border

color

color: string (optional)

The foreground color of the container. This will modify the color of all headers, items, labels, and icons within the container.
Consider setting the background color at the same time as necessary when setting this attribute.
Note: if color is set to "theme", the theme color of the app will be used.
The following colors are available:

```
blue:          #0078D7;
pomegranate:   #911844;
raspberry:     #8D398F;
darkOrange:    #D24726;
green:         #369F47;
blueberry:     #17234E;
grape:         #432158;
lightBlue:     #5DB2FF;
lightGreen:    #82BA00;
pink:          #DC4FAD;
teal:          #008299;
mediumDarkBlue: #004B8B;
cordovan:      #570000;
darkCordovan:  #380000;
black:         #000000;
lightGray:     #e8e8e8;
light:         #fff   ;
dark:          #333333;
```

> Inherited from Design.color

flexFlow

flexFlow: string (optional)

Specifying this property makes the component a flex container component. This property is an alias for the CSS property "flex-flow". Please refer to this web page for documentation on the "flex-flow" property.

> Inherited from Design.flexFlow

flexSize

flexSize: string (optional)

One number or two numbers written as a string. For example, "(size to grow) [(size-to-shrink)]" to accommodate available space in the immediate flex container. This property is an alias for the CSS property "flex". Please refer to this web page for documentation on the "flex" property.

> Inherited from Design.flexSize

fontSize

fontSize: "medium" | "xx-small" | "x-small" | "small" | "large" | "x-large" | "xx-large" (optional)

The proportional text size

> Inherited from Design.fontSize

fontWeight

fontWeight: "normal" | "bold" (optional)

Normal or bold text.

> Inherited from Design.fontWeight

justifyItems

justifyItems: "flex-start" | "flex-end" | "center" | "space-between" (optional)

This property is an alias for the CSS property "justify-content". Please refer to this web page for documentation on the "justify-content" property.

> Inherited from Design.justifyItems

label

label: string (optional)

> Inherited from Design.label

labelPosition

labelPosition: "stacked" | "hidden" | "inline" (optional)

Determines how a label is positioned, if at all. By default, labelPosition is set to stacked.

> Inherited from Design.labelPosition

name

name: string (optional)

> Inherited from Design.name

padding

padding: "none" | "small" | "std" (optional)

Allows specifying the component's padding behavior. A component will inherit the padding behavior specified by its parent container components.

> Inherited from Design.padding

type

type: ControlType (optional)

The type of the control as a string.

> Inherited from Design.type

FileUploaderMetadata type

2/18/2021 • 2 minutes to read • Edit Online

File uploader metadata type.

Hierarchy

ValueMetadata
└ FileUploaderMetadata

Index

Properties

Properties

BoundEntity

BoundEntity: string (optional)

The entity to which the control is bound.

> Inherited from ControlMetadata.BoundEntity

BoundField

BoundField: string (optional)

> Inherited from ControlMetadata.BoundField

Description

Description: string (optional)

Description of the control.

> Inherited from ControlMetadata.Description

Editable

Editable: boolean (optional)

Boolean indicating if the control is editable. False when either the control or its parent is not editable. True when both the control and its parent are editable. True when either the control or its parent is editable and the other is undefined. Undefined if both the control's edit-ability and its parent's edit-ability is undefined.

> Inherited from ControlMetadata.Editable

ExtType

ExtType: ControlType (optional)

The extended control type. For example, a control of type Input might have an extended type of Barcode.

> Inherited from ControlMetadata.ExtType

HelpText

HelpText: string (optional)

The keyboard shortcut for a command. For example, "(Shift+F5)"

> Inherited from ControlMetadata.HelpText

Hidden

Hidden: boolean (optional)

Boolean indicating if the control is hidden or not.

> Inherited from ControlMetadata.Hidden

Id

Id: string (optional)

Identification string for a control.

> Inherited from ControlMetadata.Id

Label

Label: string (optional)

Label for a control. For example, a control representing a person's first name might have a label "First Name".

> Inherited from ControlMetadata.Label

Mandatory

Mandatory: boolean (optional)

If set to true then input for the control is required for the task to be completed. Mandatory controls will have a red outline.

> Inherited from InputControlMetadata.Mandatory

Name

Name: string (optional)

Name of a control.

> Inherited from ControlMetadata.Name

NumSequence

NumSequence: NumberSequenceConfig (optional)

Used for auto detecting and changing visibility of the number sequence controls in the task or page, based on AX number sequence configuration, through extended business logic. Example:

```
// hide number sequence reference page from users
metadataService.hideNavigation('numSeqReferencePage');

// parameters to be passed to 'numSequence' flag in configureControl
var configParam = {
     referencePageName: 'numSeqReferencePage',
     dataType: 'HcmPersonnelNumberId'
};

// setup 'PersonnelNumber' control as number sequence in the task 'add-worker'
metadataService.configureControl('add-worker', 'PersonnelNumber', { numSequence: configParam });
```

> Inherited from InputControlMetadata.NumSequence

Order

Order: number (optional)

Number indicating the order in which a control will appear on a page.

> Inherited from ControlMetadata.Order

Type

Type: ControlType (optional)

String indicating the control type.

> Inherited from ControlMetadata.Type

GenericValue type

2/18/2021 • 2 minutes to read • Edit Online

Generic value control type.

Hierarchy

Value
└ GenericValue

Index

Properties

- container
- generic
- getDataSource
- hidden

Methods

- applyDesign
- dataContext
- getDesign
- getValue
- isEditable
- metadata
- parent
- root
- setValue

Events

- onDataChanged

Properties

container

container: boolean (optional)

True if the control is a container.

> Inherited from Control.container

generic

generic: boolean

True if the control is a generic.

> Overrides Control.generic

getDataSource

getDataSource: function(): any

> Inherited from Control.getDataSource

hidden

hidden: boolean

True if the control is hidden.

> Inherited from Control.hidden

Methods

applyDesign

applyDesign(design: Design): void

Applies given design to the design on the control. If a design already exists, the prototype chain of the design will be preserved.

> Inherited from Control.applyDesign

Parameters

NAME	TYPE	DESCRIPTION
design	Design	object containing design properties as keys

Returns void

dataContext

dataContext(): any

> Inherited from Control.dataContext

Returns any

getDesign

getDesign(): Design

Returns the design object of this control.

> Inherited from Control.getDesign

Returns Design

getValue

getValue(): string

Returns the value of the control.

> Inherited from Value.getValue

Returns string

isEditable

isEditable(): boolean

Boolean indicating if the control is editable. Returns false when either the control or its parent is not editable. Returns true when both the control and its parent are editable. Returns true when either the control or its parent is editable and the other is undefined. Returns undefined if both the control's edit-ability and its parent's edit-ability is undefined.

> Inherited from Control.isEditable

Returns boolean

metadata

metadata(): ValueMetadata

Returns the metadata object of this control.

> Inherited from Value.metadata
>
> Overrides InputControl.metadata

Returns ValueMetadata

parent

parent(): Control | Page

Returns the parent (control or page) of this control.

> Inherited from Control.parent

Returns Control | Page

root

root(): Page

Returns the root form instance (page) of this control.

> Inherited from Control.root

Returns Page

setValue

setValue(value: string): void

Sets the value of the control.

> Inherited from Value.setValue

Parameters

NAME	TYPE	DESCRIPTION
value	string	

Returns void

Events

onDataChanged

onDataChanged: EventHook <null>

An event that is triggered when the input control's data changes.

> Inherited from InputControl.onDataChanged

Control type

2/18/2021 • 2 minutes to read • Edit Online

Control interface with base methods and attributes for all controls. This represents the runtime instance of a control. Modifying the properties are immediately reflected in the UI.

Hierarchy

Control
- PageLink
- ContainerControl
- InputControl
- Image

Index

Properties

- container
- generic
- getDataSource
- hidden

Methods

- applyDesign
- dataContext
- getDesign
- isEditable
- metadata
- parent
- root

Properties

container

container: boolean (optional)

True if the control is a container.

generic

generic: boolean (optional)

getDataSource

getDataSource: function(): any

hidden

hidden: boolean

True if the control is hidden.

Methods

applyDesign

applyDesign(design: Design): void

Applies given design to the design on the control. If a design already exists, the prototype chain of the design will be preserved.

Parameters

NAME	TYPE	DESCRIPTION
design	Design	object containing design properties as keys

Returns void

dataContext

dataContext(): any

Returns any

getDesign

getDesign(): Design

Returns the design object of this control.

Returns Design

isEditable

isEditable(): boolean

Boolean indicating if the control is editable. Returns false when either the control or its parent is not editable. Returns true when both the control and its parent are editable. Returns true when either the control or its parent is editable and the other is undefined. Returns undefined if both the control's edit-ability and its parent's edit-ability is undefined.

Returns boolean

metadata

metadata(): ControlMetadata

Returns the metadata object of this control.

Returns ControlMetadata

parent

parent(): Control | Page

Returns the parent (control or page) of this control.

Returns Control | Page

root

root(): Page

Returns the root form instance (page) of this control.

Returns Page

Group type

2/18/2021 • 2 minutes to read • Edit Online

Group container control type. A group control is a container control that has any number of controls as children.

Hierarchy

ContainerControl
└ Group

Index

Properties

- container
- generic
- getDataSource
- hidden

Methods

- applyDesign
- dataContext
- getChildren
- getControl
- getControlById
- getDesign
- isEditable
- metadata
- parent
- root

Properties

container

container: boolean

True if the control is a container.

> Inherited from ContainerControl.container
>
> Overrides Control.container

generic

generic: boolean (optional)

> Inherited from Control.generic

getDataSource

getDataSource: function(): any

Inherited from Control.getDataSource

hidden

hidden: boolean

True if the control is hidden.

Inherited from Control.hidden

Methods

applyDesign

applyDesign(IDesign: GroupDesign): void

Applies given design to the design on the control. If a design already exists, the prototype chain of the design will be preserved.

Overrides Control.applyDesign

Parameters

NAME	TYPE	DESCRIPTION
IDesign	GroupDesign	object containing design properties as keys

Returns void

dataContext

dataContext(): any

Inherited from Control.dataContext

Returns any

getChildren

getChildren(): Control [ ]

Returns the list of children associated with this group control.

Returns Control [ ]

getControl

getControl(controlName: string): Control

Given the name of a control, returns the control instance.

Inherited from ContainerControl.getControl

Parameters

NAME	TYPE	DESCRIPTION
controlName	string	control name

Returns Control

getControlById

getControlById(id: string): Control

Given the ID of a control, returns the control instance.

> Inherited from ContainerControl.getControlById

Parameters

NAME	TYPE	DESCRIPTION
id	string	control ID

Returns Control

getDesign

getDesign(): Design

Returns the design object of this control.

> Inherited from Control.getDesign

Returns Design

isEditable

isEditable(): boolean

Boolean indicating if the control is editable. Returns false when either the control or its parent is not editable. Returns true when both the control and its parent are editable. Returns true when either the control or its parent is editable and the other is undefined. Returns undefined if both the control's edit-ability and its parent's edit-ability is undefined.

> Inherited from Control.isEditable

Returns boolean

metadata

metadata(): GroupMetadata

Returns the metadata object of this control.

> Overrides ContainerControl.metadata

Returns GroupMetadata

parent

parent(): Control | Page

Returns the parent (control or page) of this control.

> Inherited from Control.parent

Returns Control | Page

root

root(): Page

Returns the root form instance (page) of this control.

Inherited from Control.root

Returns Page

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

GroupDesign type

2/18/2021 • 3 minutes to read • Edit Online

Group design object type.

Hierarchy

ContainerControlDesign
└ GroupDesign

Index

Properties

- alignItems
- alignSelf
- allowScroll
- background
- bindings
- border
- color
- flexFlow
- flexSize
- fontSize
- fontWeight
- itemBorder
- items
- justifyItems
- label
- labelPosition
- name
- padding
- type

Properties

alignItems

alignItems: string (optional)

This property is an alias for the CSS property "align-items". Please refer to this web page for documentation on the "align-items" property.

> Inherited from Design.alignItems

alignSelf

alignSelf: string (optional)

> Inherited from Design.alignSelf

allowScroll

allowScroll: string (optional)

True if the container will allow scrolling when its items do not fit into the container's available space. If a container has an item which may scroll, then set this property to false to prevent nested scrolling areas.

> Inherited from ContainerControlDesign.allowScroll

background

background: string (optional)

The background color of the container. Consider modifying the color attribute in the same container so that fonts overlaying the background color will appear appropriately. Note: if background is set to "theme", the theme color of the app will be used. The following colors are available:

```
blue:           #0078D7;
pomegranate:    #911844;
raspberry:      #8D398F;
darkOrange:     #D24726;
green:          #369F47;
blueberry:      #17234E;
grape:          #432158;
lightBlue:      #5DB2FF;
lightGreen:     #82BA00;
pink:           #DC4FAD;
teal:           #008299;
mediumDarkBlue: #004B8B;
cordovan:       #570000;
darkCordovan:   #380000;
black:          #000000;
lightGray:      #e8e8e8;
light:          #fff   ;
dark:           #333333;
```

> Inherited from ContainerControlDesign.background

bindings

bindings: any (optional)

> Inherited from Design.bindings

border

border: "none" | "solid" | "left" | "right" | "top" | "bottom" (optional)

The border behavior of a control. This property will not be inherited by the children.

> Inherited from Design.border

color

color: string (optional)

The foreground color of the container. This will modify the color of all headers, items, labels, and icons within the container.
Consider setting the background color at the same time as necessary when setting this attribute.
Note: if color is set to "theme", the theme color of the app will be used.
The following colors are available:

```
blue:           #0078D7;
pomegranate:    #911844;
raspberry:      #8D398F;
darkOrange:     #D24726;
green:          #369F47;
blueberry:      #17234E;
grape:          #432158;
lightBlue:      #5DB2FF;
lightGreen:     #82BA00;
pink:           #DC4FAD;
teal:           #008299;
mediumDarkBlue: #004B8B;
cordovan:       #570000;
darkCordovan:   #380000;
black:          #000000;
lightGray:      #e8e8e8;
light:          #fff   ;
dark:           #333333;
```

> Inherited from Design.color

flexFlow

flexFlow: string (optional)

Specifying this property makes the component a flex container component. This property is an alias for the CSS property "flex-flow". Please refer to this web page for documentation on the "flex-flow" property.

> Inherited from Design.flexFlow

flexSize

flexSize: string (optional)

One number or two numbers written as a string. For example, "(size to grow) [(size-to-shrink)]" to accommodate available space in the immediate flex container. This property is an alias for the CSS property "flex". Please refer to this web page for documentation on the "flex" property.

> Inherited from Design.flexSize

fontSize

fontSize: "medium" | "xx-small" | "x-small" | "small" | "large" | "x-large" | "xx-large" (optional)

The proportional text size

> Inherited from Design.fontSize

fontWeight

fontWeight: "normal" | "bold" (optional)

Normal or bold text.

> Inherited from Design.fontWeight

itemBorder

itemBorder: "solid" | "none" (optional)

If true, a border will appear around each row in the list. This property is equivalent to applying the border property individually to all items in the container.

> Inherited from ContainerControlDesign.itemBorder

items

items: string | Design [ ] (optional)

An array containing the components to place inside of the container.

> Inherited from ContainerControlDesign.items

justifyItems

justifyItems: "flex-start" | "flex-end" | "center" | "space-between" (optional)

This property is an alias for the CSS property "justify-content". Please refer to this web page for documentation on the "justify-content" property.

> Inherited from Design.justifyItems

label

label: string (optional)

> Inherited from Design.label

labelPosition

labelPosition: "stacked" | "hidden" | "inline" (optional)

Determines how a label is positioned, if at all. By default, labelPosition is set to stacked.

> Inherited from Design.labelPosition

name

name: string (optional)

> Inherited from Design.name

padding

padding: "none" | "small" | "std" (optional)

Allows specifying the component's padding behavior. A component will inherit the padding behavior specified by its parent container components.

> Inherited from Design.padding

type

type: ControlType (optional)

The type of the control as a string.

> Inherited from Design.type

GroupMetadata type

2/18/2021 • 2 minutes to read • Edit Online

Group metadata type.

Hierarchy

ContainerControlMetadata
└ GroupMetadata

Index

Properties

Properties

BoundEntity

BoundEntity: string (optional)

The entity to which the control is bound.

> Inherited from ControlMetadata.BoundEntity

BoundField

BoundField: string (optional)

> Inherited from ControlMetadata.BoundField

Children

Children: ControlMetadata [ ] (optional)

List of control metadata for each child control.

Description

Description: string (optional)

Description of the control.

> Inherited from ControlMetadata.Description

Editable

Editable: boolean (optional)

Boolean indicating if the control is editable. False when either the control or its parent is not editable. True when both the control and its parent are editable. True when either the control or its parent is editable and the other is undefined. Undefined if both the control's edit-ability and its parent's edit-ability is undefined.

> Inherited from ControlMetadata.Editable

ExtType

ExtType: ControlType (optional)

The extended control type. For example, a control of type Input might have an extended type of Barcode.

> Inherited from ControlMetadata.ExtType

HelpText

HelpText: string (optional)

The keyboard shortcut for a command. For example, "(Shift+F5)"

> Inherited from ControlMetadata.HelpText

Hidden

Hidden: boolean (optional)

Boolean indicating if the control is hidden or not.

> Inherited from ControlMetadata.Hidden

Id

Id: string (optional)

Identification string for a control.

> Inherited from ControlMetadata.Id

Label

Label: string (optional)

Label for a control. For example, a control representing a person's first name might have a label "First Name".

> Inherited from ControlMetadata.Label

Name

Name: string (optional)

Name of a control.

> Inherited from ControlMetadata.Name

Order

Order: number (optional)

Number indicating the order in which a control will appear on a page.

> Inherited from ControlMetadata.Order

Type

Type: ControlType (optional)

String indicating the control type.

> Inherited from ControlMetadata.Type

HyperLink type

2/18/2021 • 2 minutes to read • Edit Online

Hyperlink control type. Hyperlink control is a control to represent hyperlinks. Pagelinks can also be used in most cases.

Hierarchy

Value
└ HyperLink

Index

Properties

- container
- generic
- getDataSource
- hidden

Methods

- applyDesign
- dataContext
- getDesign
- getHyperLinkValue
- getValue
- isEditable
- isHyperLinkURLPresent
- metadata
- parent
- root
- setBaseURL
- setValue

Events

- onDataChanged

Properties

container

container: boolean (optional)

True if the control is a container.

> Inherited from Control.container

generic

generic: boolean (optional)

> Inherited from Control.generic

getDataSource

getDataSource: function(): any

> Inherited from Control.getDataSource

hidden

hidden: boolean

True if the control is hidden.

> Inherited from Control.hidden

Methods

applyDesign

applyDesign(IDesign: HyperLinkDesign): void

Applies given design to the design on the control. If a design already exists, the prototype chain of the design will be preserved.

> Overrides Control.applyDesign

Parameters

NAME	TYPE	DESCRIPTION
IDesign	HyperLinkDesign	object containing design properties as keys

Returns void

dataContext

dataContext(): any

> Inherited from Control.dataContext

Returns any

getDesign

getDesign(): Design

Returns the design object of this control.

> Inherited from Control.getDesign

Returns Design

getHyperLinkValue

getHyperLinkValue(): string

Returns string

getValue

getValue(): string

Returns the value of the control.

> Inherited from Value.getValue

Returns string

isEditable

isEditable(): boolean

Boolean indicating if the control is editable. Returns false when either the control or its parent is not editable. Returns true when both the control and its parent are editable. Returns true when either the control or its parent is editable and the other is undefined. Returns undefined if both the control's edit-ability and its parent's edit-ability is undefined.

> Inherited from Control.isEditable

Returns boolean

isHyperLinkURLPresent

isHyperLinkURLPresent(): boolean

Returns boolean

metadata

metadata(): HyperLinkMetadata

Returns the metadata object of this control.

> Overrides Value.metadata

Returns HyperLinkMetadata

parent

parent(): Control | Page

Returns the parent (control or page) of this control.

> Inherited from Control.parent

Returns Control | Page

root

root(): Page

Returns the root form instance (page) of this control.

> Inherited from Control.root

Returns Page

setBaseURL

setBaseURL(url: string): any

Parameters

NAME	TYPE	DESCRIPTION
url	string	

Returns any

setValue

setValue(value: string): void

Sets the value of the control.

> Inherited from Value.setValue

Parameters

NAME	TYPE	DESCRIPTION
value	string	

Returns void

Events

onDataChanged

onDataChanged: EventHook <null>

An event that is triggered when the input control's data changes.

> Inherited from InputControl.onDataChanged

HyperLinkDesign type

2/18/2021 • 2 minutes to read • Edit Online

Hyperlink design object type.

Hierarchy

ValueDesign
└ HyperLinkDesign

Index

Properties

Properties

alignItems

alignItems: string (optional)

This property is an alias for the CSS property "align-items". Please refer to this web page for documentation on the "align-items" property.

> Inherited from Design.alignItems

alignSelf

alignSelf: string (optional)

> Inherited from Design.alignSelf

bindings

bindings: any (optional)

> Inherited from Design.bindings

border

border: "none" | "solid" | "left" | "right" | "top" | "bottom" (optional)

The border behavior of a control. This property will not be inherited by the children.

> Inherited from Design.border

color

color: string (optional)

The foreground color of the container. This will modify the color of all headers, items, labels, and icons within the container.
Consider setting the background color at the same time as necessary when setting this attribute.
Note: if color is set to "theme", the theme color of the app will be used.
The following colors are available:

```
blue:          #0078D7;
pomegranate:   #911844;
raspberry:     #8D398F;
darkOrange:    #D24726;
green:         #369F47;
blueberry:     #17234E;
grape:         #432158;
lightBlue:     #5DB2FF;
lightGreen:    #82BA00;
pink:          #DC4FAD;
teal:          #008299;
mediumDarkBlue: #004B8B;
cordovan:      #570000;
darkCordovan:  #380000;
black:         #000000;
lightGray:     #e8e8e8;
light:         #fff   ;
dark:          #333333;
```

> Inherited from Design.color

flexFlow

flexFlow: string (optional)

Specifying this property makes the component a flex container component. This property is an alias for the CSS property "flex-flow". Please refer to this web page for documentation on the "flex-flow" property.

> Inherited from Design.flexFlow

flexSize

flexSize: string (optional)

One number or two numbers written as a string. For example, "(size to grow) [(size-to-shrink)]" to accommodate available space in the immediate flex container. This property is an alias for the CSS property "flex". Please refer to this web page for documentation on the "flex" property.

> Inherited from Design.flexSize

fontSize

fontSize: "medium" | "xx-small" | "x-small" | "small" | "large" | "x-large" | "xx-large" (optional)

The proportional text size

> Inherited from Design.fontSize

fontWeight

fontWeight: "normal" | "bold" (optional)

Normal or bold text.

> Inherited from Design.fontWeight

justifyItems

justifyItems: "flex-start" | "flex-end" | "center" | "space-between" (optional)

This property is an alias for the CSS property "justify-content". Please refer to this web page for documentation on the "justify-content" property.

> Inherited from Design.justifyItems

label

label: string (optional)

> Inherited from Design.label

labelPosition

labelPosition: "stacked" | "hidden" | "inline" (optional)

Determines how a label is positioned, if at all. By default, labelPosition is set to stacked.

> Inherited from Design.labelPosition

name

name: string (optional)

> Inherited from Design.name

padding

padding: "none" | "small" | "std" (optional)

Allows specifying the component's padding behavior. A component will inherit the padding behavior specified by its parent container components.

> Inherited from Design.padding

type

type: ControlType (optional)

The type of the control as a string.

> Inherited from Design.type

HyperLinkMetadata type

2/18/2021 • 2 minutes to read • Edit Online

Hyperlink metadata type.

Hierarchy

ValueMetadata
└ HyperLinkMetadata

Index

Properties

Properties

BoundEntity

BoundEntity: string (optional)

The entity to which the control is bound.

> Inherited from ControlMetadata.BoundEntity

BoundField

BoundField: string (optional)

> Inherited from ControlMetadata.BoundField

Description

Description: string (optional)

Description of the control.

> Inherited from ControlMetadata.Description

Editable

Editable: boolean (optional)

Boolean indicating if the control is editable. False when either the control or its parent is not editable. True when both the control and its parent are editable. True when either the control or its parent is editable and the other is undefined. Undefined if both the control's edit-ability and its parent's edit-ability is undefined.

> Inherited from ControlMetadata.Editable

ExtType

ExtType: ControlType (optional)

The extended control type. For example, a control of type Input might have an extended type of Barcode.

> Inherited from ControlMetadata.ExtType

HelpText

HelpText: string (optional)

The keyboard shortcut for a command. For example, "(Shift+F5)"

> Inherited from ControlMetadata.HelpText

Hidden

Hidden: boolean (optional)

Boolean indicating if the control is hidden or not.

> Inherited from ControlMetadata.Hidden

Id

Id: string (optional)

Identification string for a control.

> Inherited from ControlMetadata.Id

Label

Label: string (optional)

Label for a control. For example, a control representing a person's first name might have a label "First Name".

> Inherited from ControlMetadata.Label

Mandatory

Mandatory: boolean (optional)

If set to true then input for the control is required for the task to be completed. Mandatory controls will have a red outline.

> Inherited from InputControlMetadata.Mandatory

Name

Name: string (optional)

Name of a control.

> Inherited from ControlMetadata.Name

NumSequence

NumSequence: NumberSequenceConfig (optional)

Used for auto detecting and changing visibility of the number sequence controls in the task or page, based on AX number sequence configuration, through extended business logic. Example:

```
// hide number sequence reference page from users
metadataService.hideNavigation('numSeqReferencePage');

// parameters to be passed to 'numSequence' flag in configureControl
var configParam = {
     referencePageName: 'numSeqReferencePage',
     dataType: 'HcmPersonnelNumberId'
};

// setup 'PersonnelNumber' control as number sequence in the task 'add-worker'
metadataService.configureControl('add-worker', 'PersonnelNumber', { numSequence: configParam });
```

> Inherited from InputControlMetadata.NumSequence

Order

Order: number (optional)

Number indicating the order in which a control will appear on a page.

> Inherited from ControlMetadata.Order

Type

Type: ControlType (optional)

String indicating the control type.

> Inherited from ControlMetadata.Type

Image type

2/18/2021 • 2 minutes to read • Edit Online

Image control interface for representing images in the mobile app. Images can be of any of the following types: DataUri, Base64, URL, AOTResource, or Symbol.

Hierarchy

Control
└ Image

Index

Properties

- container
- generic
- getDataSource
- hidden
- imageSource
- imageView
- placeholderClass
- symbol

Methods

- applyDesign
- dataContext
- getDesign
- isEditable
- metadata
- parent
- root

Properties

container

container: boolean (optional)

True if the control is a container.

> Inherited from Control.container

generic

generic: boolean (optional)

> Inherited from Control.generic

getDataSource

getDataSource: function(): any

Inherited from Control.getDataSource

hidden

hidden: boolean

True if the control is hidden.

Inherited from Control.hidden

imageSource

imageSource: string

Defines the imageSource.

imageView

imageView: string

Dictates the style of the image.

placeholderClass

placeholderClass: string

symbol

symbol: string

Defines the symbol if the image is of type symbol.

Methods

applyDesign

applyDesign(design: ImageDesign): void

Applies given design to the design on the control. If a design already exists, the prototype chain of the design will be preserved.

Overrides Control.applyDesign

Parameters

NAME	TYPE	DESCRIPTION
design	ImageDesign	object containing design properties as keys

Returns void

dataContext

dataContext(): any

Inherited from Control.dataContext

Returns any

getDesign

getDesign(): Design

Returns the design object of this control.

Inherited from Control.getDesign

Returns Design

isEditable

isEditable(): boolean

Boolean indicating if the control is editable. Returns false when either the control or its parent is not editable. Returns true when both the control and its parent are editable. Returns true when either the control or its parent is editable and the other is undefined. Returns undefined if both the control's edit-ability and its parent's edit-ability is undefined.

Inherited from Control.isEditable

Returns boolean

metadata

metadata(): ImageMetadata

Returns the metadata object of this control.

Overrides Control.metadata

Returns ImageMetadata

parent

parent(): Control | Page

Returns the parent (control or page) of this control.

Inherited from Control.parent

Returns Control | Page

root

root(): Page

Returns the root form instance (page) of this control.

Inherited from Control.root

Returns Page

ImageDesign type

2/18/2021 • 2 minutes to read • Edit Online

Image design object type.

Hierarchy

Design
└ ImageDesign

Index

Properties

Properties

alignItems

alignItems: string (optional)

This property is an alias for the CSS property "align-items". Please refer to this web page for documentation on the "align-items" property.

> Inherited from Design.alignItems

alignSelf

alignSelf: string (optional)

> Inherited from Design.alignSelf

bindings

bindings: any (optional)

> Inherited from Design.bindings

border

border: "none" | "solid" | "left" | "right" | "top" | "bottom" (optional)

The border behavior of a control. This property will not be inherited by the children.

> Inherited from Design.border

color

color: string (optional)

The foreground color of the container. This will modify the color of all headers, items, labels, and icons within the container.
Consider setting the background color at the same time as necessary when setting this attribute.
Note: if color is set to "theme", the theme color of the app will be used.
The following colors are available:

```
blue:          #0078D7;
pomegranate:   #911844;
raspberry:     #8D398F;
darkOrange:    #D24726;
green:         #369F47;
blueberry:     #17234E;
grape:         #432158;
lightBlue:     #5DB2FF;
lightGreen:    #82BA00;
pink:          #DC4FAD;
teal:          #008299;
mediumDarkBlue:#004B8B;
cordovan:      #570000;
darkCordovan:  #380000;
black:         #000000;
lightGray:     #e8e8e8;
light:         #fff   ;
dark:          #333333;
```

> Inherited from Design.color

flexFlow

flexFlow: string (optional)

Specifying this property makes the component a flex container component. This property is an alias for the CSS property "flex-flow". Please refer to this web page for documentation on the "flex-flow" property.

> Inherited from Design.flexFlow

flexSize

flexSize: string (optional)

One number or two numbers written as a string. For example, "(size to grow) [(size-to-shrink)]" to accommodate available space in the immediate flex container. This property is an alias for the CSS property "flex". Please refer to this web page for documentation on the "flex" property.

> Inherited from Design.flexSize

fontSize

fontSize: "medium" | "xx-small" | "x-small" | "small" | "large" | "x-large" | "xx-large" (optional)

The proportional text size

> Inherited from Design.fontSize

fontWeight

fontWeight: "normal" | "bold" (optional)

Normal or bold text.

> Inherited from Design.fontWeight

height

height: string (optional)

The relative vertical size of the image. Sizes are about equivalent to CSS em sizes.

imageStyle

imageStyle: ImageStyleType (optional)

The style of the image.

justifyItems

justifyItems: "flex-start" | "flex-end" | "center" | "space-between" (optional)

This property is an alias for the CSS property "justify-content". Please refer to this web page for documentation on the "justify-content" property.

> Inherited from Design.justifyItems

label

label: string (optional)

> Inherited from Design.label

labelPosition

labelPosition: "stacked" | "hidden" | "inline" (optional)

Determines how a label is positioned, if at all. By default, labelPosition is set to stacked.

> Inherited from Design.labelPosition

name

name: string (optional)

> Inherited from Design.name

padding

padding: "none" | "small" | "std" (optional)

Allows specifying the component's padding behavior. A component will inherit the padding behavior specified by its parent container components.

> Inherited from Design.padding

type

type: ControlType (optional)

The type of the control as a string.

> Inherited from Design.type

width

width: string (optional)

The relative horizontal size of the image. Sizes are about equivalent to CSS em sizes.

ImageMetadata type

2/18/2021 • 2 minutes to read • Edit Online

Image metadata type.

Hierarchy

ControlMetadata
└ ImageMetadata

Index

Properties

- BaseUrl
- BoundEntity
- BoundField
- Description
- Editable
- ExtType
- Height
- HelpText
- Hidden
- Id
- ImageStyle
- Label
- Name
- Order
- Type
- Width

Properties

BaseUrl

BaseUrl: string (optional)

Base URL for AOTResource type image.

BoundEntity

BoundEntity: string (optional)

The entity to which the control is bound.

> Inherited from ControlMetadata.BoundEntity

BoundField

BoundField: string (optional)

> Inherited from ControlMetadata.BoundField

Description

Description: string (optional)

Description of the control.

> Inherited from ControlMetadata.Description

Editable

Editable: boolean (optional)

Boolean indicating if the control is editable. False when either the control or its parent is not editable. True when both the control and its parent are editable. True when either the control or its parent is editable and the other is undefined. Undefined if both the control's edit-ability and its parent's edit-ability is undefined.

> Inherited from ControlMetadata.Editable

ExtType

ExtType: ControlType (optional)

The extended control type. For example, a control of type Input might have an extended type of Barcode.

> Inherited from ControlMetadata.ExtType

Height

Height: number (optional)

The relative vertical size of the image. Sizes are about equivalent to CSS em sizes.

HelpText

HelpText: string (optional)

The keyboard shortcut for a command. For example, "(Shift+F5)"

> Inherited from ControlMetadata.HelpText

Hidden

Hidden: boolean (optional)

Boolean indicating if the control is hidden or not.

> Inherited from ControlMetadata.Hidden

Id

Id: string (optional)

Identification string for a control.

> Inherited from ControlMetadata.Id

ImageStyle

ImageStyle: ImageStyleType (optional)

The style of the image.

Label

Label: string (optional)

Label for a control. For example, a control representing a person's first name might have a label "First Name".

> Inherited from ControlMetadata.Label

Name

Name: string (optional)

Name of a control.

> Inherited from ControlMetadata.Name

Order

Order: number (optional)

Number indicating the order in which a control will appear on a page.

> Inherited from ControlMetadata.Order

Type

Type: ControlType (optional)

String indicating the control type.

> Inherited from ControlMetadata.Type

Width

Width: number (optional)

The relative horizontal size of the image. Sizes are about equivalent to CSS em sizes.

InputControl type

2/18/2021 • 2 minutes to read • Edit Online

Input control interface with methods and attributes for all input controls. Input controls are typically used on task pages for collecting user input, for example, for a new control.

Hierarchy

Control
  └ InputControl
    └ Field
    └ Lookup
    └ MultiLookup
    └ Value

Index

Properties

- container
- generic
- getDataSource
- hidden

Methods

- applyDesign
- dataContext
- getDesign
- isEditable
- metadata
- parent
- root

Events

- onDataChanged

Properties

container

container: boolean (optional)

True if the control is a container.

> Inherited from Control.container

generic

generic: boolean (optional)

> Inherited from Control.generic

getDataSource

getDataSource: function(): any

> Inherited from Control.getDataSource

hidden

hidden: boolean

True if the control is hidden.

> Inherited from Control.hidden

Methods

applyDesign

applyDesign(design: Design): void

Applies given design to the design on the control. If a design already exists, the prototype chain of the design will be preserved.

> Inherited from Control.applyDesign

Parameters

NAME	TYPE	DESCRIPTION
design	Design	object containing design properties as keys

Returns void

dataContext

dataContext(): any

> Inherited from Control.dataContext

Returns any

getDesign

getDesign(): Design

Returns the design object of this control.

> Inherited from Control.getDesign

Returns Design

isEditable

isEditable(): boolean

Boolean indicating if the control is editable. Returns false when either the control or its parent is not editable. Returns true when both the control and its parent are editable. Returns true when either the control or its parent is editable and the other is undefined. Returns undefined if both the control's edit-ability and its parent's edit-ability is undefined.

> Inherited from Control.isEditable

Returns boolean

metadata

metadata(): InputControlMetadata

Returns the metadata object of this control.

> Overrides Control.metadata

Returns InputControlMetadata

parent

parent(): Control | Page

Returns the parent (control or page) of this control.

> Inherited from Control.parent

Returns Control | Page

root

root(): Page

Returns the root form instance (page) of this control.

> Inherited from Control.root

Returns Page

Events

onDataChanged

onDataChanged: EventHook <null>

An event that is triggered when the input control's data changes.

InputControlDesign type

2/18/2021 • 2 minutes to read • Edit Online

Input control design.

Hierarchy

Design
└ InputControlDesign
└ FieldDesign
└ LookupDesign
└ MultiLookupDesign
└ ValueDesign

Index

Properties

Properties

alignItems

alignItems: string (optional)

This property is an alias for the CSS property "align-items". Please refer to this web page for documentation on the "align-items" property.

> Inherited from Design.alignItems

alignSelf

alignSelf: string (optional)

> Inherited from Design.alignSelf

bindings

bindings: any (optional)

> Inherited from Design.bindings

border

border: "none" | "solid" | "left" | "right" | "top" | "bottom" (optional)

The border behavior of a control. This property will not be inherited by the children.

> Inherited from Design.border

color

color: string (optional)

The foreground color of the container. This will modify the color of all headers, items, labels, and icons within the container.
Consider setting the background color at the same time as necessary when setting this attribute.
Note: if color is set to "theme", the theme color of the app will be used.
The following colors are available:

```
blue:           #0078D7;
pomegranate:    #911844;
raspberry:      #8D398F;
darkOrange:     #D24726;
green:          #369F47;
blueberry:      #17234E;
grape:          #432158;
lightBlue:      #5DB2FF;
lightGreen:     #82BA00;
pink:           #DC4FAD;
teal:           #008299;
mediumDarkBlue: #004B8B;
cordovan:       #570000;
darkCordovan:   #380000;
black:          #000000;
lightGray:      #e8e8e8;
light:          #fff   ;
dark:           #333333;
```

> Inherited from Design.color

flexFlow

flexFlow: string (optional)

Specifying this property makes the component a flex container component. This property is an alias for the CSS property "flex-flow". Please refer to this web page for documentation on the "flex-flow" property.

> Inherited from Design.flexFlow

flexSize

flexSize: string (optional)

One number or two numbers written as a string. For example, "(size to grow) [(size-to-shrink)]" to accommodate available space in the immediate flex container. This property is an alias for the CSS property "flex". Please refer to this web page for documentation on the "flex" property.

> Inherited from Design.flexSize

fontSize

fontSize: "medium" | "xx-small" | "x-small" | "small" | "large" | "x-large" | "xx-large" (optional)

The proportional text size

> Inherited from Design.fontSize

fontWeight

fontWeight: "normal" | "bold" (optional)

Normal or bold text.

> Inherited from Design.fontWeight

justifyItems

justifyItems: "flex-start" | "flex-end" | "center" | "space-between" (optional)

This property is an alias for the CSS property "justify-content". Please refer to this web page for documentation on the "justify-content" property.

> Inherited from Design.justifyItems

label

label: string (optional)

> Inherited from Design.label

labelPosition

labelPosition: "stacked" | "hidden" | "inline" (optional)

Determines how a label is positioned, if at all. By default, labelPosition is set to stacked.

> Inherited from Design.labelPosition

name

name: string (optional)

> Inherited from Design.name

padding

padding: "none" | "small" | "std" (optional)

Allows specifying the component's padding behavior. A component will inherit the padding behavior specified by its parent container components.

> Inherited from Design.padding

type

type: ControlType (optional)

The type of the control as a string.

> Inherited from Design.type

InputControlMetadata type

2/18/2021 • 2 minutes to read • Edit Online

Metadata for input controls.

Hierarchy

ControlMetadata
  └ InputControlMetadata
    └ FieldMetadata
    └ LookupMetadata
    └ MultiLookupMetadata
    └ ValueMetadata

Index

Properties

Properties

BoundEntity

BoundEntity: string (optional)

The entity to which the control is bound.

> Inherited from ControlMetadata.BoundEntity

BoundField

BoundField: string (optional)

> Inherited from ControlMetadata.BoundField

Description

Description: string (optional)

Description of the control.

> Inherited from ControlMetadata.Description

Editable

Editable: boolean (optional)

Boolean indicating if the control is editable. False when either the control or its parent is not editable. True when both the control and its parent are editable. True when either the control or its parent is editable and the other is undefined. Undefined if both the control's edit-ability and its parent's edit-ability is undefined.

> Inherited from ControlMetadata.Editable

ExtType

ExtType: ControlType (optional)

The extended control type. For example, a control of type Input might have an extended type of Barcode.

> Inherited from ControlMetadata.ExtType

HelpText

HelpText: string (optional)

The keyboard shortcut for a command. For example, "(Shift+F5)"

> Inherited from ControlMetadata.HelpText

Hidden

Hidden: boolean (optional)

Boolean indicating if the control is hidden or not.

> Inherited from ControlMetadata.Hidden

Id

Id: string (optional)

Identification string for a control.

> Inherited from ControlMetadata.Id

Label

Label: string (optional)

Label for a control. For example, a control representing a person's first name might have a label "First Name".

> Inherited from ControlMetadata.Label

Mandatory

Mandatory: boolean (optional)

If set to true then input for the control is required for the task to be completed. Mandatory controls will have a red outline.

Name

Name: string (optional)

Name of a control.

> Inherited from ControlMetadata.Name

NumSequence

NumSequence: NumberSequenceConfig (optional)

Used for auto detecting and changing visibility of the number sequence controls in the task or page, based on AX number sequence configuration, through extended business logic. Example:

```
// hide number sequence reference page from users
metadataService.hideNavigation('numSeqReferencePage');

// parameters to be passed to 'numSequence' flag in configureControl
var configParam = {
     referencePageName: 'numSeqReferencePage',
     dataType: 'HcmPersonnelNumberId'
};

// setup 'PersonnelNumber' control as number sequence in the task 'add-worker'
metadataService.configureControl('add-worker', 'PersonnelNumber', { numSequence: configParam });
```

Order

Order: number (optional)

Number indicating the order in which a control will appear on a page.

> Inherited from ControlMetadata.Order

Type

Type: ControlType (optional)

String indicating the control type.

> Inherited from ControlMetadata.Type

List type

2/18/2021 • 3 minutes to read • Edit Online

List control type. A list is a control that contains any numbers of rows. Each row follows a template for the layout of any number of controls. Lists come in two styles: simple and card.

Hierarchy

ContainerControl
└ List

Index

Properties

- $accessibility
- DefaultSearchColumn
- container
- emptyListMessage
- enableMultiSelect
- generic
- getDataSource
- hidden
- hideEmptyListMessage
- imageFields
- performingRemoteSearch
- searchQuery

Methods

- allowsNavigation
- applyDesign
- applySearch
- canPerformRemoteSearch
- clearSearch
- dataContext
- getColumnLabel
- getControl
- getControlById
- getControlMetadata
- getControlMetadataById
- getData
- getDesign
- getListData
- getRenderedRows
- getRowNavigation
- getRowSelectionCount
- getRowSelections

- getRowTracking
- getSearchColumn
- getSearchColumnLabel
- getSearchableColumns
- hideSearchBar
- isEditable
- loadMetaData
- loadMore
- metadata
- parent
- performRemoteSearch
- root
- selectSearchColumn
- setRowSections

Events

- onRowCreate
- onRowSelect

Properties

$accessibility

$accessibility: any

DefaultSearchColumn

DefaultSearchColumn: string

container

container: boolean

True if the control is a container.

> Inherited from ContainerControl.container
>
> Overrides Control.container

emptyListMessage

emptyListMessage: string

Settable property to override default empty list message.

enableMultiSelect

enableMultiSelect: boolean

generic

generic: boolean (optional)

> Inherited from Control.generic

getDataSource

getDataSource: function(): any

> Inherited from Control.getDataSource

hidden

hidden: boolean

True if the control is hidden.

> Inherited from Control.hidden

hideEmptyListMessage

hideEmptyListMessage: boolean

If true, no message is shown if the list is empty. To set this property, update the corresponding metadata property via configureControl.

imageFields

imageFields: any [ ]

performingRemoteSearch

performingRemoteSearch: boolean

searchQuery

searchQuery: [value: string]: any

Methods

allowsNavigation

allowsNavigation(): boolean

Returns boolean

applyDesign

applyDesign(IDesign: ListDesign): void

Applies given design to the design on the control. If a design already exists, the prototype chain of the design will be preserved.

> Overrides Control.applyDesign

Parameters

NAME	TYPE	DESCRIPTION
IDesign	ListDesign	object containing design properties as keys

Returns void

applySearch

applySearch(): void

Returns void

canPerformRemoteSearch

canPerformRemoteSearch(): boolean

Returns boolean

clearSearch

clearSearch(): void

Returns void

dataContext

dataContext(): any

> Inherited from Control.dataContext

Returns any

getColumnLabel

getColumnLabel(id: string): string

Parameters

NAME	TYPE	DESCRIPTION
id	string	

Returns string

getControl

getControl(controlName: string): Control

Given the name of a control, returns the control instance.

> Inherited from ContainerControl.getControl

Parameters

NAME	TYPE	DESCRIPTION
controlName	string	control name

Returns Control

getControlById

getControlById(id: string): Control

Given the ID of a control, returns the control instance.

> Inherited from ContainerControl.getControlById

Parameters

NAME	TYPE	DESCRIPTION
id	string	control ID

Returns Control

getControlMetadata

getControlMetadata(controlName: string): Control

Parameters

NAME	TYPE	DESCRIPTION
controlName	string	

Returns Control

getControlMetadataById

getControlMetadataById(id: string): Control

Parameters

NAME	TYPE	DESCRIPTION
id	string	

Returns Control

getData

getData(): any [ ]

Returns any [ ]

getDesign

getDesign(): Design

Returns the design object of this control.

> Inherited from Control.getDesign

Returns Design

getListData

getListData(): any

Returns any

getRenderedRows

getRenderedRows(): Row [ ]

Returns Row [ ]

getRowNavigation

getRowNavigation(row: Row): Promise <any> | any

Parameters

NAME	TYPE	DESCRIPTION
row	Row	

Returns Promise <any> | any

getRowSelectionCount

getRowSelectionCount(): number

Returns number

getRowSelections

getRowSelections(): string [ ]

Returns string [ ]

getRowTracking

getRowTracking(row: any, index: string): string

Parameters

NAME	TYPE	DESCRIPTION
row	any	
index	string	

Returns string

getSearchColumn

getSearchColumn(): string

Returns string

getSearchColumnLabel

getSearchColumnLabel(): string

Returns string

getSearchableColumns

getSearchableColumns(): any [ ]

Returns any [ ]

hideSearchBar

hideSearchBar(): boolean

Returns boolean

isEditable

isEditable(): boolean

Boolean indicating if the control is editable. Returns false when either the control or its parent is not editable. Returns true when both the control and its parent are editable. Returns true when either the control or its parent is editable and the other is undefined. Returns undefined if both the control's edit-ability and its parent's edit-ability is undefined.

> Inherited from Control.isEditable

Returns boolean

loadMetaData

loadMetaData(): void

Returns void

loadMore

loadMore(): void

Returns void

metadata

metadata(): ListMetadata

Returns the metadata object of this control.

> Overrides ContainerControl.metadata

Returns ListMetadata

parent

parent(): Control | Page

Returns the parent (control or page) of this control.

> Inherited from Control.parent

Returns Control | Page

performRemoteSearch

performRemoteSearch(): void

Returns void

root

root(): Page

Returns the root form instance (page) of this control.

> Inherited from Control.root

Returns Page

selectSearchColumn

selectSearchColumn(column: string): void

Parameters

NAME	TYPE	DESCRIPTION
column	string	

Returns void

setRowSections

setRowSections(selections: string [ ]): void

Parameters

NAME	TYPE	DESCRIPTION
selections	string [ ]	

Returns void

Events

onRowCreate

onRowCreate: EventHook <Row>

onRowSelect

onRowSelect: EventHook <Row>

ListDesign type

2/18/2021 • 3 minutes to read • Edit Online

List design object type.

Hierarchy

ContainerControlDesign
└ ListDesign

Index

Properties

Properties

alignItems

alignItems: string (optional)

This property is an alias for the CSS property "align-items". Please refer to this web page for documentation on the "align-items" property.

> Inherited from Design.alignItems

alignSelf

alignSelf: string (optional)

> Inherited from Design.alignSelf

allowScroll

allowScroll: string (optional)

True if the container will allow scrolling when its items do not fit into the container's available space. If a container has an item which may scroll, then set this property to false to prevent nested scrolling areas.

> Inherited from ContainerControlDesign.allowScroll

background

background: string (optional)

The background color of the container. Consider modifying the color attribute in the same container so that fonts overlaying the background color will appear appropriately. Note: if background is set to "theme", the theme color of the app will be used. The following colors are available:

```
blue:          #0078D7;
pomegranate:   #911844;
raspberry:     #8D398F;
darkOrange:    #D24726;
green:         #369F47;
blueberry:     #17234E;
grape:         #432158;
lightBlue:     #5DB2FF;
lightGreen:    #82BA00;
pink:          #DC4FAD;
teal:          #008299;
mediumDarkBlue: #004B8B;
cordovan:      #570000;
darkCordovan:  #380000;
black:         #000000;
lightGray:     #e8e8e8;
light:         #fff   ;
dark:          #333333;
```

> Inherited from ContainerControlDesign.background

bindings

bindings: any (optional)

> Inherited from Design.bindings

border

border: "none" | "solid" | "left" | "right" | "top" | "bottom" (optional)

The border behavior of a control. This property will not be inherited by the children.

> Inherited from Design.border

color

color: string (optional)

The foreground color of the container. This will modify the color of all headers, items, labels, and icons within the container.
Consider setting the background color at the same time as necessary when setting this attribute.
Note: if color is set to "theme", the theme color of the app will be used.
The following colors are available:

```
blue:          #0078D7;
pomegranate:   #911844;
raspberry:     #8D398F;
darkOrange:    #D24726;
green:         #369F47;
blueberry:     #17234E;
grape:         #432158;
lightBlue:     #5DB2FF;
lightGreen:    #82BA00;
pink:          #DC4FAD;
teal:          #008299;
mediumDarkBlue: #004B8B;
cordovan:      #570000;
darkCordovan:  #380000;
black:         #000000;
lightGray:     #e8e8e8;
light:         #fff   ;
dark:          #333333;
```

> Inherited from Design.color

design

design: GroupDesign (optional)

The design object that will be applied to each row.

flexFlow

flexFlow: string (optional)

Specifying this property makes the component a flex container component. This property is an alias for the CSS property "flex-flow". Please refer to this web page for documentation on the "flex-flow" property.

> Inherited from Design.flexFlow

flexSize

flexSize: string (optional)

One number or two numbers written as a string. For example, "(size to grow) [(size-to-shrink)]" to accommodate available space in the immediate flex container. This property is an alias for the CSS property "flex". Please refer to this web page for documentation on the "flex" property.

> Inherited from Design.flexSize

fontSize

fontSize: "medium" | "xx-small" | "x-small" | "small" | "large" | "x-large" | "xx-large" (optional)

The proportional text size

> Inherited from Design.fontSize

fontWeight

fontWeight: "normal" | "bold" (optional)

Normal or bold text.

> Inherited from Design.fontWeight

hideArrow

hideArrow: boolean (optional)

Allows an arrow (>) on a default styled navigation control to be hidden. Note that if the list has DetailsPageId, navigationHandler, or OnNavigate in the metadata then by default the arrows are present in each row of the list to show that the rows are clickable.

This property can only be added through the design object.

hideSearchBar

hideSearchBar: boolean (optional)

If true, the search bar will be hidden.

itemBorder

itemBorder: "solid" | "none" (optional)

If true, a border will appear around each row in the list. This property is equivalent to applying the border property individually to all items in the container.

> Inherited from ContainerControlDesign.itemBorder

items

items: string | Design [ ] (optional)

An array containing the components to place inside of the container.

> Inherited from ContainerControlDesign.items

justifyItems

justifyItems: "flex-start" | "flex-end" | "center" | "space-between" (optional)

This property is an alias for the CSS property "justify-content". Please refer to this web page for documentation

on the "justify-content" property.

> Inherited from Design.justifyItems

label

label: string (optional)

> Inherited from Design.label

labelPosition

labelPosition: "stacked" | "hidden" | "inline" (optional)

Determines how a label is positioned, if at all. By default, labelPosition is set to stacked.

> Inherited from Design.labelPosition

name

name: string (optional)

> Inherited from Design.name

padding

padding: "none" | "small" | "std" (optional)

Allows specifying the component's padding behavior. A component will inherit the padding behavior specified by its parent container components.

> Inherited from Design.padding

type

type: ControlType (optional)

The type of the control as a string.

> Inherited from Design.type

ListMetadata type

2/18/2021 • 2 minutes to read • Edit Online

Metadata for list control.

Hierarchy

ContainerControlMetadata
└ ListMetadata

Index

Properties

- BoundEntity
- BoundField
- Children
- Description
- DetailsPageAppId
- DetailsPageId
- Editable
- EmptyListMessage
- ExtType
- HelpText
- Hidden
- HideEmptyListMessage
- HideSearchBar
- Id
- InfiniteScroll
- InfiniteScrollPageSize
- Label
- ListStyle
- MultiSelect
- Name
- NonEntityProjection
- Order
- Type

Methods

- navigationHandler

Events

- OnNavigate
- OnRowSelect

Properties

BoundEntity

BoundEntity: string (optional)

The entity to which the control is bound.

> Inherited from ControlMetadata.BoundEntity

BoundField

BoundField: string (optional)

> Inherited from ControlMetadata.BoundField

Children

Children: ControlMetadata [ ] (optional)

List of metadata for controls that will appear in each row of the list.

Description

Description: string (optional)

Description of the control.

> Inherited from ControlMetadata.Description

DetailsPageAppId

DetailsPageAppId: string (optional)

App ID of the page that each row in the list will navigate to.

DetailsPageId

DetailsPageId: string (optional)

The ID of the page to which each row will navigate.

Editable

Editable: boolean (optional)

Boolean indicating if the control is editable. False when either the control or its parent is not editable. True when both the control and its parent are editable. True when either the control or its parent is editable and the other is undefined. Undefined if both the control's edit-ability and its parent's edit-ability is undefined.

> Inherited from ControlMetadata.Editable

EmptyListMessage

EmptyListMessage: string (optional)

If set, overrides the default message for empty lists.

ExtType

ExtType: ControlType (optional)

The extended control type. For example, a control of type Input might have an extended type of Barcode.

> Inherited from ControlMetadata.ExtType

HelpText

HelpText: string (optional)

The keyboard shortcut for a command. For example, "(Shift+F5)"

> Inherited from ControlMetadata.HelpText

Hidden

Hidden: boolean (optional)

Boolean indicating if the control is hidden or not.

> Inherited from ControlMetadata.Hidden

HideEmptyListMessage

HideEmptyListMessage: boolean (optional)

If true, the empty list message will be hidden.

HideSearchBar

HideSearchBar: boolean (optional)

If true, the search bar will be hidden.

Id

Id: string (optional)

Identification string for a control.

> Inherited from ControlMetadata.Id

InfiniteScroll

InfiniteScroll: boolean (optional)

If set to true then the list will allow infinite scroll.

InfiniteScrollPageSize

InfiniteScrollPageSize: number (optional)

Number of rows to load initially and the number of rows to load after the user reaches the end of the currently displayed rows.

Label

Label: string (optional)

Label for a control. For example, a control representing a person's first name might have a label "First Name".

> Inherited from ControlMetadata.Label

ListStyle

ListStyle: string (optional)

Dictates the list template type. Options:

- "Simple": simple style
- "Card": card style

MultiSelect

MultiSelect: boolean (optional)

If true, then the list will be a multi-select list.

Name

Name: string (optional)

Name of a control.

> Inherited from ControlMetadata.Name

NonEntityProjection

NonEntityProjection: boolean (optional)

Order

Order: number (optional)

Number indicating the order in which a control will appear on a page.

> Inherited from ControlMetadata.Order

Type

Type: ControlType (optional)

String indicating the control type.

> Inherited from ControlMetadata.Type

Methods

navigationHandler

Optional

navigationHandler(row: Row): Promise <any> | NavigationArgs

A function that determines the navigation for a given row.

Parameters

NAME	TYPE	DESCRIPTION
row	Row	row to get navigation handler for.

Returns Promise <any> | NavigationArgs

Events

OnNavigate

OnNavigate: function(navigation: NavigationArgs): any (optional)

An event that is triggered when a pagelink control is selected.

OnRowSelect

OnRowSelect: function(row: Row): void (optional)

An event that is triggered when a row is selected.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Lookup type

2/18/2021 • 2 minutes to read • Edit Online

Lookup control type. A lookup is an input control that is used to select an input from a list of options. For example, a lookup could be used to lookup a customer when linking a customer to a new sales order.

Hierarchy

InputControl
└ Lookup

Index

Properties

- container
- generic
- getDataSource
- hidden

Methods

- applyDesign
- dataContext
- getDesign
- getDisplayValue
- getLookupPage
- getValue
- isEditable
- metadata
- parent
- root
- setEntityRef

Events

- onDataChanged

Properties

container

container: boolean (optional)

True if the control is a container.

> Inherited from Control.container

generic

generic: boolean (optional)

> Inherited from Control.generic

getDataSource

getDataSource: function(): any

> Inherited from Control.getDataSource

hidden

hidden: boolean

True if the control is hidden.

> Inherited from Control.hidden

Methods

applyDesign

applyDesign(IDesign: LookupDesign): void

Applies given design to the design on the control. If a design already exists, the prototype chain of the design will be preserved.

> Overrides Control.applyDesign

Parameters

NAME	TYPE	DESCRIPTION
IDesign	LookupDesign	object containing design properties as keys

Returns void

dataContext

dataContext(): any

> Inherited from Control.dataContext

Returns any

getDesign

getDesign(): Design

Returns the design object of this control.

> Inherited from Control.getDesign

Returns Design

getDisplayValue

getDisplayValue(): string

Returns string

getLookupPage

getLookupPage(): Page

Returns Page

getValue

getValue(): string | number

Returns string | number

isEditable

isEditable(): boolean

Boolean indicating if the control is editable. Returns false when either the control or its parent is not editable. Returns true when both the control and its parent are editable. Returns true when either the control or its parent is editable and the other is undefined. Returns undefined if both the control's edit-ability and its parent's edit-ability is undefined.

> Inherited from Control.isEditable

Returns boolean

metadata

metadata(): LookupMetadata

Returns the metadata object of this control.

> Overrides InputControl.metadata

Returns LookupMetadata

parent

parent(): Control | Page

Returns the parent (control or page) of this control.

> Inherited from Control.parent

Returns Control | Page

root

root(): Page

Returns the root form instance (page) of this control.

> Inherited from Control.root

Returns Page

setEntityRef

setEntityRef(newValue: string | number): Promise <any>

Parameters

NAME	TYPE	DESCRIPTION
newValue	string | number	

Returns Promise <any>

Events

onDataChanged

onDataChanged: EventHook <null>

An event that is triggered when the input control's data changes.

> Inherited from InputControl.onDataChanged

LookupDesign type

2/18/2021 • 2 minutes to read • Edit Online

Lookup design object type.

Hierarchy

InputControlDesign
└ LookupDesign

Index

Properties

- alignItems
- alignSelf
- bindings
- border
- color
- flexFlow
- flexSize
- fontSize
- fontWeight
- justifyItems
- label
- labelPosition
- name
- padding
- type

Properties

alignItems

alignItems: string (optional)

This property is an alias for the CSS property "align-items". Please refer to this web page for documentation on the "align-items" property.

> Inherited from Design.alignItems

alignSelf

alignSelf: string (optional)

> Inherited from Design.alignSelf

bindings

bindings: any (optional)

> Inherited from Design.bindings

border

border: "none" | "solid" | "left" | "right" | "top" | "bottom" (optional)

The border behavior of a control. This property will not be inherited by the children.

> Inherited from Design.border

color

color: string (optional)

The foreground color of the container. This will modify the color of all headers, items, labels, and icons within the container.
Consider setting the background color at the same time as necessary when setting this attribute.
Note: if color is set to "theme", the theme color of the app will be used.
The following colors are available:

```
blue:          #0078D7;
pomegranate:   #911844;
raspberry:     #8D398F;
darkOrange:    #D24726;
green:         #369F47;
blueberry:     #17234E;
grape:         #432158;
lightBlue:     #5DB2FF;
lightGreen:    #82BA00;
pink:          #DC4FAD;
teal:          #008299;
mediumDarkBlue: #004B8B;
cordovan:      #570000;
darkCordovan:  #380000;
black:         #000000;
lightGray:     #e8e8e8;
light:         #fff   ;
dark:          #333333;
```

> Inherited from Design.color

flexFlow

flexFlow: string (optional)

Specifying this property makes the component a flex container component. This property is an alias for the CSS property "flex-flow". Please refer to this web page for documentation on the "flex-flow" property.

> Inherited from Design.flexFlow

flexSize

flexSize: string (optional)

One number or two numbers written as a string. For example, "(size to grow) [(size-to-shrink)]" to accommodate available space in the immediate flex container. This property is an alias for the CSS property "flex". Please refer to this web page for documentation on the "flex" property.

> Inherited from Design.flexSize

fontSize

fontSize: "medium" | "xx-small" | "x-small" | "small" | "large" | "x-large" | "xx-large" (optional)

The proportional text size

> Inherited from Design.fontSize

fontWeight

fontWeight: "normal" | "bold" (optional)

Normal or bold text.

> Inherited from Design.fontWeight

justifyItems

justifyItems: "flex-start" | "flex-end" | "center" | "space-between" (optional)

This property is an alias for the CSS property "justify-content". Please refer to this web page for documentation on the "justify-content" property.

> Inherited from Design.justifyItems

label

label: string (optional)

> Inherited from Design.label

labelPosition

labelPosition: "stacked" | "hidden" | "inline" (optional)

Determines how a label is positioned, if at all. By default, labelPosition is set to stacked.

> Inherited from Design.labelPosition

name

name: string (optional)

> Inherited from Design.name

padding

padding: "none" | "small" | "std" (optional)

Allows specifying the component's padding behavior. A component will inherit the padding behavior specified by its parent container components.

> Inherited from Design.padding

type

type: ControlType (optional)

The type of the control as a string.

> Inherited from Design.type

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

LookupMetadata type

2/18/2021 • 2 minutes to read • Edit Online

Lookup metadata type.

Hierarchy

InputControlMetadata
└ LookupMetadata

Index

Properties

- BoundEntity
- BoundField
- Description
- DisplayField
- DisplayKey
- Editable
- ExtType
- FilterContext
- HelpText
- Hidden
- Id
- Label
- LookupEntity
- LookupPage
- LookupPageId
- Mandatory
- MultiSelect
- Name
- NumSequence
- Order
- ReferenceAppId
- ShowLookupPage
- Type
- ValueField
- ValueKey

Events

- OnOptionSelected
- OnValueChanged

Properties

BoundEntity

BoundEntity: string (optional)

The entity to which the control is bound.

> Inherited from ControlMetadata.BoundEntity

BoundField

BoundField: string (optional)

> Inherited from ControlMetadata.BoundField

Description

Description: string (optional)

Description of the control.

> Inherited from ControlMetadata.Description

DisplayField

DisplayField: string (optional)

The name of a control on the page, whose value should be displayed to the user. Usually, this value is user-friendly/user-readable text.

DisplayKey

DisplayKey: string (optional)

Editable

Editable: boolean (optional)

Boolean indicating if the control is editable. False when either the control or its parent is not editable. True when both the control and its parent are editable. True when either the control or its parent is editable and the other is undefined. Undefined if both the control's edit-ability and its parent's edit-ability is undefined.

> Inherited from ControlMetadata.Editable

ExtType

ExtType: ControlType (optional)

The extended control type. For example, a control of type Input might have an extended type of Barcode.

> Inherited from ControlMetadata.ExtType

FilterContext

FilterContext: DataFilter (optional)

HelpText

HelpText: string (optional)

The keyboard shortcut for a command. For example, "(Shift+F5)"

> Inherited from ControlMetadata.HelpText

Hidden

Hidden: boolean (optional)

Boolean indicating if the control is hidden or not.

> Inherited from ControlMetadata.Hidden

Id

Id: string (optional)

Identification string for a control.

> Inherited from ControlMetadata.Id

Label

Label: string (optional)

Label for a control. For example, a control representing a person's first name might have a label "First Name".

> Inherited from ControlMetadata.Label

LookupEntity

LookupEntity: any (optional)

The entity that is being looked up in the lookup.

LookupPage

LookupPage: string (optional)

LookupPageId

LookupPageId: string (optional)

Mandatory

Mandatory: boolean (optional)

If set to true then input for the control is required for the task to be completed. Mandatory controls will have a red outline.

> Inherited from InputControlMetadata.Mandatory

MultiSelect

MultiSelect: boolean (optional)

If true, lookup will be configured as a multi-select.

Name

Name: string (optional)

Name of a control.

> Inherited from ControlMetadata.Name

NumSequence

NumSequence: NumberSequenceConfig (optional)

Used for auto detecting and changing visibility of the number sequence controls in the task or page, based on AX number sequence configuration, through extended business logic. Example:

```
// hide number sequence reference page from users
metadataService.hideNavigation('numSeqReferencePage');

// parameters to be passed to 'numSequence' flag in configureControl
var configParam = {
     referencePageName: 'numSeqReferencePage',
     dataType: 'HcmPersonnelNumberId'
};

// setup 'PersonnelNumber' control as number sequence in the task 'add-worker'
metadataService.configureControl('add-worker', 'PersonnelNumber', { numSequence: configParam });
```

> Inherited from InputControlMetadata.NumSequence

Order

Order: number (optional)

Number indicating the order in which a control will appear on a page.

> Inherited from ControlMetadata.Order

ReferenceAppId

ReferenceAppId: string (optional)

ShowLookupPage

ShowLookupPage: boolean (optional)

Type

Type: ControlType (optional)

String indicating the control type.

> Inherited from ControlMetadata.Type

ValueField

ValueField: string (optional)

The name of a control on the page, whose value should be used when committing the data. Usually, this value is a unique key.

ValueKey

ValueKey: string (optional)

Events

OnOptionSelected

OnOptionSelected: function(lookup: any, lookupEntityData: any): void (optional)

An event that is triggered by an option being selected.

OnValueChanged

OnValueChanged: function(value: any): void (optional)

An event that is triggered by a value being changed.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

MetadataService type

2/18/2021 • 4 minutes to read • Edit Online

Provides ability to access and configure various metadata elements under the application workspace.

Hierarchy

MetadataService

Index

Properties

- version

Methods

- addControl
- compareVersion
- configureAction
- configureControl
- configureEntity
- configureLookup
- configurePage
- configureWorkspace
- findAction
- findControl
- findPage
- getFilterExpression
- getFormReference
- hideNavigation

Properties

version

version: string

(Read-only) Gets the version of the platform currently running.

Methods

addControl

addControl(componentName: string, controlName: string, controlType: ControlType, parentContainerName?: string, options?: ControlMetadata): any

Parameters

NAME	TYPE	DESCRIPTION
componentName	string	

NAME	TYPE	DESCRIPTION
controlName	string	
controlType	ControlType	
parentContainerName?	string	
options?	ControlMetadata	

Returns any

compareVersion

compareVersion(versionToCompare: string): 1 | -1

Compares the current platform version with a reference version.

Parameters

NAME	TYPE	DESCRIPTION
versionToCompare	string	The reference version to compare with

Returns 1 | -1

1 to indicate the platform version is older than the reference version, -1 to indicate that the platform version is newer or same as the reference version

configureAction

configureAction(actionName: string, options: PageMetadata): any

Configuring an action allows specifying or overriding certain behaviors specific to actions. Example:

```
metadataService.configureAction('Edit-Reservation', { properties-to-set });
```

Parameters

NAME	TYPE	DESCRIPTION
actionName	string	The action whose behavior is to be changed
options	PageMetadata	The property bag containing the properties to set on the action

Returns any

configureControl

configureControl(componentName: string, controlName: string, options: ControlMetadata): any

Configuring a control allows specifying or overriding certain behaviors specific to the control. Note that the available behaviors vary by control type. Example:

```
metadataService.configureControl('All-Customers', 'FMCustomer_RecId', { properties-to-set });
```

Parameters

NAME	TYPE	DESCRIPTION
componentName	string	A page or action that contains the control
controlName	string	The control whose behavior is to be changed
options	ControlMetadata	The property bag containing the properties to set on the control

Returns any

configureEntity

configureEntity(entityName: string, options: any): any

Configuring an entity allows specifying or overriding certain behaviors specific to the entity. Example:

```
metadataService.configureEntity("FMCustomer", { properties-to-set });
```

Parameters

NAME	TYPE	DESCRIPTION
entityName	string	An entity name
options	any	The property bag containing the properties to set on the entity

Returns any

configureLookup

configureLookup(taskName: string, lookupControlName: string, options: LookupMetadata): any

Configures a field on an action to behave as a lookup. Requires using an existing page which contains a list control. Example:

```
metadataService.configureLookup('Add-Reservation', 'FMRental_Customer', { lookupPage: 'All-Customers',
valueField: 'FMCustomer_RecId', displayField: 'FMCustomer_FullName'});
```

Parameters

NAME	TYPE	DESCRIPTION
taskName	string	Action name
lookupControlName	string	The control name of the field to be given lookup behavior
options	LookupMetadata	Lookup configuration object

Returns any

configurePage

configurePage(pageName: string, options: PageMetadata): any

Configuring a Page allows specifying or overriding certain behaviors specific to the Page. Example:

```
metadataService.configurePage('Reservation-details', { properties-to-set });
```

Parameters

NAME	TYPE	DESCRIPTION
pageName	string	The page that contains the control
options	PageMetadata	The property bag containing the properties to set on the page

Returns any

configureWorkspace

configureWorkspace(options: PageMetadata): any

Configuring a workspace allows specifying or overriding certain behaviors specific to the workspace. Example:

```
metadataService.configureWorkspace({ properties-to-set });
```

Parameters

NAME	TYPE	DESCRIPTION
options	PageMetadata	The property bag containing the properties to set on the workspace

Returns any

findAction

findAction(actionName: string): PageMetadata

Gets a copy of the current metadata instance of a specified Action, for the purpose of inspecting the metadata (not to be used for changing the metadata). Note: Since metadata can be changed at any time by business logic, you must be mindful of when you use this API to get a copy as it will reflect the state of the metadata at the time the call is made.

Example:

```
var newCustomerTaskMetadata = metadataService.findTask("New-customer");
```

Parameters

NAME	TYPE	DESCRIPTION
actionName	string	An action name

Returns PageMetadata

findControl

findControl(componentMetadata: any, controlName: string): ControlMetadata

Gets a copy of the current metadata instance of a specified control, for the purpose of inspecting the metadata (not to be used for changing the metadata). Note: Since metadata can be changed at any time by business logic, you must be mindful of when you use this API to get a copy as it will reflect the state of the metadata at the time the call is made.

Example:

```
var firstNameControl = metadataService.findControl(newCustomerTaskMetadata, 'FMCustomer_FirstName');
```

Parameters

NAME	TYPE	DESCRIPTION
componentMetadata	any	A metadata instance of the page or action
controlName	string	A control name

Returns ControlMetadata

findPage

findPage(pageName: string): PageMetadata

Gets a copy of the current metadata instance of a specified page, for the purpose of inspecting the metadata (not to be used for changing the metadata). Note: Since metadata can be changed at any time by business logic, you must be mindful of when you use this API to get a copy as it will reflect the state of the metadata at the time the call is made.

Example:

```
var reservationDetailsMetadata = metadataService.findPage("Reservation-details");
```

Parameters

NAME	TYPE	DESCRIPTION
pageName	string	A page name

Returns PageMetadata

getFilterExpression

getFilterExpression(pageName: string, listControlName: string, controlName: string, operator: ExpressionOperator, value: string): DataFilter

Create a DataFilter object for a list control based on the provided options. Example:

```
var filter = metadataService.getFilterExpression(
 pageNames.AllCustomers, controlNames.CustomerList, controlNames.CustomerFullName, "Is", firstCustomerName),
```

Parameters

NAME	TYPE	DESCRIPTION
pageName	string	
listControlName	string	
controlName	string	
operator	ExpressionOperator	

NAME	TYPE	DESCRIPTION
value	string	

Returns DataFilter

getFormReference

getFormReference(componentName: string, filterContext: DataFilter, excludeContext: boolean, filterLocalOnly?: boolean): NavigationArgs

Create an INavigationArgs object for a specific page/action to be used with a navigation control.

Parameters

NAME	TYPE	DESCRIPTION
componentName	string	Name of the action/page
filterContext	DataFilter	
excludeContext	boolean	
filterLocalOnly?	boolean	

Returns NavigationArgs

hideNavigation

hideNavigation(pageNamesToHide: string [ ]): any

Hides the specified page(s) from the default landing page. Example:

```
metadataService.hideNavigation('Select-a-customer', 'Select-a-vehicle');
```

Parameters

NAME	TYPE	DESCRIPTION
pageNamesToHide	string [ ]	Page name(s)

Returns any

MultiLookup type

2/18/2021 • 2 minutes to read • Edit Online

Multi-Lookup control type. Multi-Lookup controls are similar to regular lookups except they allow multiple selections at once.

Hierarchy

InputControl
└ MultiLookup

Index

Properties

- container
- generic
- getDataSource
- getEntityRefs
- hidden
- setEntityRefs

Methods

- applyDesign
- dataContext
- getDesign
- getLookupPage
- isEditable
- metadata
- parent
- root

Events

- onDataChanged

Properties

container

container: boolean (optional)

True if the control is a container.

> Inherited from Control.container

generic

generic: boolean (optional)

> Inherited from Control.generic

getDataSource

getDataSource: function(): any

> Inherited from Control.getDataSource

getEntityRefs

getEntityRefs: function(): string [ ] | number [ ]

hidden

hidden: boolean

True if the control is hidden.

> Inherited from Control.hidden

setEntityRefs

setEntityRefs: function(ids: string [ ] | number [ ]): Promise <any>

Methods

applyDesign

applyDesign(IDesign: MultiLookupDesign): void

Applies given design to the design on the control. If a design already exists, the prototype chain of the design will be preserved.

> Overrides Control.applyDesign

Parameters

NAME	TYPE	DESCRIPTION
IDesign	MultiLookupDesign	object containing design properties as keys

Returns void

dataContext

dataContext(): any

> Inherited from Control.dataContext

Returns any

getDesign

getDesign(): Design

Returns the design object of this control.

> Inherited from Control.getDesign

Returns Design

getLookupPage

getLookupPage(): Page

Returns Page

isEditable

isEditable(): boolean

Boolean indicating if the control is editable. Returns false when either the control or its parent is not editable. Returns true when both the control and its parent are editable. Returns true when either the control or its parent is editable and the other is undefined. Returns undefined if both the control's edit-ability and its parent's edit-ability is undefined.

> Inherited from Control.isEditable

Returns boolean

metadata

metadata(): MultiLookupMetadata

Returns the metadata object of this control.

> Overrides InputControl.metadata

Returns MultiLookupMetadata

parent

parent(): Control | Page

Returns the parent (control or page) of this control.

> Inherited from Control.parent

Returns Control | Page

root

root(): Page

Returns the root form instance (page) of this control.

> Inherited from Control.root

Returns Page

Events

onDataChanged

onDataChanged: EventHook <null>

An event that is triggered when the input control's data changes.

> Inherited from InputControl.onDataChanged

MultiLookupDesign type

2/18/2021 • 2 minutes to read • Edit Online

Multi-Lookup design object type.

Hierarchy

InputControlDesign
└ MultiLookupDesign

Index

Properties

Properties

alignItems

alignItems: string (optional)

This property is an alias for the CSS property "align-items". Please refer to this web page for documentation on the "align-items" property.

> Inherited from Design.alignItems

alignSelf

alignSelf: string (optional)

> Inherited from Design.alignSelf

bindings

bindings: any (optional)

> Inherited from Design.bindings

border

border: "none" | "solid" | "left" | "right" | "top" | "bottom" (optional)

The border behavior of a control. This property will not be inherited by the children.

> Inherited from Design.border

color

color: string (optional)

The foreground color of the container. This will modify the color of all headers, items, labels, and icons within the container.
Consider setting the background color at the same time as necessary when setting this attribute.
Note: if color is set to "theme", the theme color of the app will be used.
The following colors are available:

```
blue:          #0078D7;
pomegranate:   #911844;
raspberry:     #8D398F;
darkOrange:    #D24726;
green:         #369F47;
blueberry:     #17234E;
grape:         #432158;
lightBlue:     #5DB2FF;
lightGreen:    #82BA00;
pink:          #DC4FAD;
teal:          #008299;
mediumDarkBlue: #004B8B;
cordovan:      #570000;
darkCordovan:  #380000;
black:         #000000;
lightGray:     #e8e8e8;
light:         #fff   ;
dark:          #333333;
```

> Inherited from Design.color

flexFlow

flexFlow: string (optional)

Specifying this property makes the component a flex container component. This property is an alias for the CSS property "flex-flow". Please refer to this web page for documentation on the "flex-flow" property.

> Inherited from Design.flexFlow

flexSize

flexSize: string (optional)

One number or two numbers written as a string. For example, "(size to grow) [(size-to-shrink)]" to accommodate available space in the immediate flex container. This property is an alias for the CSS property "flex". Please refer to this web page for documentation on the "flex" property.

> Inherited from Design.flexSize

fontSize

fontSize: "medium" | "xx-small" | "x-small" | "small" | "large" | "x-large" | "xx-large" (optional)

The proportional text size

> Inherited from Design.fontSize

fontWeight

fontWeight: "normal" | "bold" (optional)

Normal or bold text.

> Inherited from Design.fontWeight

justifyItems

justifyItems: "flex-start" | "flex-end" | "center" | "space-between" (optional)

This property is an alias for the CSS property "justify-content". Please refer to this web page for documentation on the "justify-content" property.

> Inherited from Design.justifyItems

label

label: string (optional)

> Inherited from Design.label

labelPosition

labelPosition: "stacked" | "hidden" | "inline" (optional)

Determines how a label is positioned, if at all. By default, labelPosition is set to stacked.

> Inherited from Design.labelPosition

name

name: string (optional)

> Inherited from Design.name

padding

padding: "none" | "small" | "std" (optional)

Allows specifying the component's padding behavior. A component will inherit the padding behavior specified by its parent container components.

> Inherited from Design.padding

type

type: ControlType (optional)

The type of the control as a string.

> Inherited from Design.type

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

MultiLookupMetadata type

2/18/2021 • 2 minutes to read • Edit Online

Multi-Lookup metadata type.

Hierarchy

InputControlMetadata
└ MultiLookupMetadata

Index

Properties

- BoundEntity
- BoundField
- Description
- Design
- Editable
- ExtType
- FilterContext
- FilterLocalOnly
- HelpText
- Hidden
- Id
- Label
- LookupPageId
- Mandatory
- Name
- NumSequence
- Order
- ReferenceAppId
- ReverseLookupRelation
- ShowPending
- Type

Events

- OnLookupPageCreate
- OnLookupPageCreated

Properties

BoundEntity

BoundEntity: string (optional)

The entity to which the control is bound.

> Inherited from ControlMetadata.BoundEntity

BoundField

BoundField: string (optional)

> Inherited from ControlMetadata.BoundField

Description

Description: string (optional)

Description of the control.

> Inherited from ControlMetadata.Description

Design

Design: Design (optional)

Design object for the lookup page that is referenced by the LookupPageId.

Editable

Editable: boolean (optional)

Boolean indicating if the control is editable. False when either the control or its parent is not editable. True when both the control and its parent are editable. True when either the control or its parent is editable and the other is undefined. Undefined if both the control's edit-ability and its parent's edit-ability is undefined.

> Inherited from ControlMetadata.Editable

ExtType

ExtType: ControlType (optional)

The extended control type. For example, a control of type Input might have an extended type of Barcode.

> Inherited from ControlMetadata.ExtType

FilterContext

FilterContext: DataFilter (optional)

FilterLocalOnly

FilterLocalOnly: boolean (optional)

HelpText

HelpText: string (optional)

The keyboard shortcut for a command. For example, "(Shift+F5)"

> Inherited from ControlMetadata.HelpText

Hidden

Hidden: boolean (optional)

Boolean indicating if the control is hidden or not.

> Inherited from ControlMetadata.Hidden

Id

Id: string (optional)

Identification string for a control.

> Inherited from ControlMetadata.Id

Label

Label: string (optional)

Label for a control. For example, a control representing a person's first name might have a label "First Name".

> Inherited from ControlMetadata.Label

LookupPageId

LookupPageId: string (optional)

Page that is hosted within the multi-lookup.

Mandatory

Mandatory: boolean (optional)

If set to true then input for the control is required for the task to be completed. Mandatory controls will have a red outline.

> Inherited from InputControlMetadata.Mandatory

Name

Name: string (optional)

Name of a control.

> Inherited from ControlMetadata.Name

NumSequence

NumSequence: NumberSequenceConfig (optional)

Used for auto detecting and changing visibility of the number sequence controls in the task or page, based on AX number sequence configuration, through extended business logic. Example:

```
// hide number sequence reference page from users
metadataService.hideNavigation('numSeqReferencePage');

// parameters to be passed to 'numSequence' flag in configureControl
var configParam = {
     referencePageName: 'numSeqReferencePage',
     dataType: 'HcmPersonnelNumberId'
};

// setup 'PersonnelNumber' control as number sequence in the task 'add-worker'
metadataService.configureControl('add-worker', 'PersonnelNumber', { numSequence: configParam });
```

> Inherited from InputControlMetadata.NumSequence

Order

Order: number (optional)

Number indicating the order in which a control will appear on a page.

> Inherited from ControlMetadata.Order

ReferenceAppId

ReferenceAppId: string (optional)

ReverseLookupRelation

ReverseLookupRelation: boolean (optional)

ShowPending

ShowPending: boolean (optional)

Type

Type: ControlType (optional)

String indicating the control type.

> Inherited from ControlMetadata.Type

Events

OnLookupPageCreate

OnLookupPageCreate: function(args: any, multiLookup: any): void (optional)

OnLookupPageCreated

OnLookupPageCreated: function(args: any, multiLookup: any): void (optional)

NavigationArgs type

2/18/2021 • 2 minutes to read • Edit Online

Hierarchy

PageTarget
└ NavigationArgs

Index

Properties

- label
- options
- params
- replace
- to
- url

Properties

label

label: string (optional)

options

options: any (optional)

params

params: PageOptions (optional)

> Inherited from PageTarget.params

replace

replace: boolean (optional)

If set to true, removes current view firing navigation from navigation history stack.

to

to: string (optional)

> Inherited from PageTarget.to

url

url: string (optional)

If provided, this link is directly opened.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

NumberSequenceConfig type

2/18/2021 • 2 minutes to read • Edit Online

Number Sequence Configuration type.

Hierarchy

NumberSequenceConfig

Index

Properties

Properties

dataType

dataType: string

The data type is used to lookup whether the number sequence is editable or not on the reference page.

referencePageName

referencePageName: string

Page name of the page that defines if the num sequence is editable.

Page type

2/18/2021 • 3 minutes to read • Edit Online

Page object type.

Hierarchy

Page

Index

Properties

Methods

Events

Properties

children

children: Control [ ]

(Read-only) The list of all direct children controls of the page.

dataLoadedInitially

dataLoadedInitially: Promise <void>

(Read-only) A promise which resolves when the data has loaded for the first time. The promise continues to stay resolved for the rest of the page life.

initialized

initialized: boolean

(Read-only) True if the page instance has been initialized.

metadata

metadata: PageMetadata

(Read-only) The page metadata.

metadataLoaded

metadataLoaded: Promise <void>

(Read-only) A promise which resolves when the metadata has finished loading.

pageContext

pageContext: string

The current page context.

pageFilter

pageFilter: DataFilter

The current filter applied on the page.

state

state: PageState

(Read-only) The current state of the page.

syncError

syncError: boolean

(Read-only) True if the page's submission is in error state. This normally happens when the server rejects submissions due to validation errors. Refer to this topic for a detailed explanation of page data synchronization.

syncPending

syncPending: boolean

(Read-only) True if the page's submission is waiting to be synced. Refer to this topic for a detailed explanation of page data synchronization.

syncProcessing

syncProcessing: boolean

(Read-only) True if the page instance is currently syncing its submission. Refer to this topic for a detailed explanation of page data synchronization.

syncUnitEditable

syncUnitEditable: boolean

(Read-only) True if it's possible to edit a submission while it's waiting to be synchronized. Refer to this topic for a detailed explanation of page data synchronization.

title

title: string

(Read-only) The title of the page.

Methods

canSubmit

canSubmit(): boolean

Returns true if action page can be submitted and there are no validation/error messages.

Returns boolean

close

close(): void

Dispose the page instance and all its lifecycle events.

Returns void

getAction

getAction(actionName: string): PageLink

Get a page action by name. These include the actions in the action sheet/menu.

Parameters

NAME	TYPE	DESCRIPTION
actionName	string	

Returns PageLink

getActions

getActions(): PageLink [ ]

Get all page actions. These include the actions in the action sheet/menu.

Returns PageLink [ ]

getControl

getControl(controlName: string): Control

Get a page control by name. It recursively searches through all its children pages.

Parameters

NAME	TYPE	DESCRIPTION
controlName	string	

Returns Control

getDesign

getDesign(): Design

Get the design object associated with the page.

Returns Design

getEntityContext

getEntityContext(): EntityRef

Get current entity context.

Returns EntityRef

isEditable

isEditable(): boolean

Returns true if the page is an Action Page

Returns boolean

refreshData

refreshData(): Promise <void>

Force refresh page data.

Returns Promise <void>

resume

resume(): Promise <void>

Resume a temporarily suspended page.

Returns Promise <void>

submit

submit(): Promise <CompleteEventArgs>

Submit an Action.

Returns Promise <CompleteEventArgs>

suspend

suspend(): void

Temporarily suspend a page. For example, when the page is not the active view.

Returns void

Events

onClose

onClose: EventHook <null>

Event that is raised when a page is closed.

onComplete

onComplete: EventHook <any>

Event that is raised when an action is completed.

onDataLoaded

onDataLoaded: EventHook <any>

Event that fires when the page data has loaded. The event may be fired multiple times - every time new data is loaded.

onInit

onInit: EventHook <any>

Event that fires when a page instance has been initialized, and the metadata has been loaded.

onPreInit

onPreInit: EventHook <any>

Event that fires when a page instance has been initialized. This is fired before the metadata has been loaded.

onRefresh

onRefresh: EventHook <null>

Event that fires on forced page refresh, before new data has been loaded.

onStateChange

onStateChange: EventHook <null>

Event that fires when the page state changes.

onSubmit

onSubmit: EventHook <PageSubmitArgs>

Event that fires before an action is submitted. It can be intercepted for action validation/deferring Refer to IPageSubmitArgs to know more about the available options.

onSyncStatusChange

onSyncStatusChange: EventHook <null>

Event that fires when the page sync status changes.

PageData type

2/18/2021 • 2 minutes to read • Edit Online

Represents the data that is loaded into a page.

Hierarchy

PageData

Index

Methods

- getControlValue
- setControlValue

Methods

getControlValue

getControlValue(controlName: string): any

Gets the value of a control directly from the data set loaded in the page. The "value" is loosely defined across all different types of controls and generally indicates the primary single field value displayed or interacted with the control. Some complex controls (e.g a lookup or a list) may not have a simple value and thus cannot be accessed via this API.

Parameters

NAME	TYPE	DESCRIPTION
controlName	string	name of the control whose value is to be retrieved

Returns any

setControlValue

setControlValue(controlName: string, value: any): any

Sets the value of a control directly into the data set loaded in the page. The "value" is loosely defined across all different types of controls and generally indicates the primary single field value displayed or interacted with the control. Some complex controls (e.g a lookup or a list) may not have a simple value and thus cannot be accessed via this API.

Parameters

NAME	TYPE	DESCRIPTION
controlName	string	Name of the control whose value is to be set
value	any	The value to be set

Returns any

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

PageLink type

2/18/2021 • 2 minutes to read • Edit Online

Pagelink control type. A pagelink is a control that navigates to another page.

Hierarchy

Control
└ PageLink

Index

Properties

- container
- generic
- getDataSource
- hidden

Methods

- allowsNavigation
- applyDesign
- dataContext
- getCount
- getDesign
- getNavigationHandler
- isEditable
- metadata
- parent
- root
- showCount

Properties

container

container: boolean (optional)

True if the control is a container.

> Inherited from Control.container

generic

generic: boolean (optional)

> Inherited from Control.generic

getDataSource

getDataSource: function(): any

> Inherited from Control.getDataSource

hidden

hidden: boolean

True if the control is hidden.

> Inherited from Control.hidden

Methods

allowsNavigation

allowsNavigation(): boolean

Returns boolean

applyDesign

applyDesign(design: PageLinkDesign): void

Applies given design to the design on the control. If a design already exists, the prototype chain of the design will be preserved.

> Overrides Control.applyDesign

Parameters

NAME	TYPE	DESCRIPTION
design	PageLinkDesign	object containing design properties as keys

Returns void

dataContext

dataContext(): any

> Inherited from Control.dataContext

Returns any

getCount

getCount(): number | string

Returns number | string

getDesign

getDesign(): Design

Returns the design object of this control.

> Inherited from Control.getDesign

Returns Design

getNavigationHandler

getNavigationHandler(): NavigationArgs

Returns NavigationArgs

isEditable

isEditable(): boolean

Boolean indicating if the control is editable. Returns false when either the control or its parent is not editable. Returns true when both the control and its parent are editable. Returns true when either the control or its parent is editable and the other is undefined. Returns undefined if both the control's edit-ability and its parent's edit-ability is undefined.

> Inherited from Control.isEditable

Returns boolean

metadata

metadata(): PageLinkMetadata

Returns the metadata object of this control.

> Overrides Control.metadata

Returns PageLinkMetadata

parent

parent(): Control | Page

Returns the parent (control or page) of this control.

> Inherited from Control.parent

Returns Control | Page

root

root(): Page

Returns the root form instance (page) of this control.

> Inherited from Control.root

Returns Page

showCount

showCount(): boolean

Returns boolean

PageLinkDesign type

2/18/2021 • 3 minutes to read • Edit Online

Pagelink design object type.

Hierarchy

Design
└ PageLinkDesign

Index

Properties

Properties

alignItems

alignItems: string (optional)

This property is an alias for the CSS property "align-items". Please refer to this web page for documentation on the "align-items" property.

> Inherited from Design.alignItems

alignSelf

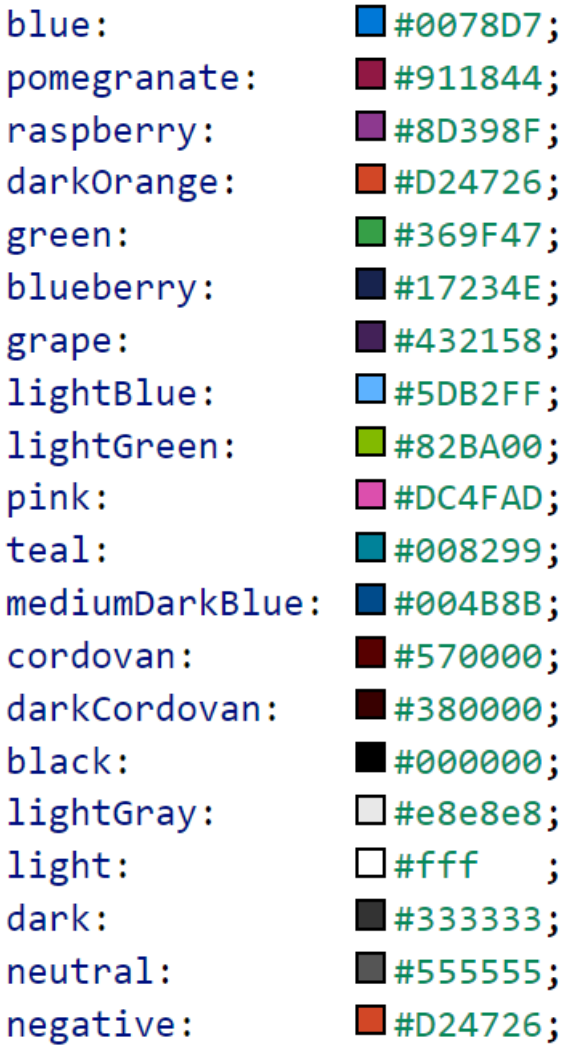
alignSelf: string (optional)

> Inherited from Design.alignSelf

background

background: string (optional)

Sets the background color. If "theme" is used, then the color will match the app's theme color.



bindings

bindings: any (optional)

> Inherited from Design.bindings

border

border: "none" | "solid" | "left" | "right" | "top" | "bottom" (optional)

The border behavior of a control. This property will not be inherited by the children.

> Inherited from Design.border

color

color: string (optional)

The foreground color of the container. This will modify the color of all headers, items, labels, and icons within the container.

Consider setting the background color at the same time as necessary when setting this attribute.
Note: if color is set to "theme", the theme color of the app will be used.
The following colors are available:

> Inherited from Design.color

excludeContext

excludeContext: boolean (optional)

flexFlow

flexFlow: string (optional)

Specifying this property makes the component a flex container component. This property is an alias for the CSS property "flex-flow". Please refer to this web page for documentation on the "flex-flow" property.

> Inherited from Design.flexFlow

flexSize

flexSize: string (optional)

One number or two numbers written as a string. For example, "(size to grow) [(size-to-shrink)]" to accommodate available space in the immediate flex container. This property is an alias for the CSS property "flex". Please refer to this web page for documentation on the "flex" property.

> Inherited from Design.flexSize

fontSize

fontSize: "medium" | "xx-small" | "x-small" | "small" | "large" | "x-large" | "xx-large" (optional)

The proportional text size

> Inherited from Design.fontSize

fontWeight

fontWeight: "normal" | "bold" (optional)

Normal or bold text.

> Inherited from Design.fontWeight

hideArrow

hideArrow: boolean (optional)

Allows an arrow (>) on a default styled navigation control to be hidden. By default, arrows are present in a navigation control.
This property can only be added through the design object.

icon

icon: string (optional)

Name of the icon that is displayed in the pagelink control. Here is a list of available icons.

justifyItems

justifyItems: "flex-start" | "flex-end" | "center" | "space-between" (optional)

This property is an alias for the CSS property "justify-content". Please refer to this web page for documentation on the "justify-content" property.

> Inherited from Design.justifyItems

label

label: string (optional)

> Inherited from Design.label

labelPosition

labelPosition: "stacked" | "hidden" | "inline" (optional)

Determines how a label is positioned, if at all. By default, labelPosition is set to stacked.

> Inherited from Design.labelPosition

name

name: string (optional)

> Inherited from Design.name

navigation

navigation: NavigationArgs (optional)

Navigation object of the pagelink.

padding

padding: "none" | "small" | "std" (optional)

Allows specifying the component's padding behavior. A component will inherit the padding behavior specified by its parent container components.

> Inherited from Design.padding

showCount

showCount: boolean (optional)

If true, shows a count of the records present in the list on the target page. This property is only suitable when the navigation target is a Page which contains on a List control.

style

style: string (optional)

Determines the visual style of the pagelink control. Options:

- "inline": takes up the full width its container, with the label in-line with the icon
- "button": takes up only as much width as needed by the label, with the label below the icon

type

type: ControlType (optional)

The type of the control as a string.

> Inherited from Design.type

PageLinkMetadata type

2/18/2021 • 2 minutes to read • Edit Online

Pagelink metadata type.

Hierarchy

ControlMetadata
└ PageLinkMetadata

Index

Properties

- BoundEntity
- BoundField
- Description
- Editable
- ExcludeContext
- ExtType
- HelpText
- Hidden
- Icon
- IconSize
- Id
- Label
- Name
- Navigation
- Order
- ShowCount
- Style
- Target
- Type
- UseDataContext

Events

- OnNavigate

Properties

BoundEntity

BoundEntity: string (optional)

The entity to which the control is bound.

Inherited from ControlMetadata.BoundEntity

BoundField

BoundField: string (optional)

Inherited from ControlMetadata.BoundField

Description

Description: string (optional)

Description of the control.

Inherited from ControlMetadata.Description

Editable

Editable: boolean (optional)

Boolean indicating if the control is editable. False when either the control or its parent is not editable. True when both the control and its parent are editable. True when either the control or its parent is editable and the other is undefined. Undefined if both the control's edit-ability and its parent's edit-ability is undefined.

Inherited from ControlMetadata.Editable

ExcludeContext

ExcludeContext: boolean (optional)

ExtType

ExtType: ControlType (optional)

The extended control type. For example, a control of type Input might have an extended type of Barcode.

Inherited from ControlMetadata.ExtType

HelpText

HelpText: string (optional)

The keyboard shortcut for a command. For example, "(Shift+F5)"

Inherited from ControlMetadata.HelpText

Hidden

Hidden: boolean (optional)

Boolean indicating if the control is hidden or not.

Inherited from ControlMetadata.Hidden

Icon

Icon: string (optional)

Name of the icon that is displayed in the page link control. Here is a list of available icons.

IconSize

IconSize: number (optional)

Determines the size of the icon that is displayed in the page link control.

Id

Id: string (optional)

Identification string for a control.

Inherited from ControlMetadata.Id

Label

Label: string (optional)

Label for a control. For example, a control representing a person's first name might have a label "First Name".

Inherited from ControlMetadata.Label

Name

Name: string (optional)

Name of a control.

Inherited from ControlMetadata.Name

Navigation

Navigation: NavigationArgs (optional)

Navigation object of the page link.

Order

Order: number (optional)

Number indicating the order in which a control will appear on a page.

Inherited from ControlMetadata.Order

ShowCount

ShowCount: boolean (optional)

If true, shows a count of the records present in the list on the target page. This property is only suitable when the navigation target is a Page which contains on a List control.

Style

Style: string (optional)

Determines the visual style of the page link control. Options:

- "inline": takes up the full width its container, with the label in-line with the icon
- "button": takes up only as much width as needed by the label, with the label below the icon

Target

Target: string (optional)

Name of the target action or page to navigate to when the page link is selected.

Type

Type: ControlType (optional)

String indicating the control type.

Inherited from ControlMetadata.Type

UseDataContext

UseDataContext: boolean (optional)

Events

OnNavigate

OnNavigate: function(navigation: NavigationArgs | string): any (optional)

An event that is triggered when the navigation is triggered.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

PageMetadata type

2/18/2021 • 2 minutes to read • Edit Online

Hierarchy

PageMetadata

Index

Properties

- Controls
- Design
- ID
- QuickSubmit
- SourcePageId
- SubmitButtonDesign
- Tasks
- Title

Events

- OnDataLoaded
- OnInit
- OnPreInit
- OnSubmit
- OnTaskSubmitted
- OnTaskSubmitting

Properties

Controls

Controls: ControlMetadata[] (optional)

Design

Design: Design (optional)

ID

ID: string (optional)

QuickSubmit

QuickSubmit: boolean (optional)

SourcePageId

SourcePageId: string (optional)

SubmitButtonDesign

SubmitButtonDesign: Design (optional)

Tasks

Tasks: PageMetadata[] (optional)

Title

Title: string (optional)

Events

OnDataLoaded

OnDataLoaded: function(sender: Page, dataWrapper: any): void (optional)

OnInit

OnInit: function(sender: Page): void (optional)

OnPreInit

OnPreInit: function(sender: Page): void (optional)

OnSubmit

OnSubmit: function(dataValues: any, args: any): void (optional)

OnTaskSubmitted

OnTaskSubmitted: function(taskHandle: any, taskOptions: any): any (optional)

OnTaskSubmitting

OnTaskSubmitting: function(taskOptions: any): any (optional)

PageOptions type

2/18/2021 • 2 minutes to read • Edit Online

Hierarchy

PageOptions

Index

Properties

Properties

appId

appId: string (optional)

design

design: Design (optional)

excludeContext

excludeContext: boolean (optional)

filter

filter: DataFilter (optional)

filterLocalOnly

filterLocalOnly: boolean (optional)

pageContext

pageContext: string (optional)

pageId

pageId: string (optional)

readOptions

readOptions: IReadOptions (optional)

PageSubmitArgs type

2/18/2021 • 2 minutes to read • Edit Online

Args supplied to the OnSubmit event of the page.

Hierarchy

PageSubmitArgs

Index

Properties

- dataValues
- sender

Methods

- addMessage
- cancel
- getMessages
- isCancelled
- wait

Properties

dataValues

dataValues: any

Get the payload of the submit action.

sender

sender: Page

Get the sender page instance of the submit action.

Methods

addMessage

addMessage(message: string, type: any): any

Add a validation/error message to be displayed.

Parameters

NAME	TYPE	DESCRIPTION
message	string	
type	any	

Returns any

cancel

cancel(): any

Prevent the action from submitting.

Returns any

getMessages

getMessages(): string [ ]

Get all previously added messages

Returns string [ ]

isCancelled

isCancelled(): boolean

Check if the submit action is cancelled.

Returns boolean

wait

wait(promise: Promise <any>): any

Wait on a given promise before continuing with the submission. All promises attached via wait must resolve before the submit action is performed.

Parameters

NAME	TYPE	DESCRIPTION
promise	Promise <any>	

Returns any

PageTarget type

2/18/2021 • 2 minutes to read • Edit Online

Hierarchy

PageTarget
└ NavigationArgs

Index

Properties

- params
- to

Properties

params

params: PageOptions (optional)

to

to: string (optional)

Part type

2/18/2021 • 2 minutes to read • Edit Online

Part control type. A part is a container control that contains only a page, allowing for a page to be embedded within a page.

Hierarchy

ContainerControl
└ Part

Index

Properties

- container
- generic
- getDataSource
- hidden

Methods

- applyDesign
- dataContext
- getControl
- getControlById
- getDesign
- getEntityRef
- getPartPage
- hasTarget
- isEditable
- metadata
- parent
- root

Properties

container

container: boolean

True if the control is a container.

> Inherited from ContainerControl.container
>
> Overrides Control.container

generic

generic: boolean (optional)

> Inherited from Control.generic

getDataSource

getDataSource: function(): any

> Inherited from Control.getDataSource

hidden

hidden: boolean

True if the control is hidden.

> Inherited from Control.hidden

Methods

applyDesign

applyDesign(IDesign: PartDesign): void

Applies given design to the design on the control. If a design already exists, the prototype chain of the design will be preserved.

> Overrides Control.applyDesign

Parameters

NAME	TYPE	DESCRIPTION
IDesign	PartDesign	object containing design properties as keys

Returns void

dataContext

dataContext(): any

> Inherited from Control.dataContext

Returns any

getControl

getControl(controlName: string): Control

Given the name of a control, returns the control instance.

> Inherited from ContainerControl.getControl

Parameters

NAME	TYPE	DESCRIPTION
controlName	string	control name

Returns Control

getControlById

getControlById(id: string): Control

Given the ID of a control, returns the control instance.

> Inherited from ContainerControl.getControlById

Parameters

NAME	TYPE	DESCRIPTION
id	string	control ID

Returns Control

getDesign

getDesign(): Design

Returns the design object of this control.

> Inherited from Control.getDesign

Returns Design

getEntityRef

getEntityRef(): string

Gets value of entityRef binding to control.

Returns string

getPartPage

getPartPage(): Page

Gets the page of the part.

Returns Page

hasTarget

hasTarget(): boolean

Returns true if the part has a target page.

Returns boolean

isEditable

isEditable(): boolean

Boolean indicating if the control is editable. Returns false when either the control or its parent is not editable. Returns true when both the control and its parent are editable. Returns true when either the control or its parent is editable and the other is undefined. Returns undefined if both the control's edit-ability and its parent's edit-ability is undefined.

> Inherited from Control.isEditable

Returns boolean

metadata

metadata(): PartMetadata

Returns the metadata object of this control.

> Overrides ContainerControl.metadata

Returns PartMetadata

parent

parent(): Control | Page

Returns the parent (control or page) of this control.

> Inherited from Control.parent

Returns Control | Page

root

root(): Page

Returns the root form instance (page) of this control.

> Inherited from Control.root

Returns Page

PartDesign type

2/18/2021 • 3 minutes to read • Edit Online

Part design object type.

Hierarchy

ContainerControlDesign
└ PartDesign

Index

Properties

Properties

alignItems

alignItems: string (optional)

This property is an alias for the CSS property "align-items". Please refer to this web page for documentation on the "align-items" property.

> Inherited from Design.alignItems

alignSelf

alignSelf: string (optional)

Inherited from Design.alignSelf

allowScroll

allowScroll: string (optional)

True if the container will allow scrolling when its items do not fit into the container's available space. If a container has an item which may scroll, then set this property to false to prevent nested scrolling areas.

Inherited from ContainerControlDesign.allowScroll

background

background: string (optional)

The background color of the container. Consider modifying the color attribute in the same container so that fonts overlaying the background color will appear appropriately. Note: if background is set to "theme", the theme color of the app will be used. The following colors are available:

```
blue:          #0078D7;
pomegranate:   #911844;
raspberry:     #8D398F;
darkOrange:    #D24726;
green:         #369F47;
blueberry:     #17234E;
grape:         #432158;
lightBlue:     #5DB2FF;
lightGreen:    #82BA00;
pink:          #DC4FAD;
teal:          #008299;
mediumDarkBlue: #004B8B;
cordovan:      #570000;
darkCordovan:  #380000;
black:         #000000;
lightGray:     #e8e8e8;
light:         #fff   ;
dark:          #333333;
```

Inherited from ContainerControlDesign.background

bindings

bindings: any (optional)

Inherited from Design.bindings

border

border: "none" | "solid" | "left" | "right" | "top" | "bottom" (optional)

The border behavior of a control. This property will not be inherited by the children.

> Inherited from Design.border

color

color: string (optional)

The foreground color of the container. This will modify the color of all headers, items, labels, and icons within the container.
Consider setting the background color at the same time as necessary when setting this attribute.
Note: if color is set to "theme", the theme color of the app will be used.
The following colors are available:

> Inherited from Design.color

design

design: PartDesign (optional)

Design for the target page.

flexFlow

flexFlow: string (optional)

Specifying this property makes the component a flex container component. This property is an alias for the CSS property "flex-flow". Please refer to this web page for documentation on the "flex-flow" property.

> Inherited from Design.flexFlow

flexSize

flexSize: string (optional)

One number or two numbers written as a string. For example, "(size to grow) [(size-to-shrink)]" to accommodate available space in the immediate flex container. This property is an alias for the CSS property "flex". Please refer to this web page for documentation on the "flex" property.

> Inherited from Design.flexSize

fontSize

fontSize: "medium" | "xx-small" | "x-small" | "small" | "large" | "x-large" | "xx-large" (optional)

The proportional text size

> Inherited from Design.fontSize

fontWeight

fontWeight: "normal" | "bold" (optional)

Normal or bold text.

> Inherited from Design.fontWeight

itemBorder

itemBorder: "solid" | "none" (optional)

If true, a border will appear around each row in the list. This property is equivalent to applying the border property individually to all items in the container.

> Inherited from ContainerControlDesign.itemBorder

items

items: string | Design [ ] (optional)

An array containing the components to place inside of the container.

> Inherited from ContainerControlDesign.items

justifyItems

justifyItems: "flex-start" | "flex-end" | "center" | "space-between" (optional)

This property is an alias for the CSS property "justify-content". Please refer to this web page for documentation on the "justify-content" property.

> Inherited from Design.justifyItems

label

label: string (optional)

> Inherited from Design.label

labelPosition

labelPosition: "stacked" | "hidden" | "inline" (optional)

Determines how a label is positioned, if at all. By default, labelPosition is set to stacked.

> Inherited from Design.labelPosition

name

name: string (optional)

> Inherited from Design.name

padding

padding: "none" | "small" | "std" (optional)

Allows specifying the component's padding behavior. A component will inherit the padding behavior specified by its parent container components.

> Inherited from Design.padding

target

target: PageTarget (optional)

Target page of the part.

type

type: ControlType (optional)

The type of the control as a string.

> Inherited from Design.type

PartMetadata type

2/18/2021 • 2 minutes to read • Edit Online

Part metadata type.

Hierarchy

ContainerControlMetadata
└ PartMetadata

Index

Properties

- BoundEntity
- BoundField
- Description
- Design
- Editable
- ExtType
- HelpText
- Hidden
- Id
- Label
- Name
- Order
- Target
- Type

Properties

BoundEntity

BoundEntity: string (optional)

The entity to which the control is bound.

> Inherited from ControlMetadata.BoundEntity

BoundField

BoundField: string (optional)

> Inherited from ControlMetadata.BoundField

Description

Description: string (optional)

Description of the control.

> Inherited from ControlMetadata.Description

Design

Design: PartDesign (optional)

Design for the target page.

Editable

Editable: boolean (optional)

Boolean indicating if the control is editable. False when either the control or its parent is not editable. True when both the control and its parent are editable. True when either the control or its parent is editable and the other is undefined. Undefined if both the control's edit-ability and its parent's edit-ability is undefined.

> Inherited from ControlMetadata.Editable

ExtType

ExtType: ControlType (optional)

The extended control type. For example, a control of type Input might have an extended type of Barcode.

> Inherited from ControlMetadata.ExtType

HelpText

HelpText: string (optional)

The keyboard shortcut for a command. For example, "(Shift+F5)"

> Inherited from ControlMetadata.HelpText

Hidden

Hidden: boolean (optional)

Boolean indicating if the control is hidden or not.

> Inherited from ControlMetadata.Hidden

Id

Id: string (optional)

Identification string for a control.

> Inherited from ControlMetadata.Id

Label

Label: string (optional)

Label for a control. For example, a control representing a person's first name might have a label "First Name".

> Inherited from ControlMetadata.Label

Name

Name: string (optional)

Name of a control.

> Inherited from ControlMetadata.Name

Order

Order: number (optional)

Number indicating the order in which a control will appear on a page.

> Inherited from ControlMetadata.Order

Target

Target: PageTarget (optional)

Target page of the part.

Type

Type: ControlType (optional)

String indicating the control type.

> Inherited from ControlMetadata.Type

Row type

2/18/2021 • 2 minutes to read • Edit Online

Row controls are what make up a list. A list contains any number of row controls.

Hierarchy

Row

Index

Properties

- fieldList
- headerField
- hidden
- imageFields
- isSelected
- item
- template

Methods

- getControl
- getControlById
- getControlValueById
- getRowHeader
- getRowId
- hasImageField
- isEntityCreatedNew
- isEntityDeleted
- isEntityModified
- isEntitySyncPending
- select

Properties

fieldList

fieldList: Control [ ]

headerField

headerField: Control

hidden

hidden: boolean

If true then the row will be hidden.

imageFields

imageFields: Control [ ]

isSelected

isSelected: boolean

item

item: any

A container of rendered data.

template

template: Group

Group control that represents the template for a row.

Methods

getControl

getControl(controlName: string): Control

Parameters

NAME	TYPE	DESCRIPTION
controlName	string	

Returns Control

getControlById

getControlById(id: string): Control

Parameters

NAME	TYPE	DESCRIPTION
id	string	id can be valueKey or the displayKey of the list control metadata

Returns Control

getControlValueById

getControlValueById(id: string): string

Parameters

NAME	TYPE	DESCRIPTION
id	string	id can be valueKey or the displayKey of the list control metadata

Returns string

getRowHeader

getRowHeader(): Control

Returns Control

getRowId

getRowId(): string

Returns string

hasImageField

hasImageField(): boolean

Returns true if the row has an image field.

Returns boolean

isEntityCreatedNew

isEntityCreatedNew(): boolean

Returns boolean

isEntityDeleted

isEntityDeleted(): boolean

Returns boolean

isEntityModified

isEntityModified(): boolean

Returns boolean

isEntitySyncPending

isEntitySyncPending(): boolean

Returns boolean

select

select(): any

Returns any

Value type

2/18/2021 • 2 minutes to read • Edit Online

Value control type. This is the base class for single value controls.

Hierarchy

InputControl
└ Value
  └ FileUploader
  └ HyperLink
  └ GenericValue

Index

Properties

- container
- generic
- getDataSource
- hidden

Methods

- applyDesign
- dataContext
- getDesign
- getValue
- isEditable
- metadata
- parent
- root
- setValue

Events

- onDataChanged

Properties

container

container: boolean (optional)

True if the control is a container.

> Inherited from Control.container

generic

generic: boolean (optional)

> Inherited from Control.generic

getDataSource

getDataSource: function(): any

> Inherited from Control.getDataSource

hidden

hidden: boolean

True if the control is hidden.

> Inherited from Control.hidden

Methods

applyDesign

applyDesign(design: Design): void

Applies given design to the design on the control. If a design already exists, the prototype chain of the design will be preserved.

> Inherited from Control.applyDesign

Parameters

NAME	TYPE	DESCRIPTION
design	Design	object containing design properties as keys

Returns void

dataContext

dataContext(): any

> Inherited from Control.dataContext

Returns any

getDesign

getDesign(): Design

Returns the design object of this control.

> Inherited from Control.getDesign

Returns Design

getValue

getValue(): string

Returns the value of the control.

Returns string

isEditable

isEditable(): boolean

Boolean indicating if the control is editable. Returns false when either the control or its parent is not editable. Returns true when both the control and its parent are editable. Returns true when either the control or its parent is editable and the other is undefined. Returns undefined if both the control's edit-ability and its parent's edit-ability is undefined.

> Inherited from Control.isEditable

Returns boolean

metadata

metadata(): ValueMetadata

Returns the metadata object of this control.

> Overrides InputControl.metadata

Returns ValueMetadata

parent

parent(): Control | Page

Returns the parent (control or page) of this control.

> Inherited from Control.parent

Returns Control | Page

root

root(): Page

Returns the root form instance (page) of this control.

> Inherited from Control.root

Returns Page

setValue

setValue(value: string): void

Sets the value of the control.

Parameters

NAME	TYPE	DESCRIPTION
value	string	

Returns void

Events

onDataChanged

onDataChanged: EventHook <null>

An event that is triggered when the input control's data changes.

> Inherited from InputControl.onDataChanged

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

ValueDesign type

2/18/2021 • 2 minutes to read • Edit Online

Value design object type.

Hierarchy

InputControlDesign
└ ValueDesign
└ FileUploaderDesign
└ HyperLinkDesign

Index

Properties

Properties

alignItems

alignItems: string (optional)

This property is an alias for the CSS property "align-items". Please refer to this web page for documentation on the "align-items" property.

> Inherited from Design.alignItems

alignSelf

alignSelf: string (optional)

> Inherited from Design.alignSelf

bindings

bindings: any (optional)

> Inherited from Design.bindings

border

border: "none" | "solid" | "left" | "right" | "top" | "bottom" (optional)

The border behavior of a control. This property will not be inherited by the children.

> Inherited from Design.border

color

color: string (optional)

The foreground color of the container. This will modify the color of all headers, items, labels, and icons within the container.
Consider setting the background color at the same time as necessary when setting this attribute.
Note: if color is set to "theme", the theme color of the app will be used.
The following colors are available:

```
blue:           #0078D7;
pomegranate:    #911844;
raspberry:      #8D398F;
darkOrange:     #D24726;
green:          #369F47;
blueberry:      #17234E;
grape:          #432158;
lightBlue:      #5DB2FF;
lightGreen:     #82BA00;
pink:           #DC4FAD;
teal:           #008299;
mediumDarkBlue: #004B8B;
cordovan:       #570000;
darkCordovan:   #380000;
black:          #000000;
lightGray:      #e8e8e8;
light:          #fff   ;
dark:           #333333;
```

> Inherited from Design.color

flexFlow

flexFlow: string (optional)

Specifying this property makes the component a flex container component. This property is an alias for the CSS property "flex-flow". Please refer to this web page for documentation on the "flex-flow" property.

> Inherited from Design.flexFlow

flexSize

flexSize: string (optional)

One number or two numbers written as a string. For example, "(size to grow) [(size-to-shrink)]" to accommodate available space in the immediate flex container. This property is an alias for the CSS property "flex". Please refer to this web page for documentation on the "flex" property.

> Inherited from Design.flexSize

fontSize

fontSize: "medium" | "xx-small" | "x-small" | "small" | "large" | "x-large" | "xx-large" (optional)

The proportional text size

> Inherited from Design.fontSize

fontWeight

fontWeight: "normal" | "bold" (optional)

Normal or bold text.

> Inherited from Design.fontWeight

justifyItems

justifyItems: "flex-start" | "flex-end" | "center" | "space-between" (optional)

This property is an alias for the CSS property "justify-content". Please refer to this web page for documentation on the "justify-content" property.

> Inherited from Design.justifyItems

label

label: string (optional)

> Inherited from Design.label

labelPosition

labelPosition: "stacked" | "hidden" | "inline" (optional)

Determines how a label is positioned, if at all. By default, labelPosition is set to stacked.

> Inherited from Design.labelPosition

name

name: string (optional)

> Inherited from Design.name

padding

padding: "none" | "small" | "std" (optional)

Allows specifying the component's padding behavior. A component will inherit the padding behavior specified by its parent container components.

> Inherited from Design.padding

type

type: ControlType (optional)

The type of the control as a string.

> Inherited from Design.type

ValueMetadata type

2/18/2021 • 2 minutes to read • Edit Online

Value metadata type.

Hierarchy

InputControlMetadata
└ ValueMetadata
└ FileUploaderMetadata
└ HyperLinkMetadata

Index

Properties

- BoundEntity
- BoundField
- Description
- Editable
- ExtType
- HelpText
- Hidden
- Id
- Label
- Mandatory
- Name
- NumSequence
- Order
- Type

Properties

BoundEntity

BoundEntity: string (optional)

The entity to which the control is bound.

> Inherited from ControlMetadata.BoundEntity

BoundField

BoundField: string (optional)

> Inherited from ControlMetadata.BoundField

Description

Description: string (optional)

Description of the control.

> Inherited from ControlMetadata.Description

Editable

Editable: boolean (optional)

Boolean indicating if the control is editable. False when either the control or its parent is not editable. True when both the control and its parent are editable. True when either the control or its parent is editable and the other is undefined. Undefined if both the control's edit-ability and its parent's edit-ability is undefined.

> Inherited from ControlMetadata.Editable

ExtType

ExtType: ControlType (optional)

The extended control type. For example, a control of type Input might have an extended type of Barcode.

> Inherited from ControlMetadata.ExtType

HelpText

HelpText: string (optional)

The keyboard shortcut for a command. For example, "(Shift+F5)"

> Inherited from ControlMetadata.HelpText

Hidden

Hidden: boolean (optional)

Boolean indicating if the control is hidden or not.

> Inherited from ControlMetadata.Hidden

Id

Id: string (optional)

Identification string for a control.

> Inherited from ControlMetadata.Id

Label

Label: string (optional)

Label for a control. For example, a control representing a person's first name might have a label "First Name".

> Inherited from ControlMetadata.Label

Mandatory

Mandatory: boolean (optional)

If set to true then input for the control is required for the task to be completed. Mandatory controls will have a red outline.

> Inherited from InputControlMetadata.Mandatory

Name

Name: string (optional)

Name of a control.

> Inherited from ControlMetadata.Name

NumSequence

NumSequence: NumberSequenceConfig (optional)

Used for auto detecting and changing visibility of the number sequence controls in the task or page, based on AX number sequence configuration, through extended business logic. Example:

```
// hide number sequence reference page from users
metadataService.hideNavigation('numSeqReferencePage');

// parameters to be passed to 'numSequence' flag in configureControl
var configParam = {
     referencePageName: 'numSeqReferencePage',
     dataType: 'HcmPersonnelNumberId'
};

// setup 'PersonnelNumber' control as number sequence in the task 'add-worker'
metadataService.configureControl('add-worker', 'PersonnelNumber', { numSequence: configParam });
```

> Inherited from InputControlMetadata.NumSequence

Order

Order: number (optional)

Number indicating the order in which a control will appear on a page.

> Inherited from ControlMetadata.Order

Type

Type: ControlType (optional)

String indicating the control type.

> Inherited from ControlMetadata.Type

Process automation framework development

2/18/2021 • 2 minutes to read • Edit Online

Process automation enables simple scheduling of processes that will be run by the batch server. The process automation framework is a set of APIs that lets you implement process automation.

You should use only the public APIs to implement process automation, and you should follow these guidelines:

- Don't select from, insert into, or directly reference the process automation tables.
- Don't extend the framework or integrate your code with the classes.
- Don't subscribe to table events such as insert, update, and delete. Finance and Operations apps skip most of those events.
- If functionality that you require is missing, submit feature requests.

Microsoft plans to add features in the future. If you integrate too deeply with the process automation framework, your integration might break when those features are added.

Some of the examples for the process automation framework aren't representative of release-quality code. As always, the expectation is that processes that are built by using the framework will follow all best practices and quality standards.

For more information about process automation, see Process automation.

Definitions

TERM	DEFINITION
Poller	The poller is a system-critical batch process that runs every minute and invokes various subsystems of the process automation framework. It consults the schedule to determine which processes are ready to run, and then it invokes the runtime side of the framework to ensure that processes are run.
Scheduled process	A scheduled process is a process that is scheduled in the user interface (UI) by a user. Occurrences for these processes can be seen in a calendar view.
Background process	A background process is also known as a *polled process*. It's a process that runs frequently, without requiring user input, and performs some background processing. Subledger transfer to the general ledger is an example.
Type	In this topic and related topics, the term *type* refers to **ProcessScheduleType**, as discussed in Type registration.
Series	Every process that has a registered type must have a series. Series for scheduled processes are created in the UI by users. Series for background processes are created through series registration. For more information, see Series registration.
Date and time	All framework dates are stored in Coordinated Universal Time (UTC) but shown in the user's preferred time zone.

Tasks

Implementation of a process automation solution consists of a set of tasks, some of which are required and some of which are optional.

Most of the UI customizations aren't supported for background processes. The **Series** list page and logging of results and messages are supported.

TASK	REQUIRED FOR A SCHEDULED PROCESS	REQUIRED FOR A BACKGROUND PROCESS
Type registration	Yes	Yes
Series registration	Not supported	Yes
Process parameters	No	Not supported
User-configurable queries	No	Not supported
Run processes	Yes	Yes
Log results and messages	Yes	Yes
Customize the user interface	No	See Customize the user interface.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Type registration

2/18/2021 • 6 minutes to read • Edit Online

To implement a process by using the process automation framework, you must first understand the concept of a *type* in the framework. A type is a unique process that is integrated with the batch framework and that uses the **SysOperations** framework, specifically the **SysOperationServiceController** class. The types are stored in the **ProcessScheduleType** table.

Here are a few examples of different types that are registered with the process automation framework:

- Vendor Payment Proposal (**VendPaymProposalAutomationTypeRegistrationProvider** class)
- Vendor Invoice Posting (**VendInvoicePostProcessScheduleTypeRegistration** class)
- Subledger transfer to general ledger (**SubledgerJournalVoucherTransferServiceRegistration** class)

If an existing process uses **RunBaseBatch**, consider wrapping it with **SysOperationServiceController**. The process automation framework doesn't support **RunBaseBatch**.

To register your type with the process automation framework, you must implement the **ProcessScheduleITypeRegistration** interface. This interface has a single method that returns an instance of **ProcessScheduleTypeRegistrationItem**.

If your process uses a feature flag, you must disable and enable the type as the feature is disabled and enabled, respectively.

- If you disable the feature flag for a type, the type doesn't appear in the user interface (UI). The scheduler won't schedule any occurrences or background processes of that type to run, and the runtime side of the process automation framework won't create any batch jobs for that type.
- If you enable the feature flag for a type, any occurrences or background processes that are scheduled to run in the past will be run immediately. Usually, this behavior is what you want. However, if it isn't what you want, consider disabling any series that is related to the type before you disable the feature flag.

Feature management has events that you can subscribe to. The method that you use to enable and disable types is **ProcessScheduleTypeRegistration.enableOrDisableType**.

Every time that a database synchronization runs, types and series are updated from their definitions in code. The only background settings that aren't updated are those that can be edited by the system admin. The process automation framework does this update by hooking **SysSetup**. The system admin can manually trigger this update through the settings at **System administration > Setup > Initialize background**.

The following example shows a process for a scheduled type. Note the following points:

- A background process doesn't have to set the **Parameter** tab list, because background processes don't support parameters.
- The type name isn't shown in the UI. The name should be a developer-created string such as **VendorInvoiceBatchPosting**. It's used internally as a key to reference your type for various purposes. It **cannot** be a label.
- The type name is used heavily with the **SysPlugIn** pattern. Most of the interfaces that are implemented for the process automation framework follow the **SysPlugIn** pattern and require that the type name be supplied by the **ExportMetadataAttribute**. In most cases in this pattern, the framework invokes the implementation of an interface only for the type that is being operated on, not for other types. Code examples in this topic and related topics follow this pattern.

```
using System.ComponentModel.Composition;

// The VendPaymProposalAutomationTypeRegistrationProvider class handles type registration for Vendor payment
proposal automations.
[Export(identifierStr(Dynamics.AX.Application.ProcessScheduleITypeRegistration))]
public final class VendPaymProposalAutomationTypeRegistrationProvider
implements ProcessScheduleITypeRegistration
{
    private const LabelId Caption = literalStr('@CashManagement:VendPaymProposalAutomationTypeName');
    private const LabelId HelpText =
literalStr('@CashManagement:VendPaymProposalAutomationSeriesWizardHelpText');
    private const MenuItemName SeriesFormMenuItemName =
menuItemDisplayStr(VendPaymProposalAutomationCriteriaSeries);

    [Wrappable(false)]
    public ProcessScheduleTypeRegistrationItem getScheduleTypeRegistrationItem()
    {
        ProcessScheduleTypeRegistrationItem item =
        ProcessScheduleTypeRegistrationItem::construct();
        item.parmName(VendPaymProposalAutomationConstants::RegisteredTypeName);
        item.parmLabelId(Caption);
        item.parmScheduleType(ProcessScheduleProcessType::Scheduled);
        item.parmCompanyScope(ProcessScheduleTypeCompanyScope::SingleCompany);
        item.parmProcessAutomationTaskClassName(classStr(VendPaymProposalAutomationTask));
        item.parmParameterTabItemList(this.constructParameterTabItemList());
        item.parmIsEnabled(VendPaymProposalAutomationFeature::isEnabled());
        return item;
        }

    private List constructParameterTabItemList()
    {
        List criteriaTabItemList = new List(Types::Class);
        ProcessScheduleTypeRegistrationParameterTabItem criteriaTabItem =
        ProcessScheduleTypeRegistrationParameterTabItem::newFromMenuItem(SeriesFormMenuItemName);

        criteriaTabItem.parmCaption(Caption);
        criteriaTabItem.parmHelpText(HelpText);
        criteriaTabItemList.addEnd(criteriaTabItem);
        return criteriaTabItemList;
        }
}
```

The following example shows feature management for a process automation framework type.

```
using System.ComponentModel.Composition;
using Microsoft.Dynamics.ApplicationPlatform.FeatureExposure;
using Microsoft.Dynamics.ApplicationPlatform.FeatureExposure.Implementation;

// The VendPaymProposalAutomationFeature class defines the Vendor Payment Proposal Automation feature.
[Export(identifierStr(Microsoft.Dynamics.ApplicationPlatform.FeatureExposure.IFeatureMetadata))]
internal final class VendPaymProposalAutomationFeature implements IFeatureMetadata,
IFeatureMetadataEnablementNotifiable
{
    private static VendPaymProposalAutomationFeature instance;
    private void new()
    {
    }

    private static void typeNew()
    {
        instance = new VendPaymProposalAutomationFeature();
    }

    [Hookable(false)]
    public static VendPaymProposalAutomationFeature instance()
    {
```

```
        return VendPaymProposalAutomationFeature::instance;
    }

    [Hookable(false)]
    public FeatureLabelId label()
    {
        return literalStr("@CashManagement:VendPaymProposalAutomationFeatureName");
    }

    [Hookable(false)]
    public int module()
    {
        return FeatureModuleV0::AccountsPayable;
    }

    [Hookable(false)]
    public FeatureLabelId summary()
    {
        return literalStr("@CashManagement:VendPaymProposalAutomationFeatureSummary");
    }

    [Hookable(false)]
    public WebSiteURL learnMoreUrl()
    {
        return "<your URL>";
    }

    [Hookable(false)]
    public boolean isEnabledByDefault()
    {
        return false;
    }

    [Hookable(false)]
    public boolean canDisable()
    {
        return true;
    }

    [Hookable(false)]
    public void onEnabled()
    {
        this.enableOrDisableRegisteredType(NoYes::Yes);
    }

    [Hookable(false)]
    public void onDisabled()
    {
        this.enableOrDisableRegisteredType(NoYes::No);
    }

    private void enableOrDisableRegisteredType(NoYes _isEnabled)
    {

ProcessScheduleTypeRegistration::enableOrDisableType(VendPaymProposalAutomationConstants::RegisteredTypeName
,
    _isEnabled);
    }

    internal static boolean isEnabled()
    {
        return
Dynamics.AX.Application.FeatureStateProvider::isFeatureEnabled(VendPaymProposalAutomationFeature::instance()
);
    }
}
```

ProcessScheduleTypeRegistrationItem class

The **ProcessScheduleTypeRegistrationItem** class is used as a part of type registration and contains information that is specific to your type.

METHOD	DESCRIPTION
`public ProcessScheduleTypeName parmName(ProcessScheduleTypeName _name = name)`	The type name that is passed to the **item.parmName** method isn't shown to users. This value is a developer-defined string that is used when various events are invoked. It should be assigned as a constant, not as a label. **Never use a label as a type name.** Use names such as **VendPaymentProposal**.
`public ProcessScheduleProcessType parmScheduleType(ProcessScheduleProcessType _scheduleType = scheduleType)`	This method determines whether the process is scheduled or polled.
`public ProcessScheduleTypeCompanyScope parmCompanyScope(ProcessScheduleTypeCompanyScope _companyScope = companyScope)`	This method determines whether the process is a single-company process or a global process. A single-company process sets the company context in a batch, depending on the company that the user is in when a series is created. If the scope is global, the company context is ignored, and all jobs are in **dat**.
`public LabelId parmLabelId(LabelId _labelId = labelId)`	The label that this method returns is shown to users and represents the display name for your type. An example is **Vendor payment proposal**.
`public className parmProcessAutomationTaskClassName(ClassName _processAutomationTaskClassName = processAutomationTaskClassName)`	The class name that this method returns is the class name of the class that will implement the **ProcessAutomationTask** interface.
`public NoYes parmIsEnabled(NoYes _isEnabled = isEnabled)`	This method determines whether the type that you're registering is enabled by default. If your type is feature-managed, a default value is taken from the state of the feature. Be sure to implement the enabled and disabled feature management events. Enable and disable your type in the process automation framework in the appropriate way, by using **ProcessScheduleTypeRegistration.enableOrDisableType** method. Example code is shown earlier in this topic.
`public List parmParameterTabItemList(List _parameterTabList = parameterTabList)`	A process can have many parameter pages in the UI. These parameter pages contain parameters that are specific to the process. They are surfaced as form parts in the **Create series** wizard and edit occurrence dialog box. For each parameter page, an instance of the **ProcessSchedulelTypeRegistrationParameterTabItem** class must be constructed and returned in the list. If the process doesn't require parameter pages, return **null**. For more information, see Process parameters.
`public static ProcessScheduleTypeRegistrationItem construct()`	This method constructs an instance of the **ProcessScheduleTypeRegistrationItem** class.

ProcessSchedulelTypeRegistrationParameterTabItem class

The **ProcessSchedulelTypeRegistrationParameterTabItem** class represents information that is specific to a

single parameter page.

METHOD	DESCRIPTION
`public MenuItemName parmMenuItemName(MenuItemName _menuItemName = menuItemName)`	The **Create series** wizard supports parameter pages by using embedded form parts. This value is the menu item that goes to the form part that is created by the team that owns the process.
`public LabelId parmCaption(LabelId _caption = caption)`	The caption on the parameter page.
`public LabelId parmHelpText(LabelId _helpText = helpText)`	The Help text for the parameter page.
`public static ProcessScheduleTypeRegistrationParameterTabItem newFromMenuItem(MenuItemName _menuItemName)`	Constructor that initializes the instance with the specified menu item name.

Series registration

2/18/2021 • 6 minutes to read • Edit Online

Every process must have a *series*. The concept of a series in process automation resembles the concept of a meeting series in Microsoft Outlook. However, a series in process automation is a series of scheduled runs of a process. For most scheduled process types, users create the series in the user interface (UI), and series registration never has to be implemented. However, if the process that is being implemented is a schedule series, you can skip this task.

Background processes typically create a series via code, by using series registration, because background processes tend to be "under the hood" processes that don't allow for user interaction. To create a series via code, you implement the **ProcessScheduleISeriesRegistration** interface. This interface contains a single method that returns an instance of **ProcessScheduleSeriesRegistrationItem**.

The process automation framework lets system admins change the default polling interval and unit for background processes.

Here is an example of a test series that is used to test the process automation framework.

```
using System.ComponentModel.Composition;
// Implements the ProcessScheduleISeriesRegistration to register the vendor invoice batch posting task
'Series' with the Process Automation.
[Export(identifierStr(Dynamics.AX.Application.ProcessScheduleISeriesRegistration))]
[ExportMetadata(classStr(ProcessScheduleISeriesRegistration),
classStr(VendInvoicePostProcessScheduleSeriesRegistration))]
internal final class VendInvoicePostProcessScheduleSeriesRegistration implements
ProcessScheduleISeriesRegistration
{

    [Hookable(false)]
    public ProcessScheduleSeriesRegistrationItem
    getProcessScheduleSeriesRegistrationItem()
    {
        ProcessScheduleSeriesRegistrationItem
        processScheduleSeriesRegistrationItem =
        ProcessScheduleSeriesRegistrationItem::construct();

processScheduleSeriesRegistrationItem.parmDescription("@AccountsPayable:VendInvoicePostTaskFeatureSummary");
        processScheduleSeriesRegistrationItem.parmOwnerId(curUserId());

processScheduleSeriesRegistrationItem.parmProcessScheduleSeriesPatternList(this.getSeriesPatternList());

processScheduleSeriesRegistrationItem.parmSeriesName("@AccountsPayable:VendInvoicePostTaskFeatureLabel");

processScheduleSeriesRegistrationItem.parmTypeName(VendInvoicePostTaskConstants::VendorInvoiceBatchPosting);
        return processScheduleSeriesRegistrationItem;
        }

    private List getSeriesPatternList()
    {
        ProcessScheduleSeriesPatternItem processScheduleSeriesPatternItem =
        ProcessScheduleSeriesPatternItem::construct();
        processScheduleSeriesPatternItem.parmUnit(ProcessScheduleUnit::Minute);
        processScheduleSeriesPatternItem.parmPollingInterval(VendParameters::pollingIntervalMinutes());
        List list = new List(Types::Class);
        list.addEnd(processScheduleSeriesPatternItem);
        return list;
    }
}
```

ProcessScheduleSeriesRegistrationItem class

METHOD	DESCRIPTION
`public ProcessScheduleTypeName parmTypeName(ProcessScheduleTypeName _typeName = typeName)`	The name of the type.
`public ProcessScheduleSeriesName parmSeriesName(ProcessScheduleSeriesName _SeriesName = seriesName)`	The name of the series. Be descriptive, so that the purpose of the series is clear from the name.
`public Description parmDescription(Description _description = description)`	The description of the series.
`public UserGroupId parmOwnerId(UserGroupId _ownerId = ownerId)`	The user ID of the owner of the series.

METHOD	DESCRIPTION
`public List parmProcessScheduleSeriesPatternList(List _seriesPatternList = seriesPatternList)`	The list of patterns for the series. Currently, only one pattern per series is supported. However, this limitation might change in the future. Insert an instance of the **ProcessScheduleSeriesPatternItem** class into the list.

ProcessScheduleSeriesPatternItem class

When the pattern is configured, applicable fields are determined based on the unit. Not all methods that are defined in the following table work for all units. The methods that apply to units are defined in the tables later in this section. Other combinations will be ignored. For polled processes, only the unit and the polling interval are used. Other fields are ignored.

METHOD	DESCRIPTION
`public ProcessScheduleUnit parmUnit(ProcessScheduleUnit _unit = unit)`	The unit of time that the series runs in. The unit can be minutes or hours.
`public ProcessScheduleInterval parmPollingInterval(ProcessScheduleInterval _pollingInterval = pollingInterval)`	For polled processes, this value is an integer that, together with the unit, defines how often the process runs.
`public ProcessScheduleDateTime parmStartDate(ProcessScheduleDateTime _startDate = startDate)`	The start date of the series. The time should be set to the empty time.
`public ProcessScheduleDateTime parmEndDate(ProcessScheduleDateTime _endDate = endDate)`	The end date of the series. The time should be set to the empty time.
`public ProcessScheduleDateTime parmTime(ProcessScheduleDateTime _time = time)`	The time when the series should run. The date should be set to the empty date.

Methods that are applicable to the week unit

METHOD	DESCRIPTION
`public NoYes parmOnSunday(NoYes _onSunday = onSunday)`	The days of the week that you want the process to run on by selecting the appropriate methods for the day of the week. For example, **parmOnMonday()** will run the job on a Monday.

Methods that are applicable to the day unit

METHOD	DESCRIPTION
`public NoYes parmDoesRepeatEveryNumberOfDays(NoYes _doesRepeatEveryNumberOfDays = doesRepeatEveryNumberOfDays)`	Indicates whether the process should run every *X* number of days.
`public int parmDailyRepeatInterval(int _dailyRepeatInterval = dailyRepeatInterval)`	Indicates the number of days.
`public NoYes parmDoesRepeatEveryWeekDay(NoYes _doesRepeatEveryWeekDay = doesRepeatEveryWeekDay)`	Indicates whether the process should run every weekday.

Methods that are applicable to the month unit

METHOD	DESCRIPTION
`public NoYes parmDoesRepeatOnDayOfMonth(NoYes _doesRepeatOnDayOfMonth = doesRepeatOnDayOfMonth)`	The process should run on a specific day of every month.
`public Day parmMonthlyRepeatDayOfMonth(Day _monthlyRepeatDayOfMonth = monthlyRepeatDayOfMonth)`	The day of month when the process should run.

Modifying background processes

System admins can modify the polling interval and unit in the process automation framework. However, many background processes that currently exist have their own specific UI that is built to manage these changes. Microsoft provides a way to programmatically modify these values via the following APIs.

METHOD	DESCRIPTION
`public static ProcessScheduleSeriesPollingDetails getPollingDetailsForSeries(ProcessScheduleTypeName _typeName, ProcessScheduleSeriesName _seriesName)`	Gets the polling interval, the unit, and the next scheduled date/time for a polled process.
`public static void setPollingDetailsForSeries(ProcessScheduleTypeName _typeName, ProcessScheduleSeriesName _seriesName, ProcessScheduleSeriesPollingDetails _pollingDetails)`	Enables changes to the polling interval, the unit, and the next scheduled date/time for a polled process.

Validating background process settings

In version 10.0.13, the process automation framework lets system admins modify background process settings via the **Edit background process** section. Some background processes have restrictions on the frequency of their runs. Microsoft has introduced an interface that a background process can implement. When this interface is invoked, it enables the background process to ensure that the unit and polling interval are within their allowed range.

The following example prevents this process from ever being run every minute or every hour. The process can run a maximum of one time per day. However, a process can implement the rules in such a way that more frequent runs are required.

```
using System.ComponentModel.Composition;

// Provider to validate background settings.
[Export(identifierStr(Dynamics.AX.Application.ProcessScheduleISeriesValidateBackgroundDialog))]
[ExportMetadata(extendedTypeStr(ProcessScheduleTypeName), 'ProcessAutomationExploder')]
internal final class ProcessScheduleExplodeAutomationBackgroundDialogValidationProvider implements
ProcessScheduleISeriesValidateBackgroundDialog
{

    public boolean
    validateBackgroundProcessParameters(ProcessScheduleSeriesBackgroundValidationParameters
    _validationParameters)
    {
        ProcessScheduleUnit currentUnit = _validationParameters.parmUnit();
        if (currentUnit == ProcessScheduleUnit::Minute || currentUnit == ProcessScheduleUnit::Hour)
        {
            SysDictEnum enum = new SysDictEnum(enumNum(ProcessScheduleUnit));
            throw Error(
                strFmt("@ProcessAutomationFramework:ProcessScheduleExplodeProcessInvalidUnit",
                enum.value2Label(ProcessScheduleUnit::Minute),
                enum.value2Label(ProcessScheduleUnit::Hour)));
        }
        return true;
    }
}
```

ProcessScheduleISeriesValidateBackgroundDialog interface

The **ProcessScheduleISeriesValidateBackgroundDialog** interface enables background processes to validate user input when users edit background settings via **ProcessScheduleSeriesBackgroundDialog**.

METHOD	DESCRIPTION
`boolean validateBackgroundProcessParameters(ProcessScheduleSeriesBackgroundValidationParameters _validationParameters)`	Implements any validation rules that you have for the process.

ProcessScheduleSeriesBackgroundValidationParameters class

The **ProcessScheduleSeriesBackgroundValidationParameters** class contains the validation parameters that are validated for the background processes that are being edited.

METHOD	DESCRIPTION
`public UserId parmOwnerId(UserId _ownerId = ownerId)`	The owner of the process. It will be used when batch jobs are created, because batch jobs will be created under this user's context.
`public ProcessScheduleUnit parmUnit(ProcessScheduleUnit _unit = unit)`	The unit of time.
`public ProcessScheduleInterval parmPollingInterval(ProcessScheduleInterval _pollingInterval = pollingInterval)`	The polling interval. It defines the number of units of time (as specified by **parmUnit()**) that the process should be run.
`public ProcessScheduleDateTime parmPolledNextScheduledDateTime(ProcessScheduleDateTime _polledNextScheduledDateTime = polledNextScheduledDateTime)`	The next scheduled run of the process in Coordinated Universal Time (UTC).

METHOD	DESCRIPTION
`public ProcessScheduleDateTime parmSleepFromTime(ProcessScheduleDateTime _polledSleepFromTime = polledSleepFromTime)`	Specifies when the sleep should start. The process automation framework lets system admins put a process to sleep for a time range. The process isn't run during this time range, regardless of the setting of **parmPolledNextScheduleDateTime()**. This time range is a maximum of 16 hours and can span the date boundary.
`public ProcessScheduleDateTime parmSleepToTime(ProcessScheduleDateTime _polledSleepToTime = polledSleepToTime)`	Specifies when the sleep should end. The process automation framework lets system admins put a process to sleep for a time range. The process isn't run during this time range, regardless of the setting of **parmPolledNextScheduleDateTime()**. This time range is a maximum of 16 hours and can span the date boundary.

Process parameters

2/18/2021 • 4 minutes to read • Edit Online

In most cases, a process must store custom parameters that are specific to its processes. For example, a process might require a date range or a customer number. You must create your own user interface (UI) and custom tables to show and store these parameters. If a type doesn't have any parameters, you can skip this task.

When a user creates a series in the UI, the **Create series** wizard hosts multiple form parts, each of which contains a related set of parameters. The form parts for the process contain the UI that the user uses to enter the parameters. These form parts are built by the developer of the process and provided through type registration. A form part implements interfaces that let you initialize, validate, and write the custom parameters.

The custom parameter tables typically have two types of records:

- A template record that is bound to the series that serves as a template for all occurrences.
- A record that is specific to an occurrence and contains the parameters that will be used when that occurrence runs. Users can override the parameters for each occurrence as they require.

Parameter tables typically have one foreign key (**RecId**) to the **ProcessScheduleSeries** table and another foreign key (**RecId**) to the **ProcessScheduleOccurrence** table. The template record has a series foreign key, but it doesn't have a foreign key to the occurrence. All other records have both foreign keys.

The following interfaces are used to maintain these parameters.

ProcessScheduleParametersIInitialize interface

The **ProcessScheduleParametersIInitialize** interface lets you initialize parameters when the user interacts with the UI of the process automation framework. The form part that is built for the wizard that shows process-specific parameters implements this interface.

ProcessScheduleParametersIValidate interface

The **ProcessScheduleParametersIValidate** interface lets you validate the parameters that the user enters in the form part.

ProcessScheduleParametersIWrite interface

The **ProcessScheduleParametersIWrite** interface lets you write the parameters to their custom parameter tables.

Example

In the following example, the three interfaces that were just described are used for a sample test process. In this example, the form part contains a single string that is known as a *message*.

```
[Form]
public class ProcessScheduleSampleUptakeFirstFormPart
extends ProcessScheduleParametersFormPart
implements ProcessScheduleParametersIWrite, ProcessScheduleParametersIValidate,
ProcessScheduleParametersIInitialize
{

    private ProcessScheduleSchedulingContract schedulingContract;
```

```
    public void setSchedulingContract(ProcessScheduleSchedulingContract _schedulingContract)
    {
        schedulingContract = _schedulingContract;
    }

    public void initializeForSeriesCreate()
    {
        str text = strFmt("@ProcessAutomationFramework:ProcessScheduleSeriesTestTypeInitSeriesCreate",
curExt());
        ProcessScheduleSampleUptakeParameters_Message.text(text);
    }

    public void initializeForSeriesUpdate()
    {
        ProcessScheduleSampleUptakeParameters parameters =

ProcessScheduleSampleUptakeParameters::findForProcessScheduleSeries(schedulingContract.processScheduleSeries
, true);
        ProcessScheduleSampleUptakeParameters_Message.text(parameters.Message);
    }

    public void initializeForOccurrenceCreate()
    {
        str text = strFmt("@ProcessAutomationFramework:ProcessScheduleSeriesTestTypeInitOccurrenceCreate",
curExt());
        ProcessScheduleSampleUptakeParameters_Message.text(text);
    }

    public void initializeForOccurrenceUpdate()
    {
        ProcessScheduleSampleUptakeParameters parameters =

ProcessScheduleSampleUptakeParameters::findForProcessScheduleOccurrence(schedulingContract.processScheduleOc
currence);
        ProcessScheduleSampleUptakeParameters_Message.text(parameters.Message);
    }

    public void createScheduleSeries(ProcessScheduleSchedulingContract
    _schedulingContract)
    {
        ProcessScheduleSampleUptakeParameters sampleParameters;
        sampleParameters.Message = ProcessScheduleSampleUptakeParameters_Message.valueStr();
        sampleParameters.ProcessScheduleSeries = _schedulingContract.processScheduleSeries.RecId;
        sampleParameters.insert();
    }

    public void updateScheduleSeries(ProcessScheduleSchedulingContract
    _schedulingContract)
    {
        ProcessScheduleSampleUptakeParameters parameters =

ProcessScheduleSampleUptakeParameters::findForProcessScheduleSeries(_schedulingContract.processScheduleSerie
s, true);

        if (parameters)
        {
            ttsbegin;
            parameters.Message = ProcessScheduleSampleUptakeParameters_Message.valueStr();
            parameters.update();
            ttscommit;
        }
    }

    public void createScheduledOccurrence(ProcessScheduleSchedulingContract _schedulingContract)
    {
        ProcessScheduleSampleUptakeParameters occurrenceParameters;
        occurrenceParameters.ProcessScheduleOccurrence =
_schedulingContract.processScheduleOccurrence.RecId;
```

```
        occurrenceParameters.Message = ProcessScheduleSampleUptakeParameters_Message.valueStr();
        occurrenceParameters.insert();
    }

    public void updateScheduledOccurrence(ProcessScheduleSchedulingContract
    _schedulingContract)
    {
        ProcessScheduleSampleUptakeParameters parameters =

ProcessScheduleSampleUptakeParameters::findForProcessScheduleOccurrence(_schedulingContract.processScheduleO
ccurrence,
            true);

        ttsbegin;

        parameters.ProcessScheduleSeries = _schedulingContract.processScheduleSeries.RecId;
        parameters.ProcessScheduleOccurrence = _schedulingContract.processScheduleOccurrence.RecId;
        parameters.Message = ProcessScheduleSampleUptakeParameters_Message.valueStr();

        if (parameters.RecId != 0)
        {
            parameters.update();
        }
        else
        {
            parameters.insert();
        }

        ttscommit;
    }

    public boolean validate()
    {
      boolean isValid = true;
      if (ProcessScheduleSampleUptakeParameters_Message.valueStr() == '')
      {
          isValid = checkFailed("@ProcessAutomationFramework:ProcessScheduleSeriesTestTypeMessageWarning");
      }
      return isValid;
    }
}
```

ProcessScheduleIDeleteOccurrence interface

Implement the **ProcessScheduleIDeleteOccurrence** interface to receive an event that indicates that a user or the system has deleted occurrences. The parameters that are related to that occurrence should be deleted.

This interface is invoked via **SysPlugIn** for a specific type. Use the type name that was created when the type was registered.

Note that a Microsoft Azure SQL Database temp table is passed in so that you can do set-based deletes.

```
using System.ComponentModel.Composition;

// The VendPaymProposalAutomationOccurrenceDeleteProvider class is designed to handle
// deleting the appropriate VendPaymProposalAutomationCriteria records when ProcessScheduleOccurrence
records are deleted.
[ExportMetadata(extendedTypeStr(ProcessScheduleTypeName), 'VendPaymProposalAutomation')]
[Export(identifierStr(Dynamics.AX.Application.ProcessScheduleIDeleteOccurrence))]
internal final class VendPaymProposalAutomationOccurrenceDeleteProvider
implements ProcessScheduleIDeleteOccurrence
{
    private ProcessScheduleSeriesOccurrenceTmp occurrencesExplodedTmp;

    [Wrappable(false)]
    public void deleteOccurrences(ProcessScheduleSeriesOccurrenceTmp _occurrencesExplodedTmp)
    {
        this.initialize(_occurrencesExplodedTmp);
        this.deleteVendPaymProposalAutomationCriteria();
    }

    private void initialize(ProcessScheduleSeriesOccurrenceTmp _occurrencesExplodedTmp)
    {
        occurrencesExplodedTmp.linkPhysicalTableInstance(_occurrencesExplodedTmp);
    }

    private void deleteVendPaymProposalAutomationCriteria()
    {
        VendPaymProposalAutomationCriteria automationCriteria;
        automationCriteria.skipDeleteActions(true);
        automationCriteria.skipDataMethods(true);
        automationCriteria.skipAosValidation(true);
        automationCriteria.skipDatabaseLog(true);
        automationCriteria.skipEvents(true);

        delete_from automationCriteria
            exists join occurrencesExplodedTmp
            where automationCriteria.ProcessScheduleOccurrence ==
occurrencesExplodedTmp.ProcessScheduleOccurrence;
    }
}
```

ProcessScheduleIDeleteSeries interface

The **ProcessScheduleIDeleteSeries** interface resembles **ProcessScheduleIDeleteOccurrence**. The event is invoked whenever a series is deleted. You should delete all parameter records for all occurrences. These records include the series template record.

A SQL Database temp table is passed in so that you can do set-based deletes.

```
using System.ComponentModel.Composition;

// The VendPaymProposalAutomationSeriesDeleteProvider class is designed to handle
// deleting the appropriate VendPaymProposalAutomationCriteria records when ProcessScheduleSeries records
are deleted.
[ExportMetadata(extendedTypeStr(ProcessScheduleTypeName), 'VendPaymProposalAutomation')]
[Export(identifierStr(Dynamics.AX.Application.ProcessScheduleIDeleteSeries))]
internal final class VendPaymProposalAutomationSeriesDeleteProvider
implements ProcessScheduleIDeleteSeries
{

    [Wrappable(false)]
    public void deleteSeries(RefRecId _seriesRecId)
    {
        this.deleteVendPaymProposalAutomationCriteria(_seriesRecId);
    }

    private void deleteVendPaymProposalAutomationCriteria(RefRecId _seriesRecId)
    {
        VendPaymProposalAutomationCriteria automationCriteria;
        automationCriteria.skipDeleteActions(true);
        automationCriteria.skipDataMethods(true);
        automationCriteria.skipAosValidation(true);
        automationCriteria.skipDatabaseLog(true);
        automationCriteria.skipEvents(true);

        delete_from automationCriteria
            where automationCriteria.ProcessScheduleSeries == _seriesRecId;
    }
}
```

ProcessScheduleIExplodeOccurrences interface

When a user creates a new series through the UI, all the future occurrences are generated. Therefore, if the series runs every day, the process automation framework creates an occurrence for every day. This action is known as *generating the series*. The **ProcessScheduleIExplodeOccurrences** event is fired when the series is generated. The series template record should be used as a template to create parameter records for each occurrence in the parameter tables.

A SQL Database temp table is passed in so that you can do set-based creation of parameter records for optimal performance.

In the following example, a parameter table stores a single parameter that is named **Type**. This parameter isn't related to the process automation framework type but is specific to cash flow forecasting.

```
using System.ComponentModel.Composition;

// Provider for cash flow forecast automation generate occurrences.
[Export(identifierStr(Dynamics.AX.Application.ProcessScheduleIExplodeOccurrences))]
[ExportMetadata(extendedTypeStr(ProcessScheduleTypeName), 'LedgerCovTotalProcessAutomation')]
internal final class LedgerCovTotalProcessAutomationExplodeOccurrencesProvider
implements ProcessScheduleIExplodeOccurrences
{
    private void new()
    {
    }

    [Wrappable(false)]
    public void explodeOccurrences(ProcessScheduleSeriesOccurrenceTmp _occurrencesExplodedTmp)
    {
        LedgerCovTotalProcessAutomationSchedulingParameters parameters;
        LedgerCovTotalProcessAutomationSchedulingParameters
        parametersSeriesRecord;

        insert_recordset parameters
        (
            Type,
            ProcessScheduleSeries,
            ProcessScheduleOccurrence
        )

        select Type, ProcessScheduleSeries from parametersSeriesRecord
            where parametersSeriesRecord.ProcessScheduleOccurrence == 0
            join ProcessScheduleOccurrence from _occurrencesExplodedTmp
            where _occurrencesExplodedTmp.ProcessScheduleSeries ==
parametersSeriesRecord.ProcessScheduleSeries
                && _occurrencesExplodedTmp.TypeName ==
LedgerCovTotalProcessAutomationConstants::RegisteredTypeName;
    }
}
```

User-configurable queries

2/18/2021 • 2 minutes to read • Edit Online

This topic describes how to create configurable queries and use them with the process automation framework. If a process won't support user-configurable queries via the **SysQueryForm** form, you can skip this task.

The process automation framework provides limited support for custom queries via the **SysQueryForm** form. A custom query lets a user add custom criteria to limit how a process runs. The framework has logic to extract user-provided custom criteria and tables to store those criteria. The custom query criteria are stored for each occurrence of a given series and can be modified individually. The framework also provides an API to apply the custom criteria to the query that is used to run the process for each occurrence.

> **NOTE**
> When a user applies query criteria, the whole query object isn't saved. Instead the query criteria are saved individually, to allow for better support of query extensions. Therefore, extensions that are made to existing queries for existing query criteria should not cause breaking changes when this approach is used. In this case, a new extension should not require modification or re-creation of the query criteria for a saved series or occurrences. However, modification or re-creation of the query criteria is allowed.

ProcessScheduleIQueryable interface

The **ProcessScheduleIQueryable** interface retrieves the original query (that is, the query as it was before the user modified it) and the user-modified query. These queries are used either to apply criteria when the process runs or to extract criteria when the user makes changes. These criteria are stored by the process automation framework.

This interface is accessed in the criteria form for a given implementation of process automation. For an example of access to this interface, see the **VendPaymProposalAutomationCriteria** form. That form also has a sample implementation of the **SysQueryForm** form.

METHOD	DESCRIPTION
`public Query getOriginalQuery()`	This method gets the original, unmodified query to use as a basis for comparison.
`public Query getQueryForApplicationOrExtractionOfQueryCriteria()`	This method gets the query that has been or will be modified, and that is used to apply or extract query criteria.

> **NOTE**
> The query that is used on the implementation of **ProcessScheduleIQueryable** must have the same structure as the query that is used during the run of the underlying process that is being automated. Any structural deviation that isn't additive in nature will cause runtime errors when the saved query criteria are applied at runtime. To ensure that the query structure remains the same, you should either use a designed query or use shared logic that builds up the query.

ProcessScheduleQueryCriteriaApplicator class

The **ProcessScheduleQueryCriteriaApplicator** class is used to apply the saved query criteria for a given occurrence to the runtime instance of the query that is used when a process is run. This API must be called by an

uptaking process at a point in the run where the query is ready to accept the saved criteria. If a designed query or shared logic that builds up the query is used, this call can typically occur after the query has been correctly initialized. For an example that shows how this API is used, see the **CustVendCreatePaymJournal.constructFromAutomationExecutionContract** method.

METHOD	DESCRIPTION
`public static void applyCriteriaForOccurrenceExecution(Query _queryToApplyCriteria, RefRecId _scheduleOccurrenceRecId)`	

Run processes

2/18/2021 • 4 minutes to read • Edit Online

To run in the process automation framework, a process must implement the **ProcessAutomationTask** interface. The framework uses this interface to provide an instance of **ProcessScheduleWorkItem**. That instance contains information about the series, the occurrence that is being run (if applicable), and the execution ID. You must create the batch task that has to be run, and you must provide the task to the framework. The framework then creates the batch header and the tasks that are provided.

Polled processes don't have occurrences, because they might be run frequently, and those frequent runs will create many more occurrences than you want to track. Instead, you use a unique execution ID to track every run of a polled process. An execution ID is also assigned to scheduled processes.

The **getListOfWorkToBePerformed** method determines whether there is work that must be done. If there is, the method returns a list of batch-enabled classes that inherit from **SysOperationServiceController**. This method must be efficient and fast, because it's run in the context of the polling process. Therefore, you should not do any work in this method except check whether work must be done and create batch tasks for any work that must be done. It's OK if the method returns an empty list, because an empty list indicates that no work must be done. In this case, the process automation framework doesn't create any batch processes.

The process automation framework supports a list for those processes that do parallel processing and that want multiple batch tasks. If the process that is being run is an older process that implements **RunBaseBatch**, it can't be returned through the list. In this case, you have two options:

- Convert the process to **SysOperationServiceController**. For more information, see SysOperations Framework. This option is recommended, if it's feasible.
- Wrap the legacy **RunBaseBatch** class with a new class that inherits from **SysOperationServiceController**. For an example, see **LedgerCovTotalProcessAutomationProcessor** in the Application Object Tree (AOT).

The following example shows an implementation of the **ProcessAutomationTask** interface.

```
using System.ComponentModel.Composition;

// The test class to test the process automation task engine.
[ExportMetadataAttribute(classStr(ProcessAutomationTask),
classStr(ProcessAutomationTaskTestImplementation))]
[ExportAttribute(identifierStr('Microsoft.Dynamics.AX.Application.ProcessAutomationTask'))]
internal final class ProcessAutomationTaskTestImplementation extends ProcessAutomationTask
{

    protected boolean isProcessAutomationEnabledForThisTask()
    {
        return true;
    }

    protected List getListOfWorkToBePerformed()
    {
        ProcessScheduleWorkItem scheduleWorkItemToExecute = this.parmProcessScheduleWorkItem();
        List taskList = new List(Types::Class);
        ProcessAutomationTestProcess testProcess =
ProcessAutomationTestProcess::construct(scheduleWorkItemToExecute);
        taskList.addEnd(testProcess);
        return taskList;
    }

    // Gets and sets the batch job caption that will be used by the batch job scheduled to run this process.
    // Returns the batch caption that will be used for the batch job that will perform the task.
    protected BatchCaption batchJobCaption()
    {
        return "@ProcessAutomationFramework:ProcessScheduleExplodeProcessLabel";
    }
}
```

ProcessScheduleWorkItem class

The **ProcessScheduleWorkItem** class has many pieces of information and is designed to represent a process that must be run.

METHOD	DESCRIPTION
`public ProcessExecutionId parmExecutionId(ProcessExecutionId _executionId = executionId)`	The execution ID is used to log errors. A polled process gets a new, unique execution ID every time that it's run. Because scheduled processes are only ever run one time for each occurrence, each occurrence only ever has one execution ID.
`public RefRecId parmProcessScheduleOccurrenceRecId(RefRecId _scheduleOccurrenceRecId = scheduleOccurrenceRecId)`	The occurrence that is being run. Use this method to reference occurrence-specific parameter information in parameter tables. Polled processes don't have a **RecId** value for an occurrence.
`public RefRecId parmProcessScheduleSeriesPatternRecId(RefRecId _scheduleSeriesPatternRecId = scheduleSeriesPatternRecId)`	The series pattern that the process is associated with. Currently, series can have only one pattern. All parameter records typically have a foreign key to this pattern.
`public ProcessScheduleProcessType parmProcessScheduleType(ProcessScheduleProcessType _scheduleType = scheduleType)`	The schedule type of the process: polled or scheduled.
`public ProcessScheduleTypeName parmProcessScheduleTypeName(ProcessScheduleTypeName _scheduleTypeName = scheduleTypeName)`	The type name that the process and series are associated with, such as **VendPaymentProposal**. This name is an internal developer name and isn't shown to the user.

METHOD	DESCRIPTION
`public List parmLegalEntityList(List _legalEntityList = legalEntityList)`	The list of legal entities that the process will be run against. If the process is a global process, this list will be **null**. If the process is run against a single company, this list will contain a single company. Multiple companies aren't currently supported, but they might be supported in the future.
`public Name getName()`	Returns the occurrence name. If a type is polled, this method returns the series name.
`public ProcessScheduleSeriesName parmSeriesName(ProcessScheduleSeriesName _seriesName = seriesName)`	The name of the series.
`public Name parmOccurrenceName(Name _occurrenceName = occurrenceName)`	The occurrence name. If the process is a polled process, the value will be empty.
`public List parmTaskList(List _taskList = taskList)`	The list of batch tasks that the process automation framework should add to the batch. If this list is **null**, it's assumed that no work must be done, and nothing will be created in a batch.
`public ProcessScheduleDateTime parmScheduledDateTime(ProcessScheduleDateTime _scheduledDateTime = scheduledDateTime)`	The date and time when the process was scheduled to run. This date and time might differ from the actual date and time when the process runs.
`public UserGroupId parmOwnerId(UserGroupId _ownerId = ownerId)`	The owner of the occurrence that is being run.
`public void initializeFromScheduleWorkItem(ProcessScheduleWorkItem _item)`	Initializes an instance of **ProcessScheduleWorkItem** from another instance.

Log results and messages

2/18/2021 • 5 minutes to read • Edit Online

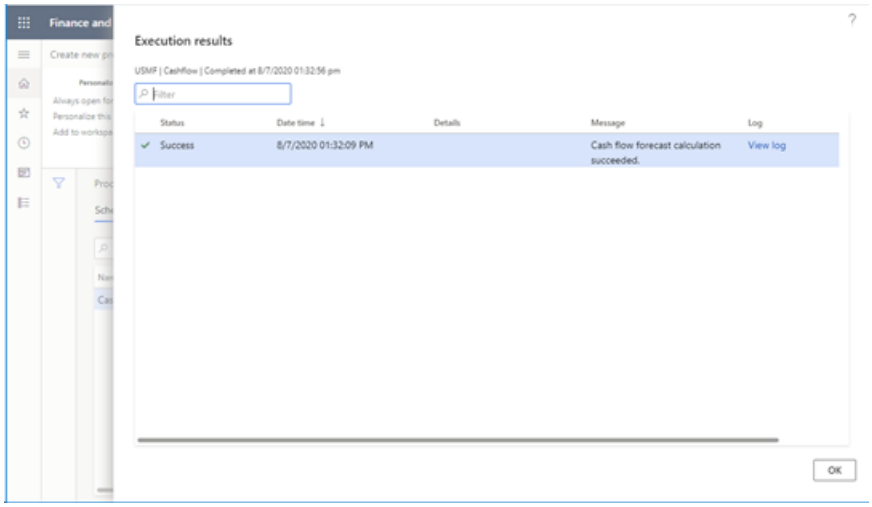
The process automation framework supports logging of results and messages. There are two reasons why a process should log results and messages:

- Results and the message log communicate the state of a process to the system admin or other roles that have access. It's important that process results be monitored and seen by someone. If failures occur, they can be fixed, or an issue can be raised with the process owner.
- Results should communicate what the process did. For example, if the process posts vendor invoices, the results can show all the vendor invoices. In this case, each result will show the status and a link to the vendor invoice.

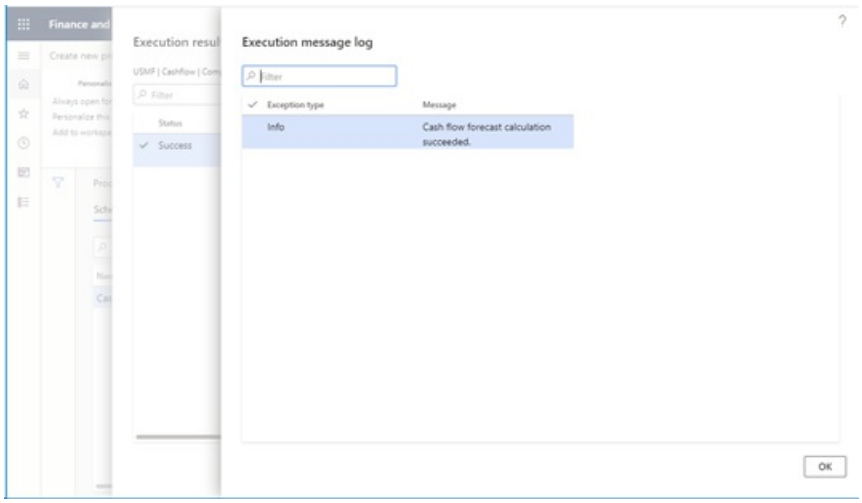
Results and messages are a multilevel logging system. A process has one or more results. Each result has one or more messages that are specific to it. A message is a composition child of the result. A result is typically something that the process is processing. For example, if the process posts vendor invoices, you log each vendor invoice as a result. Then, for each posting of each vendor invoice, you can log multiple messages that are associated with that result. If a vendor invoice is successfully posted, it's OK to leave the message log blank, because the success of the operation should be obvious when users look at the result. If warnings occur, you should write them to the message log, even if posting was successful. These messages provide transparency, so that users can see what each process is doing and what the results are.

Both scheduled processes and polled processes support logging of results and messages. All processes should create a result. At a minimum, the result should communicate the fact that everything was successful. Processes that do work that affects user work and is visible to users should create more detailed results. For the vendor invoice posting example that was used earlier, users might want to see that their invoices have been posted. Results that don't show that information can cause confusion, even if the information is available in other ways.

The following illustration shows the result view.



The following illustration shows the message view. To open this view, select **View log** in the **Log** column in the result view.



ProcessExecutionSourceLink table

The **ProcessExecutionSourceLink** table contains the results that are created while the processing is running. This table contains **RefTableId** and **RefRecId** fields. These fields are links to any source record in Microsoft SQL Server and are typically something that the process is processing. This table also contains **Header** and **Message** fields. The **Header** field will be shown as a column in the result grid. The message can be anything that you want it to be.

For example, if vendor payment proposal is creating a payment journal, the **RefTableId** value is the table ID of the **LedgerJournalTable** table. The **RefRecId** value is the **RecId** value of the **LedgerJournalTable** record that the payment journal created by the running process. In this case, you can set the **Header** field to the journal number. You can even make this value a jump reference, so that users can select the journal number to go directly to the payment journal. You can set the **Message** field to any message that you want to show, such as **Payment journal created successfully**.

If the process is processing many items (for example, if it's posting many invoices), you can create a **ProcessExecutionSourceLink** record for each invoice.

If the number of items that the process is processing is very large (in the millions), and users don't have to see the details of each item, consider summarizing the items into batches. For example, the process for subledger transfer to the general ledger creates a **ProcessExecutionSourceLink** record for each transfer ID that is transferred to the general ledger, not for each voucher that is transferred.

METHOD	DESCRIPTION
`public static void insertSourceLinks(ProcessScheduleTypeName _typeName, ProcessExecutionSourceLinkTmp _tmp)`	This method is new in version 10.0.14. It inserts many results into the **ProcessExecutionSourceLink** table by using set-based insert.
`public static ProcessExecutionSourceLink insertSourceLink(ProcessExecutionSourceLinkItem _sourceLinkItem)`	This method inserts a record into the **ProcessExecutionSourceLink** table.

ProcessExecutionSourceLinkItem class

Instantiate the **ProcessExecutionSourceLinkItem** class, and fill it with values that are required to correctly show your source item.

METHOD	DESCRIPTION
`public static ProcessExecutionSourceLinkItem newFromProcessScheduleWorkItemAndStatus(ProcessScheduleWorkItem _workItem, ProcessExecutionSourceStatus _status)`	Use this constructor to create an instance of **ProcessExecutionSourceLinkItem**. This method correctly initializes many of the required fields from **ProcessScheduleWorkItem**.
`public static ProcessExecutionSourceLinkItem newFromProcessExecutionSourceLink(RefRecId _processExecutionSourceLinkRecId)`	This method constructs an instance of **ProcessExecutionSourceLinkItem** and initializes the instance by using the specified record ID of a **ProcessExecutionSourceLink** record.
`public RefRecId parmSourceRecId(RefRecId _sourceRecId = sourceRecId)`	Set the record ID of the source record. For example, this value might be the record ID of the vendor invoice header table.
`public RefTableId parmSourceTableId(RefTableId _sourceTableId = sourceTableId)`	Set the table ID of the source table. For example, this value might be the table ID of the vendor invoice header table.
`public ProcessExecutionSourceLinkHeader parmHeader(ProcessExecutionSourceLinkHeader _header = header)`	Set the value for the header field. For the vendor invoice posting example that was used earlier, this value might be the invoice number.
`public ProcessExecutionSourceLinkMessage parmMessage(ProcessExecutionSourceLinkMessage _message = message)`	Set the message. For the vendor invoice posting example that was used earlier, this value might be **Posting successful**.
`public ProcessExecutionId parmExecutionId(ProcessExecutionId _executionId = executionId)`	This method sets the execution ID. This value was provided via **ProcessScheduleWorkItem** in the implementation of the **ProcessAutomationTask** interface.

ProcessExecutionMessageLog table

The **ProcessExecutionMessageLog** table contains messages that are related to a single **ProcessExecutionSourceLink** record. You can write any type of message to this table. The message will then be shown to users.

METHOD	DESCRIPTION
`public static void insertMessages(ProcessScheduleTypeName _typeName, ProcessExecutionMessageLogTmp _tmp)`	This method inserts messages into the **ProcessExecutionMessageLog** table by using a set-based insert.
`public static ProcessExecutionMessageLog insertMessage(ProcessExecutionMessageLogItem _errorLogItem)`	This method inserts a message into the message log.

ProcessExecutionMessageLogItem class

The message log stores messages as both strings and label IDs. You don't have to set both values. Label IDs are preferred, because they support translation of messages in the user interface (UI) for the message log. However, the messages are provided for backward compatibility with processes that don't support logging of label IDs.

Use the appropriate constructor for your scenario.

METHOD	DESCRIPTION
`public static ProcessExecutionMessageLogItem newFromProcessExecutionSourceLinkAndMessage(RefRecId _processExecutionSourceLinkRecId, Exception _exception, ProcessExecutionMessage _message)`	
`public static ProcessExecutionMessageLogItem newFromProcessExecutionSourceLinkAndLabel(RefRecId _processExecutionSourceLinkRecId, Exception _exception, LabelId _labelId, container _labelParameters)`	

Customize the user interface

2/18/2021 • 7 minutes to read • Edit Online

The process automation framework supports some customizations of the user interface (UI). Most of this topic is optional, because the framework provides default values for everything. The only exception is the **ProcessScheduleSeries** form. If you intend to show the **ProcessScheduleSeries** form for a specific product area, customizations are required so that the framework can show data that is specific to that product area.

Weekly calendar view

ProcessScheduleIBuildOccurrenceCard interface

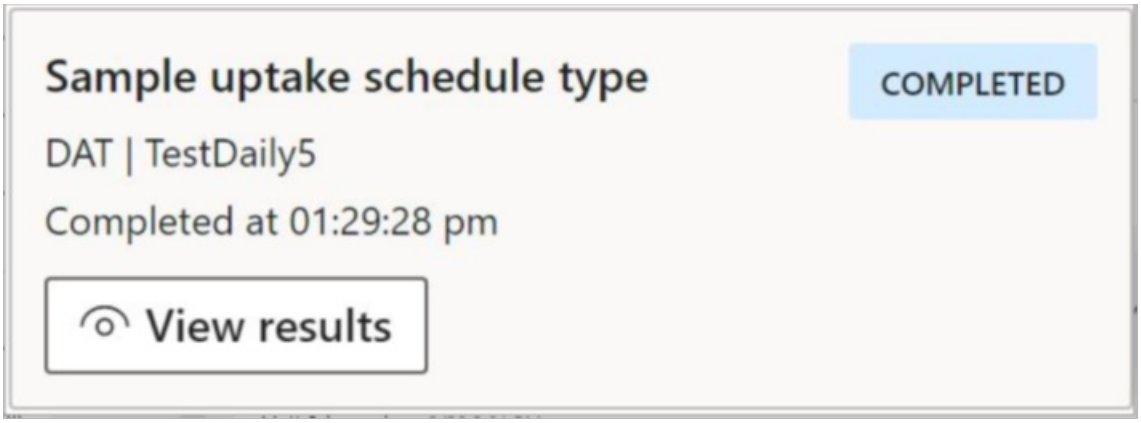
The **ProcessScheduleIBuildOccurrenceCard** interface lets you customize the appearance of occurrence cards in the weekly calendar view. There is a static method on the interface for each status of an occurrence: **Scheduled**, **Waiting**, **Running**, **Successful**, **Failed**, and **Disabled**. You can create a customized occurrence card for each status value. Each of these methods returns an instance of **ProcessScheduleOccurrenceCard**.

The process automation framework provides a default implementation in the **ProcessScheduleOccurrenceCardBuilder** class. You inherit from this class and override the functionality as you require. You then register your derived class via the **SysPlugin** for your specific type. The registration process resembles the process for many of the plug-ins in the framework documentation.

An instance of **ProcessScheduleOccurrenceCardBuilderContract** is passed into each of the methods and can be used to retrieve information about the occurrence. The derived class can invoke the default implementation for each static method that returns the **ProcessScheduleOccurrenceCard** instance, modify whatever is required, and return it.

ProcessScheduleOccurrenceCard class

The **ProcessScheduleOccurrenceCard** class lets you customize the appearance of an occurrence card that is shown in the calendar view. The first two lines are controlled by the process automation framework and can't be modified. In the following illustration, the subheader is the **Completed at** phrase, and the status message is the word **Completed** that has a blue background.



METHOD	DESCRIPTION
`public str parmSubHeader(str _subHeader = cardSubHeader)`	The subheader, which is the third line of the occurrence card that is shown in the previous illustration.

METHOD	DESCRIPTION
`public str parmStatusMessage(str _statusMessage = statusMessage)`	The status message, which represents the status of the process and has a colored background.
`public ProcessExecutionOccurrenceCardStatusColor parmStatusColor(ProcessExecutionOccurrenceCardStatusColor _statusColor = statusColor)`	The color of the background for the status message.

ProcessScheduleIShowOccurrenceCalendarView interface

The **ProcessScheduleIShowOccurrenceCalendarView** interface must be implemented by forms that will show the weekly calendar view. The **ProcessScheduleSeries** form is an example of a form that implements this interface.

METHOD	DESCRIPTION
`ProcessScheduleOccurrenceCalendarViewContract getProcessScheduleOccurrenceCalendarViewContract()`	Return the contract that the weekly view will use to determine which types should be shown.
`void refreshAfterChangeToCalendarView()`	This value is a callback from the weekly view. It indicates that the parent form should be refreshed because of changes in the weekly calendar view.

ProcessScheduleOccurrenceCalendarViewContract class

Use the **ProcessScheduleOccurrenceCalendarViewContract** class to limit the series that the weekly calendar view should show. For an example, see **ProcessScheduleSeries.getProcessScheduleOccurrenceCalendarViewcontract** in the Application Object Tree (AOT).

METHOD	DESCRIPTION
`public static ProcessScheduleOccurrenceCalendarViewContract construct()`	Use this constructor if the intention is to show occurrences for one to many types.
`internal static ProcessScheduleOccurrenceCalendarViewContract newFromScheduleSeries(ProcessScheduleSeries _scheduleSeries)`	Use this constructor if the intention is to show occurrences for a single series.
`public void AddScheduleType(ProcessScheduleTypeName _scheduleTypeName)`	If you aren't showing just one series, use this value to add the types that should be shown.

ProcessScheduleOccurrenceCalendarViewRenderer class

Use the **ProcessScheduleOccurrenceCalendarViewRenderer** class to render the weekly calendar view in an existing form. A form part will be created and correctly initialized. An example of this class is used in the **ProcessScheduleSeries** form.

METHOD	DESCRIPTION
`public static ProcessScheduleICalendarView renderCalendarViewInFormControl(FormGroupControl _containingGroupControl)`	This method renders the weekly calendar view in the specified form group control.

Render interfaces

Several interfaces enable customization of the way that occurrence processes are rendered in the calendar view.

There is one interface for each status that a process can have:

- ProcessScheduleIRenderDisabledOccurrenceCard
- ProcessScheduleIRenderFailedOccurrenceCard
- ProcessScheduleIRenderRunningOccurrenceCard
- ProcessScheduleIRenderScheduledOccurrenceCard
- ProcessScheduleIRenderSuccessfulOccurrenceCard
- ProcessScheduleIRenderWaitingOccurrenceCard

All these interfaces follow the same pattern. An instance of **ProcessScheduleOccurrenceCardRendering** is sent to them. That instance is used to control how the occurrence card is rendered.

For an example, see the **CustVendPaymProposalAutomationOccurrenceCardRenderer** class in the AOT.

ProcessScheduleOccurrenceCardRendering class

METHOD	DESCRIPTION
`public ProcessScheduleOccurrence getOccurrenceBeingRendered()`	This method returns the occurrence that is being rendered on the occurrence card.
`public ProcessExecutionExecutingInformation getOccurrenceExecutionInformation()`	This method returns the running information for the occurrence. This information typically includes the results of the batch job, the start time, and the end time.
`public void makeCardSubHeaderInvisible()`	This method makes the card's subheader invisible. See the illustration earlier in this topic and the content below it to determine which line is the subheader.
`public void makeCardButtonsInvisible()`	This method specifies whether the **Disable** and **Edit** buttons on the occurrence card are invisible.
`public void setColumnsOnOccurrenceCardDetailGroup(int _numberOfColumns)`	This method enables the number of columns on the occurrence card to be customized. By default, there are two columns.
`public FormButtonControl addButtonControl(FormControlName _buttonControlName)`	This method enables a new button to be added to the occurrence card.
`public FormStaticTextControl addStaticTextControl(FormControlName _staticTextControlName)`	This method enables a static text control to be added to the occurrence card.
`public FormStringControl addStringControl(FormControlName _stringControlName, LabelId _stringControlLabel)`	This method adds a string control to the occurrence card.

Series list page

ProcessScheduleISeriesFormController interface

The **Series** list page uses the **ProcessScheduleISeriesFormController** controller to determine which types series will be shown for on the **ProcessScheduleSeries** list page. This class uses the **SysPlugIn** class. The menu item that you use to open the **ProcessScheduleSeries** form is used as the key to invoke the specified plug-in. This key enables each use of this form to customize which types are shown.

```
// Implementation of the ProcessScheduleISeriesFormController for the admin view of the process schedule
Series form.
// This implementation will show series for all process types, both scheduled and polled, on the Series
form.
[Export(identifierStr(Dynamics.AX.Application.ProcessScheduleISeriesFormController))]
[ExportMetadata(classStr(ProcessScheduleISeriesFormController),
menuItemDisplayStr(ProcessScheduleSeriesAdmin))]
internal class ProcessScheduleSeriesFormAdminController implements ProcessScheduleISeriesFormController
{
    [Hookable(false)]
    public ProcessScheduleSeriesFormContract getSeriesFormContract()
    {
        return ProcessScheduleSeriesFormContract::newForAllScheduleTypes();
    }
}
```

ProcessScheduleSeriesFormContract class

The **ProcessScheduleSeriesFormContract** class is a contract that the series list page uses to determine which **ProcessScheduleType** is shown on it. This class can be used in a workspace to show series only for specific types that are related to that workspace.

METHOD	DESCRIPTION
`public void addScheduledScheduleType(ProcessScheduleTypeName _scheduleTypeName)`	Add a specific scheduled type.
`public void addPolledScheduleType(ProcessScheduleTypeName _scheduleTypeName)`	Add a specific polled type.

Results and messages

ProcessExecutionIResultsController interface

The **ProcessExecutionIResultsController** interface lets you customize the results dialog box according to your process. It lets you set the column header label for the **Header** field in the results grid and make the value in the header column a hyperlink. This interface should be implemented by a class that is a plug-in. Here is a sample plug-in.

```
using System.ComponentModel.Composition;

[Export(identifierStr(Dynamics.AX.Application.ProcessExecutionIResultsController))]
[ExportMetadata(classStr(ProcessExecutionIResultsController), 'TestScheduledType')]
public final class ProcessExecutionSampleUptakeExecutionResultsController implements
ProcessExecutionIResultsController
{
    [Hookable(false)]
    public ProcessExecutionResultsDialogContract getResultsDialogContract()
    {
        ProcessExecutionResultsDialogContract contract = ProcessExecutionResultsDialogContract::construct();
        contract.parmSourceLinkHeaderLabel('Sample Header');
        contract.parmShouldSourceLinkHeaderBeLinkToSourceLinkDetails(true);
        return contract;
    }

    [Hookable(false)]
    public void openSourceLinkDetails(RefTableId _refTableId, RefRecId
    _refRecId)
    {
        if (_refTableId == tableNum(SystemParameters))
        {
            Args args = new Args();
            MenuFunction systemParametersMenuFunction = new
MenuFunction(menuItemDisplayStr(SystemParameters), MenuItemType::Display);
            systemParametersMenuFunction.run(args);
        }
    }
}
```

METHOD	DESCRIPTION
`ProcessExecutionResultsDialogContract getResultsDialogContract()`	Return an instance of **ProcessExecutionResultsDialogContract**.
`void openSourceLinkDetails(RefTableId _refTableId, RefRecId _refRecId)`	Implement the logic to open the appropriate menu item that can show the record that is passed in.

ProcessExecutionResultsDialogContract class

The **ProcessExecutionResultsDialogContract** class lets you customize the header column label and specify whether the data in the header column should be rendered as a hyperlink.

METHOD	DESCRIPTION
`public static ProcessExecutionResultsDialogContract newForSourceLinkHeader(LabelId _sourceLinkHeaderLabel, boolean _shouldSourceLinkHeaderBeLinkToSourceLinkDetails)`	Provide the text that should be used as a header column label. Also provide a Boolean value that indicates whether the value in the header column should be rendered as a hyperlink.
`public LabelId parmExecutionResultsDialogCaption(LabelId _executionResultsDialogCaption = executionResultsDialogCaption)`	This method sets the caption for the results dialog box.

ProcessExecutionMessageLogDialog class

The **ProcessExecutionMessageLogDialog** interface enables the message log to be opened in the context of something from the source domain. For example, the message log can be opened from the page for a posted vendor invoice to show the messages that were logged while the vendor invoice was being posted by a process that is enabled for the process automation framework. For this example, the posted vendor invoice page must

implement the **ProcessExecutionMessageLogDialog** interface. By using this interface, you don't have to build your own private results/messaging subsystems.

METHOD	DESCRIPTION
`ProcessExecutionMessageLogContract getContractForMessageLog()`	This method returns an instance of **ProcessExecutionMessageLogContract**.

ProcessExecutionMessageLogContract class

The **ProcessExecutionMessageLogContract** contract lets you limit the message log to a specific item from the source domain. In the **ProcessExecutionSourceLink** table, there must be a record where the **RefRecId** and **RefTableId** values match the values that the contract sends.

METHOD	DESCRIPTION
`public static ProcessExecutionMessageLogContract newForSourceRecord(ProcessScheduleTypeName _typeName, RefTableId _refTableId, RefRecId _refRecId, guid _executionId = emptyGuid())`	This method initializes the contract by using the specified type name, **RefTableId** value, and **RefRecId** value. There should be a matching record in the **ProcessExecutionSourceLink** table. Background processes will have multiple execution IDs. Therefore, the optional parameter for the execution ID should be provided for background processes. For more information, see Type registration.

Regression Suite Automation Tool

2/18/2021 • 5 minutes to read • Edit Online

The Regression suite automation tool (RSAT) significantly reduces the time and cost of user acceptance testing (UAT) of Finance and Operations apps. UAT is typically required before you take a Microsoft application update, or before you apply custom code and configurations to your production environment. RSAT lets functional power users record business tasks by using Task recorder and then convert the recordings into a suite of automated tests, without having to write source code. For more information about Task recorder, see Task recorder resources.

RSAT is fully integrated with Microsoft Azure DevOps for test execution, reporting, and investigation. Test parameters are decoupled from test steps and stored in Microsoft Excel files.

RSAT usage is described in these topics:

- Regression Suite Automation Tool (this topic)
- Regression Suite Automation Tool installation and configuration
- Run Regression Suite Automation Tool test cases
- Validate expected values
- Chain test cases
- Derived test cases
- Regression Suite Automation Tool best practices
- Troubleshoot the Regression Suite Automation Tool

Getting started videos

These videos will help introduce RSAT and get you started.

Use task recorder to create a test case for RSAT

The How to use task recorder to create a test case for the Regression suite automation tool (RSAT) video (shown above) is included in the Finance and Operations playlist available on YouTube.

Create a test plan in Azure DevOps to use with RSAT

The How to create a test plan in Azure DevOps to use with the Regression suite automation tool (RSAT) video (shown above) is included in the Finance and Operations playlist available on YouTube.

How to use RSAT

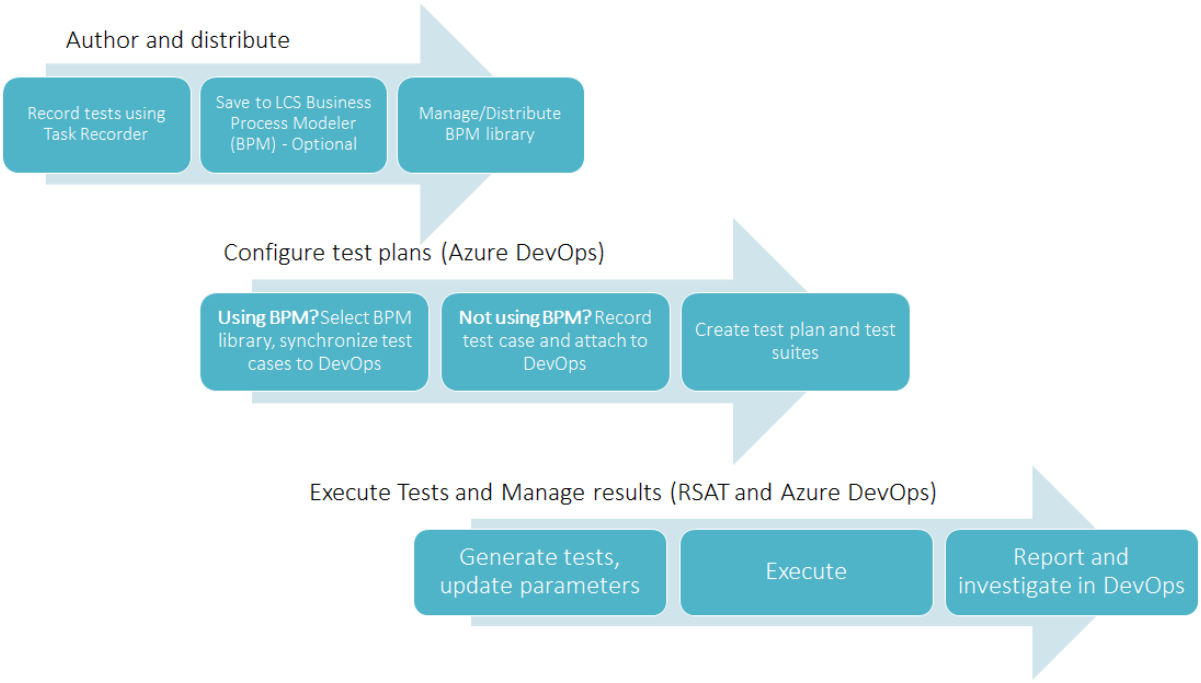
The How to use the Regression suite automation tool (RSAT) video (shown above) is included in the Finance and Operations playlist available on YouTube.

The improved Excel experience in RSAT 2.0

The Improved Excel experience in RSAT 2.0 video (shown above) is included in the Finance and Operations playlist available on YouTube.

End-to-end flow

RSAT is part of the end to end flow described below. RSAT, Microsoft Dynamics Lifecycle Services (LCS), and Azure DevOps provide a set of tools for test case authoring (using Task recorder), distribution, configuration, execution, investigation, and reporting.

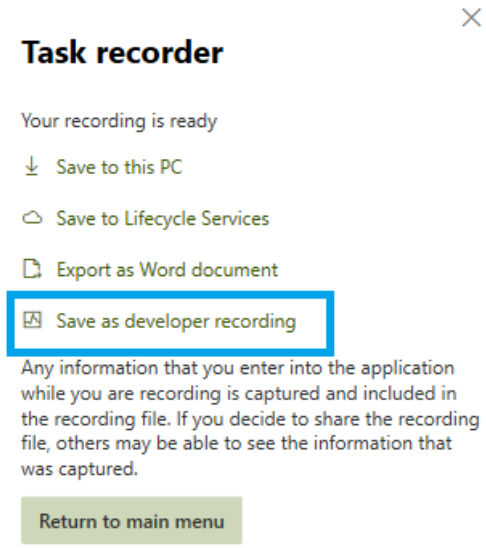


To learn more about this this process, see Create and automate user acceptance tests.

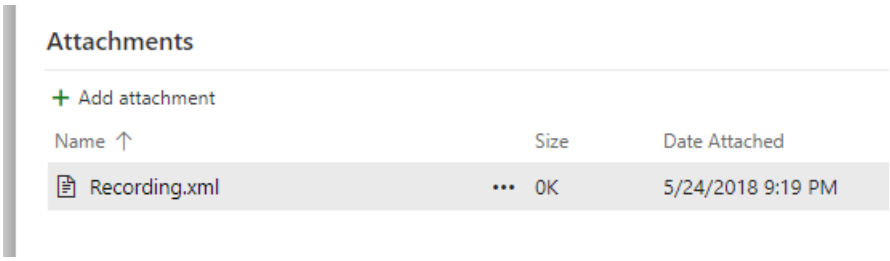
LCS, BPM, and Task Recordings

You aren't required to use the Business process modeler (BPM) tool in LCS. BPM is recommended if you want to enable the management and distribution of test libraries across projects and tenants. These capabilities are especially useful for Microsoft partners and independent software vendors (ISVs). BPM enables the distribution of test libraries as part of LCS solutions.

If you are not using BPM, you can manually create test cases in Azure DevOps and attach developer recording files to your Azure DevOps test cases. You can create developer recording files directly from the Task recorder pane.



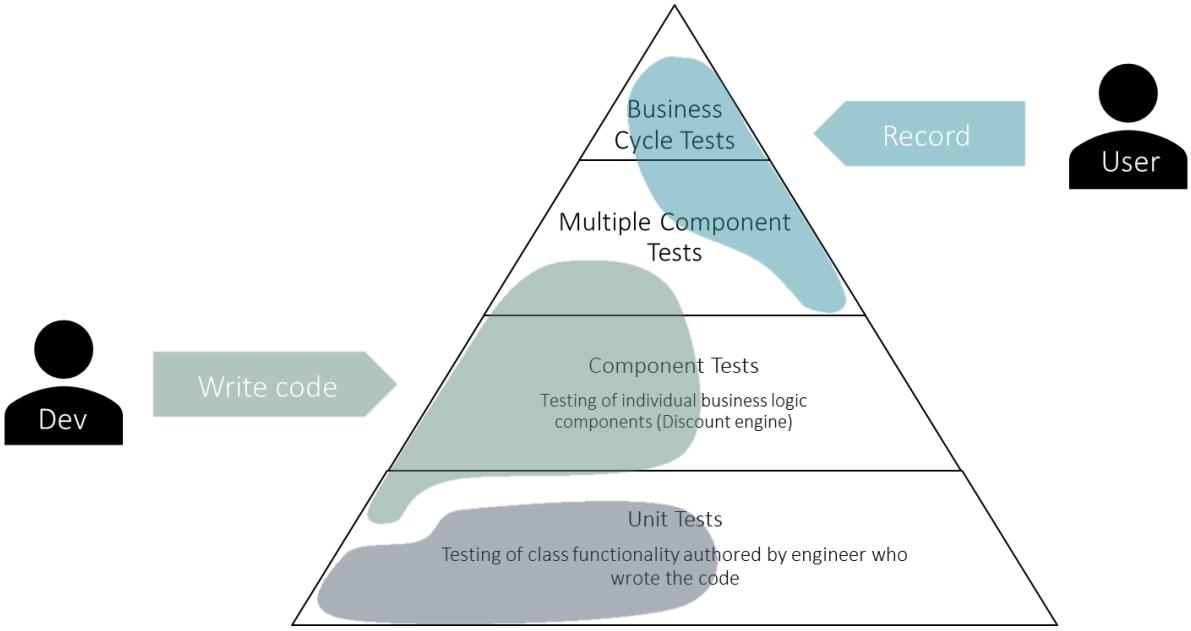
You must name the developer recording file **Recording.xml** before attaching it to the Azure DevOps test case. Alternatively, you can name the recording file **-Test Case Title-.xml**, where **-Test Case Title-** is the DevOps title of the test case.



Intended usage and test classification

Business cycle (business process) testing

The Regression suite automation tool is intended to be used for business cycle tests and scenario tests (multiple component tests) that usually occur at the end of the development lifecycle. This is also referred to as *user acceptance testing*. Business cycle testing consists of a smaller number of test cases than component or unit testing. This is illustrated in the following graphic.



Cloud POS

In addition to testing processes recorded using the Finance and Operations Task recorder, RSAT also supports testing of Cloud POS processes in Dynamics 365 Commerce. For more information about RSAT with Cloud POS, see Test recorder and Regression suite automation tool for Cloud POS.

Warehouse mobile app

You can use RSAT in combination with the Warehouse App Task Validation Framework to automate the testing of warehouse processes. This Tech Talk is a good reference to get started.

Unit and component testing

For unit tests, we do not recommend that you use RSAT. Instead, use the SysTest framework and the build/test automation tools. For component tests, take advantage of the Acceptance test library resources (ATL). ATL is a library of X++ test helpers. When used with the SysTest framework, it offers the following benefits:

- Lets you create consistent test data.
- Increases the readability of test code.
- Provides improved discoverability of the methods that are used to create test data.
- Hides the complexity of setting up prerequisites.
- Supports high performance of test cases.

For more details, see Continuous delivery home page.

Data integration testing

Do not use RSAT for integration tests, instead rely on the data management framework (also known as DIXF). The Data task automation framework enables you to configure and automate the testing of your data integration scenarios.

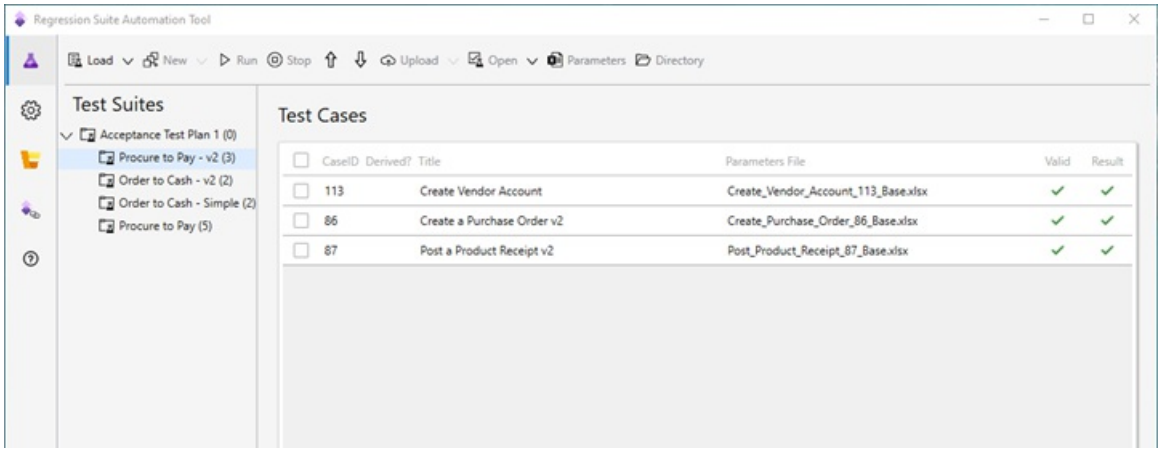
RSAT User interface overview

RSAT 2.1 introduced a modern user interface that simplifies navigation through the main components of the app, including a **Quick links** tab, and quick navigation to DevOps test suites and test runs.

Use the left navigation pane to navigate between the test plan, settings, Cloud POS settings and the quick links page.

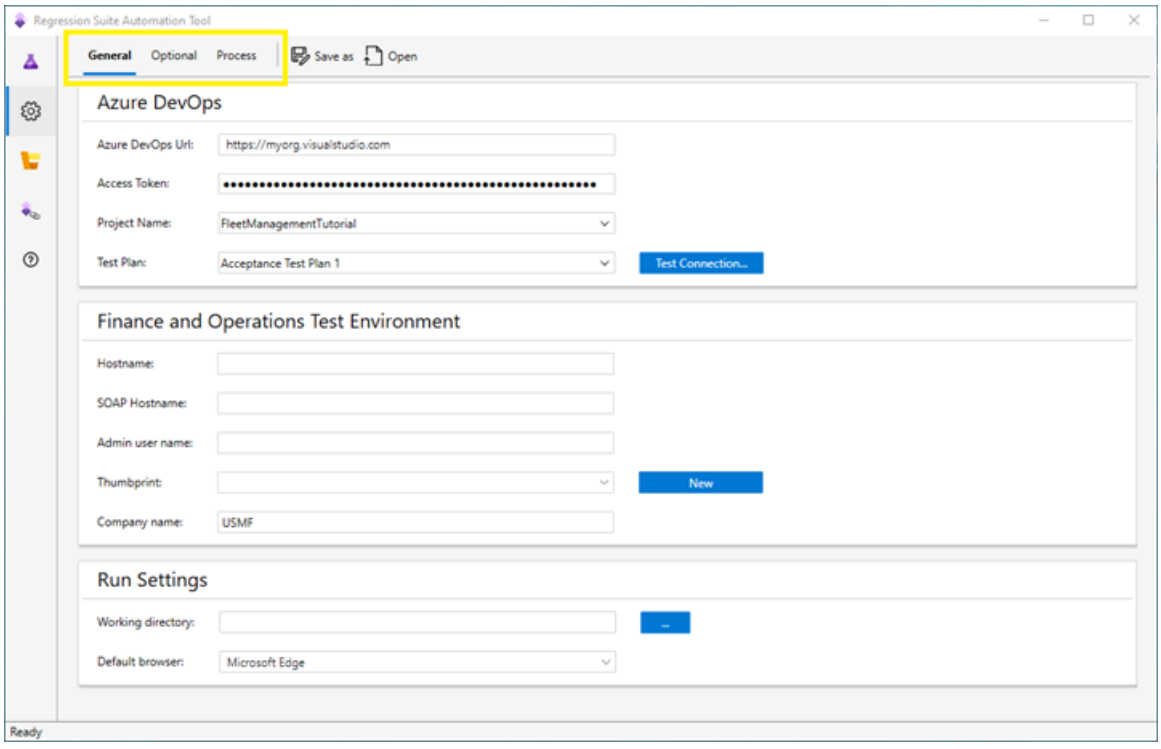
Test Plan

The **Test plan** tab is the main tab that allows you to interact with and execute test cases.



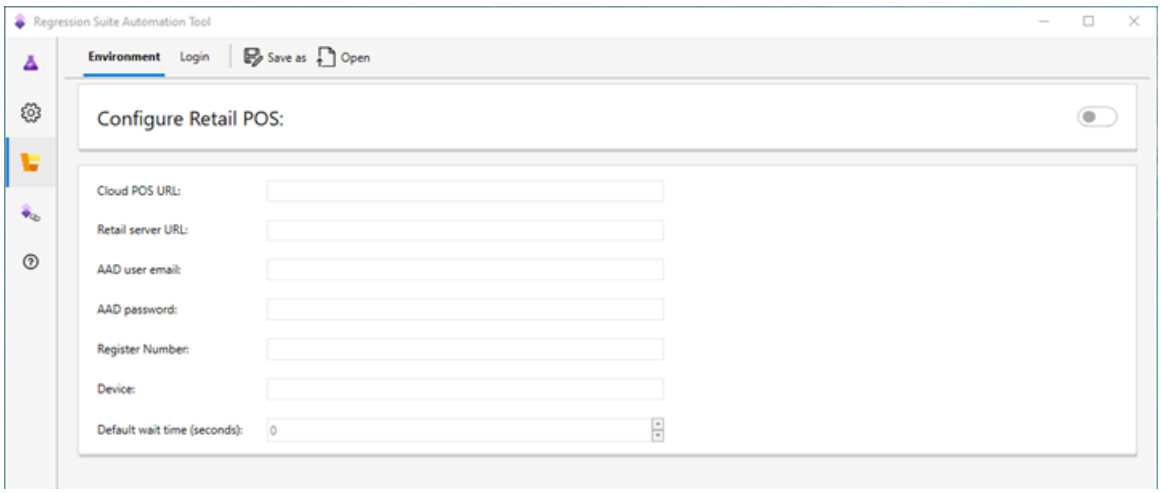
Settings

Select the **Settings** tab to configure RSAT settings. Use the top bar to navigate between general, optional and process settings. You do not need to save your settings, settings are automatically saved as soon as you navigate out of the settings page. You can also save your settings in an RSAT settings file or open an existing settings file.



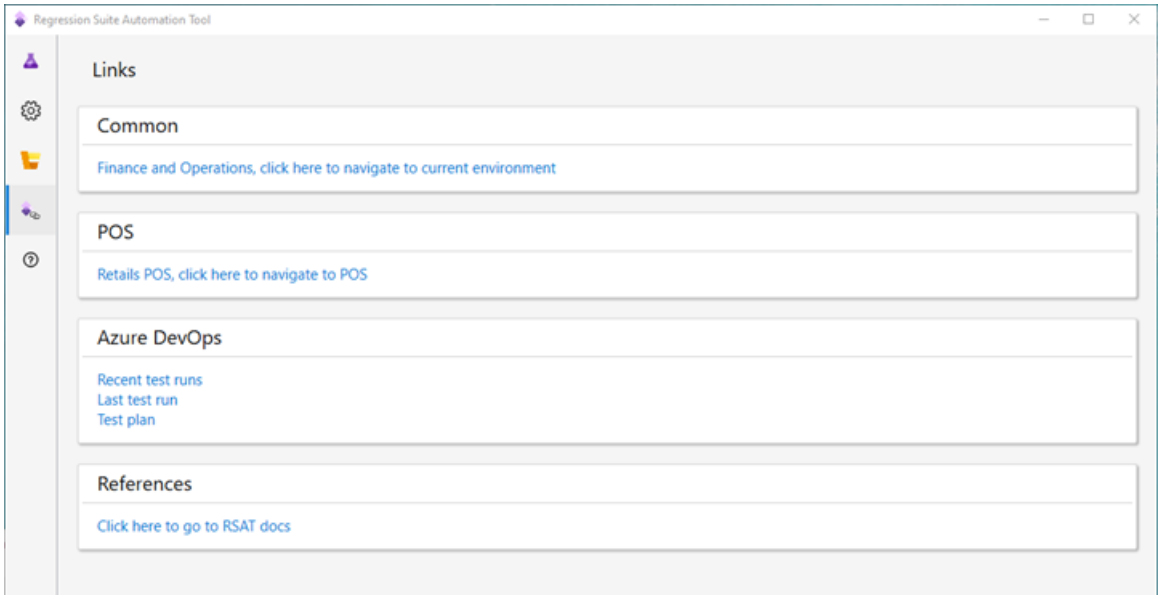
Cloud POS Settings

Select the **Cloud POS Settings** tab to configure RSAT to execute Cloud POS test cases. You do not need to save your settings, settings will automatically be saved as soon as you navigate out of the settings page.



Useful links

The **Links** tab provides new functionality. Select the **Links** tab to quickly navigate to your Finance and Operations environment, Cloud POS, or go to useful Azure DevOps pages showing recent test runs, the last test run, and current test plan. There is also a link to the RSAT docs page.

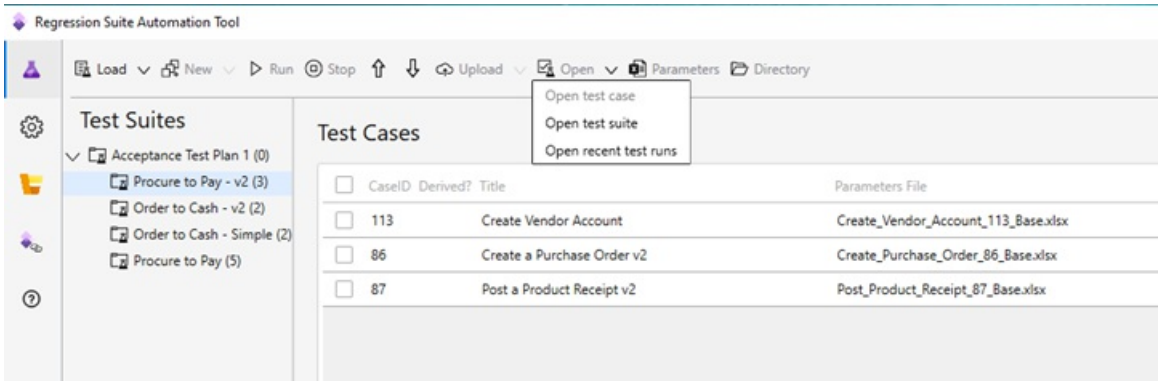


Quick navigation to Azure DevOps

When working with your test plan, the **Open** button now provides 3 options.

- Open the selected test case in Azure DevOps.
- Open the selected test suite.
- Open the recent test runs.

This tab provides quick access to the most relevant pages in Azure DevOps.



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Regression Suite Automation Tool installation and configuration

2/18/2021 • 10 minutes to read • Edit Online

This topic contains information about how to install and configure the Regression suite automation tool (RSAT).

Prerequisites

Test environment (Prerequisite)

Your test environment must be running Platform update 15 or newer. The Regression suite automation tool must have access to your test environment via a web browser.

Excel

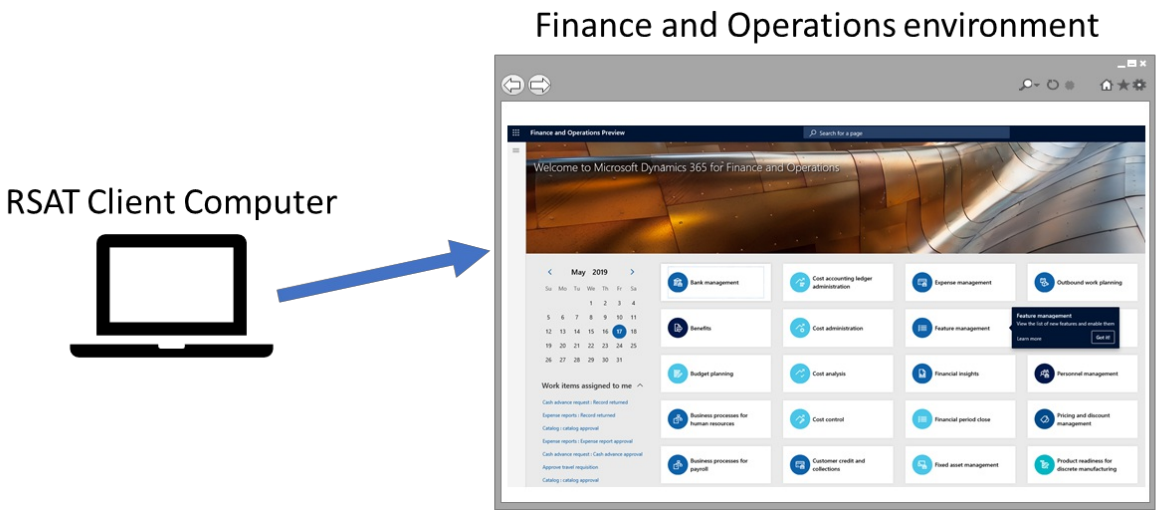
You need Microsoft Excel installed to generate and edit test parameters.

Azure DevOps (Prerequisite)

You must have an Azure DevOps project to store and manage your test cases, test plans, and test case results. You will need an Azure DevOps Test Manager or Test Plans license. For example, if you have a Visual Studio Enterprise subscription, you already have a license to Test Plans. For more information, see Pricing for Azure DevOps Services or Pricing for Azure DevOps Server.

Authentication Certificate

RSAT is designed to be installed on any Windows 10 computer and connect remotely via a web browser to an environment.



To enable secure authentication, RSAT requires a certificate to be installed on the RSAT client computer. The RSAT settings dialog box allows you to automatically create and install the authentication certificate. You will also need to configure the virtual machine (VM) to trust the connection. Follow the instructions in the next sections to install and configure RSAT.

Installation

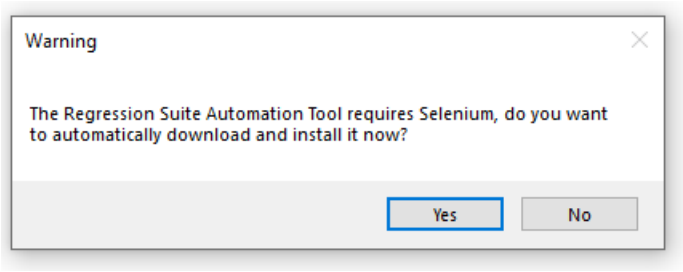
Installer

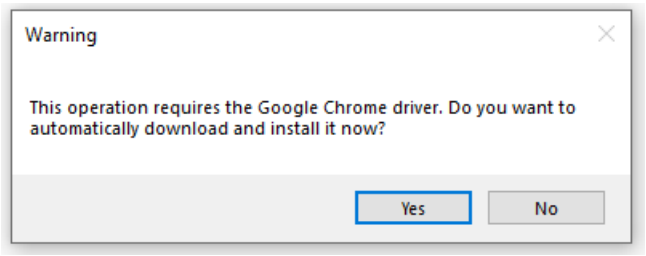
Download the .msi file from the Regression Suite Automation Tool Download to your machine and double-click it to run the installer.

> **NOTE**
>
> If you're using Azure DevOps Server, download and install version 1.210.48249.4 or later.

Selenium and Browser Drivers

RSAT requires Selenium and web browser driver libraries. RSAT will prompt you if needed libraries are missing and will automatically install them for you. Select Yes when you see the following (or similar) messages.



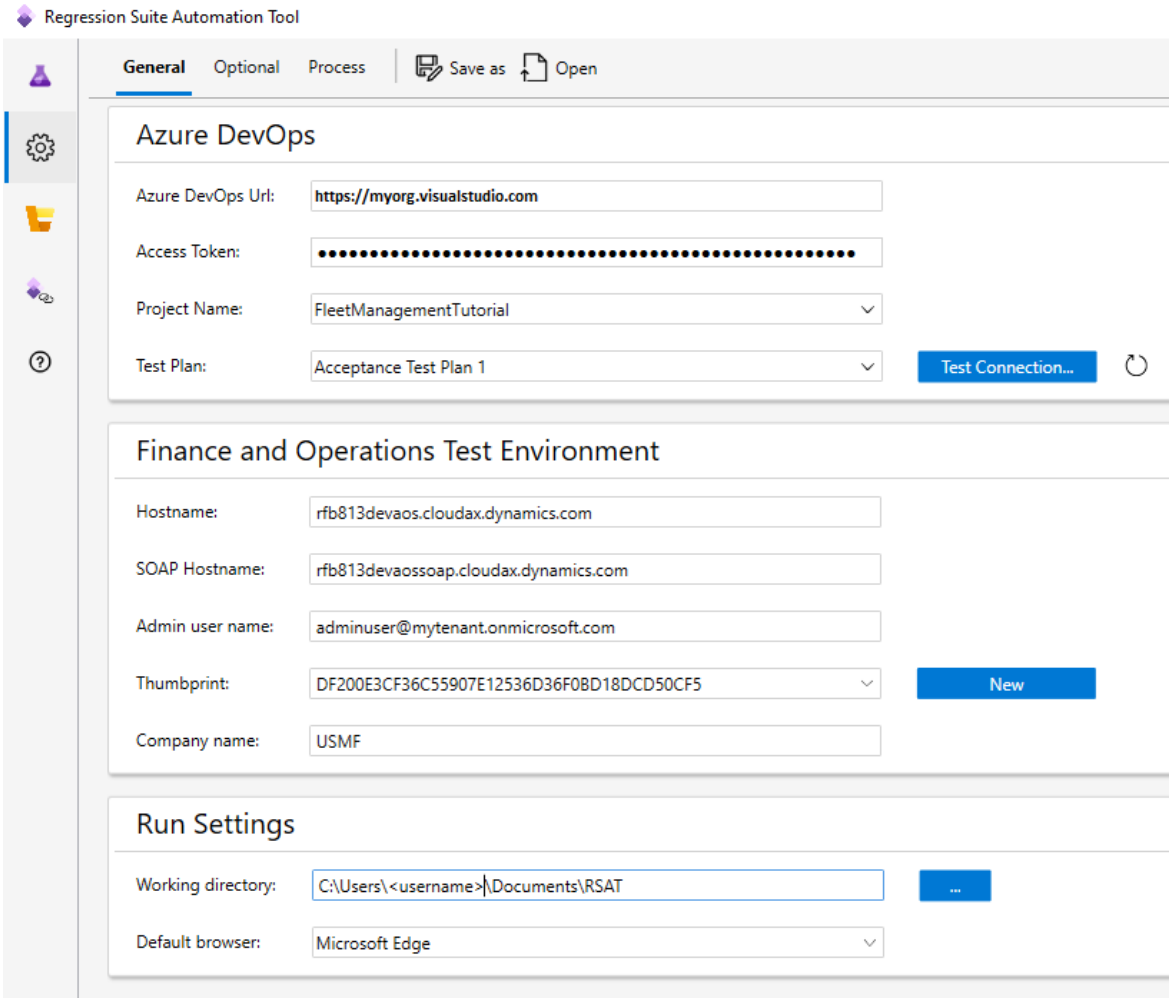


RSAT uses Selenium 3.13.1. The Webdriver library and browser-specific drivers are downloaded to **C:\Program Files (x86)\Regression Suite Automation Tool\Common\External\Selenium**.

Configuration

1. Open RSAT from your desktop.

2. Select the **Settings** tab on the upper left to configure RSAT.



General settings

These settings are required.

Azure DevOps (General settings)

Configure your connection to the Azure DevOps project and test plan.

- **Azure DevOps URL** - This is the URL of your Azure DevOps organization. For example, `https://yourAzureDevOpsUrlHere.visualStudio.com`.

  > **NOTE**
  > If you're using Azure DevOps Server, add **/DefaultCollection** to the end of your Azure DevOps URL.

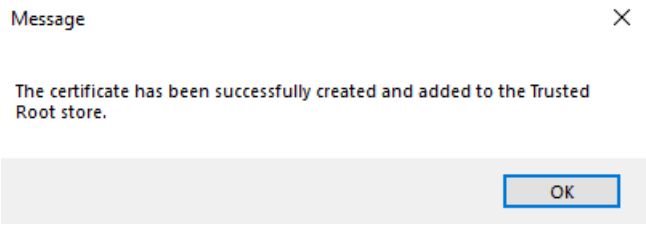
- **Access Token** - The access token that allows the tool to connect to Azure DevOps. You need to create a personal access token or use an existing one that you have saved. For more information, see Authenticate access with personal access tokens.
- **Project Name** - The name of your Azure DevOps project. RSAT will automatically detect project names and test plans available based the Azure DevOps URL specified. You can then select the Test Project and Test Plan.
- **Test Plan** - The Azure DevOps test plan that contains your test cases. For more information, see Create test plans and test suites.

Select **Test Connection** to test your connection to Azure DevOps.

Test environment (General settings)

Configure your connection to the test environment.

- **Hostname** – The hostname of the test environment, such as myhost.cloudax.dynamics.com. Don't include the https:// or http:// prefix.
- **SOAP Hostname** – The SOAP hostname of the test environment.
  - For demo and development environments (also known as one-box environments), add a **soap** suffix to the hostname. For example, if your hostname is `myhost.cloudax.dynamics.com`, use `myhost.soap.cloudax.dynamics.com` as the SOAP hostname.
  - If you don't know the SOAP hostname of your test environment, you can find it in the web.config file for the AOS server in Infrastructure.SoapServicesUrl.
  - If your test environment is a user acceptance testing (UAT) or higher-tier sandbox environment that has no Remote Desktop access, the SOAP hostname is equal to the hostname.
- **Admin User Name** – The email address of an admin user in the test environment. The admin user name must be the email address of a user who belongs to the System Administrator role on the Finance and Operations test environment that RSAT is connecting to. The user account (email address) must also belong to the same tenant as the test environment. For example, if your test environment's default tenant is contoso.com, the admin user must end with @constoso.com.
- **Thumbprint** – The thumbprint of the authentication certificate that you're using. If you don't have Remote Desktop Protocol (RDP) access to your environment, follow the steps lower in this article to download the certificate from Lifecycle Services and paste the thumbprint here. Otherwise, if you do have RDP access to the environment, follow these steps to generate a self-signed certificate.
  1. Select **New** to create and install a new authentication certificate. When prompted, place the .cer file somewhere so you have it saved for your records.
  2. When the process completes, the new certification is installed in the local machine's trusted root store.

  3. The thumbprint of the newly created certificate is automatically inserted on this form. Copy this thumbprint, you will use it in the next section to configure the AOS to trust the connection.
- **Company name** – Specify a company name to use as your default company during creation of Excel parameters files. It can be changed later by editing an Excel file.

Run setting

Configure your local settings.

- **Working directory** - Folder location for storing test automation files, including Excel test data files. For example: **C:\Temp\RegressionTool**.
- **Default browser** - Select the browser to use for test execution. RSAT supports (the new) Microsoft Edge, Microsoft Internet Explorer, and Google Chrome. We recommend Microsoft Edge, which you can download from Introducing the new Microsoft Edge.

Select **Ok** to apply your settings and close the dialog box. Select **Cancel** to cancel your changes and close the dialog. The **Save As** and **Open** buttons allow you to save your settings for reuse later. Select **Save As** to save your current settings into a configuration file on your computer. Select **Open** to restore your settings from a configuration file.

Optional settings

Select the **Optional** tab to configure optional settings.

- **Test Run Prefix** – RSAT reports test run results to Azure DevOps. Test runs are named using the following convention: **<Run ID> <Prefix> <Test Suite>**. Use this setting to set the **<Prefix>**.
- **Test Run Timeout** – The time-out (in minutes) of a test run. All active windows are closed and pending test cases fail when this time-out is reached.
- **Test Action Timeout** – The time-out (in minutes) of individual test steps. When a test step times out, the test case fails.
- **Pause between steps** – The number of seconds to pause between test steps during automated execution of a test case. The default value is **0** (zero). Set this value to force a pause during test execution, for auditing or investigative purposes. You can also specify a pause for an individual test case by changing the **Pause between steps (Seconds)** parameter on the **General** tab of the Excel parameter file for the test case.
- **Fail test on first validation error** – By default, if a test case has multiple validation steps, and there is a validation failure, the test case stops running when the first failure occurs. The test case is then marked as failed. If you want test cases to continue to run until all validations are completed, clear this option. The test case can then evaluate all validations.
- **Fail test on Infolog error** - Check this option to force test cases to fail when an error is encountered in the Finance and Operations Infolog during test case execution.
- **Abort test suite execution on failure** – By default, a test suite run continues even if one of the test cases fails. If you check this setting, the test run is aborted if a test case fails. All the remaining test cases will have a status of **Not Executed**.
- **Enable local file validation rules** - Check this setting to validate whether your test cases are ready for execution. See Validate readiness of test automation files for more details.
- **Enable upload to Azure DevOps** - To prevent accidental upload to Azure DevOps (therefore overriding project-wide recordings and automation files), you can uncheck this setting. This is especially useful when RSAT is deployed on a client machine for execution purposes only, and you want to prevent users from making permanent changes to the test cases.
- **Cloud provider** – Select the provider of the cloud tenant of your test environment. Supported providers are **Global** (Public cloud) and **China** (Sovereign cloud).

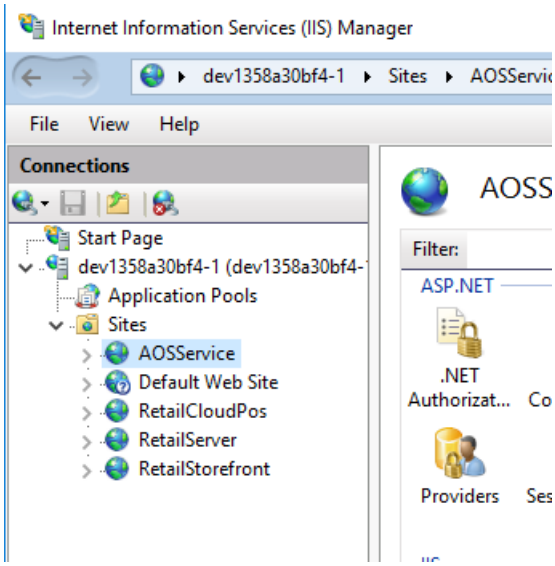
> **IMPORTANT**
> The **Cloud provider** setting is required, and the selected value must be **China** if your Finance and Operations apps were deployed in 21Vianet.

Configure the test environment to trust the connection

If your AOS allows for Remote Desktop connections

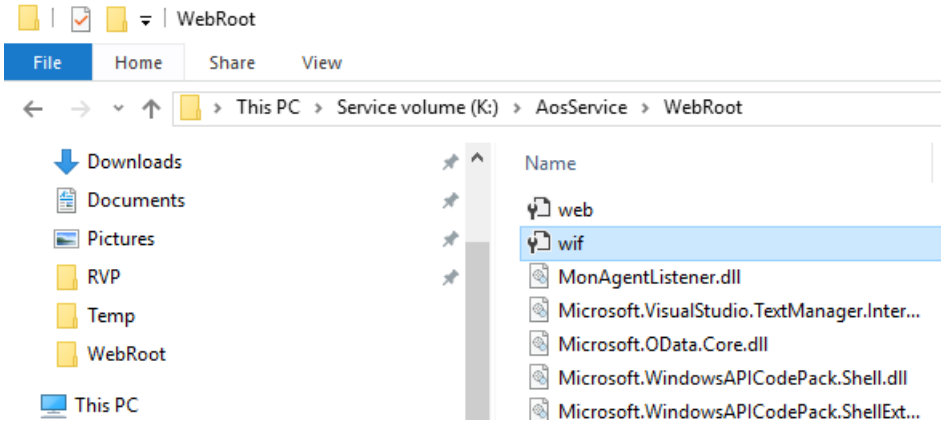
After creating the certificate, configure AOS to trust the test automation connection. On a multi-AOS environment, repeat the following steps for all AOS machines.

1. Open a Remote Desktop connection to the AOS machine.
2. Open IIS and find AOSService in the list of sites.



3. Right-click **AOSService**, then select **Explore**.

4. Open and find the file **wif.config**.



5. Update the **wif.config** file by adding a new authority entry, as shown in the following example. Use **127.0.0.1** for the authority name and paste your certificate thumbprint.

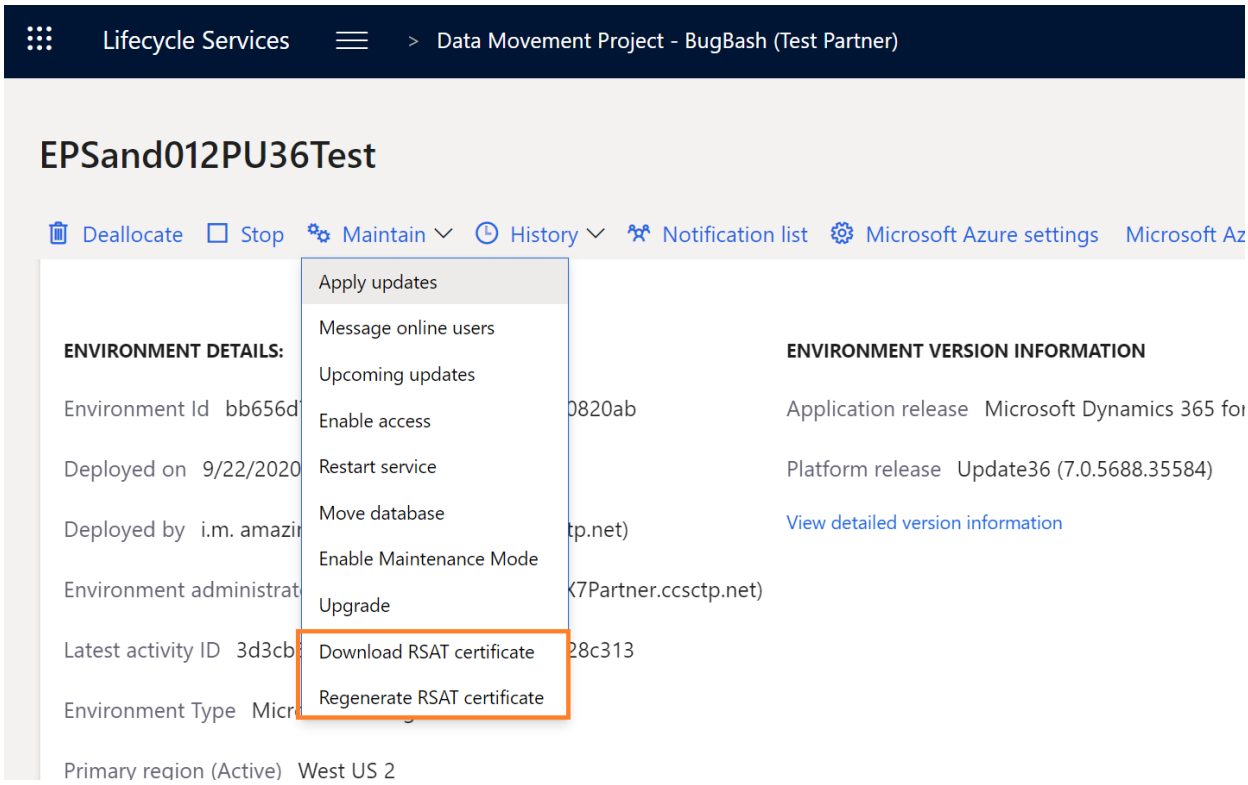
```
<issuerNameRegistry type="Microsoft.Dynamics.AX.Security.SharedUtility.AxIssuerNameRegistry,
Microsoft.Dynamics.AX.Security.SharedUtility">
    <authority name="CN=127.0.0.1">
        <keys>
            <add thumbprint="ccbc124d0a119xxxxxxxxxxxxxxxxxxxx841e797" />
        </keys>
            <validIssuers>
                <add name="CN=127.0.0.1" />
            </validIssuers>
    </authority>
```

If you have no Remote Desktop access to the server

In cases where your Remote Desktop Protocol (RDP) access is removed, such as Microsoft-managed or self-service type sandboxes, Microsoft will generate the certificate for your environment and have it pre-configured. Follow these steps to retrieve the RSAT certificate and use it.

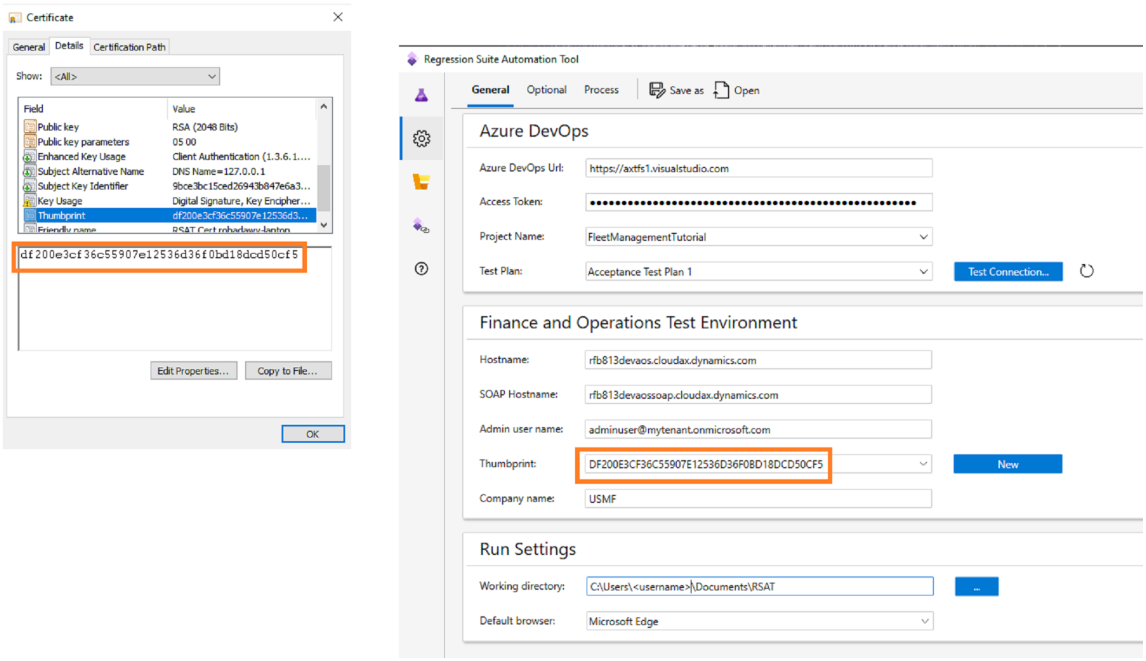
1. Under **Maintain** on your environment details page in Lifecycle Services you'll see two new options.

   - Download RSAT certificate
   - Regenerate RSAT certificate



Use the **Download** button to retrieve the certificate bundle as a .zip file.

2. You'll receive a warning that a clear-text password will be displayed on your screen. You will need the password in subsequent steps. Select **Yes** to continue.
3. Copy the clear-text password for later use. You'll see the .zip file has been downloaded. Inside the .zip file is a certificate (.cer) and a personal information exchange (.pfx) file. Unzip the file.
4. Install the certificate in the local machine's trusted root store:
   - Double-click the certificate (.cer) to open it, and then select **Install Certificate**.
   - Select **local machine**, and then browse to the **Trusted Root Certification Authorities** store to install it in the trusted root store.
5. Install the pfx file in the local machine's personal store:
   - Double-click the personal information exchange (.pfx) file to open it, and select **Local Machine**.
   - Enter the password saved in step 2, and browse to the **Personal** store.
6. Double-click the certificate file to open it. Browse to the **Details** tab, and scroll down until you see the **Thumbprint** section. Select **Thumbprint**, and note the ID in the text box. Select or paste this thumbprint in RSAT settings.



You can now run your tests against the environment using this certificate. The certificate will be autorotated by Microsoft before it expires, at which time you will need to download a new version of this certificate starting from step 1 above. For self-service environments, this will be rotated every 90 days during a downtime window that is closest to the expiry. These downtime windows include customer initiated package deployment, and database movement operations that target the environment.

Manual configuration of authentication certificates

Optionally, you can manually configure the RSAT authentication certificate.

If you are not familiar with this process, get help from your system administrator. Make sure you have Windows Kits installed on your machine. If you do not have Windows Kits installed on your machine, you can download the Windows 10 SDK from Windows 10 SDK. You will need these components for the steps described in this document.

- Windows SDK Signing Tools for Desktop Apps
- Windows SDK for UWP-Managed Apps.

Generate the certificate

You must generate the certificate file on the RSAT client computer. **The certificate must be generated on the same computer that the test tool is running on.** To generate the certificate file, follow these steps:

1. Create the **C:\Temp** folder if it does not already exist on your computer.
2. Open a command-line window as Administrator.
3. Go to the folder where you installed the Windows SDK. Your exact folder may be different, depending on where you have installed the windows SDK). You can also use Windows Kits 8.1.

```
cd c:\Program Files (x86)\Windows Kits\10\bin\10.0.17763.0\x64
```

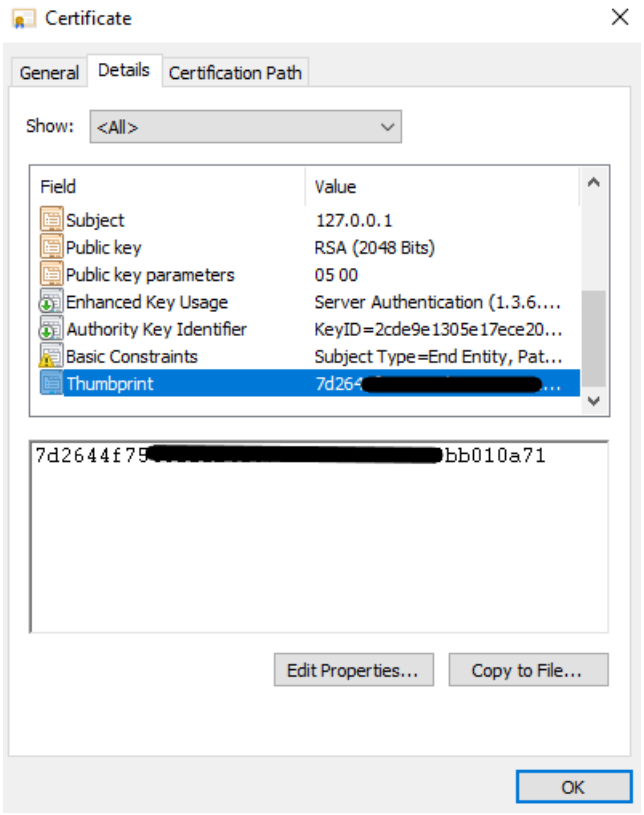
4. Run the following command. When you are prompted to enter a private key password, enter **None**.

```
makecert.exe -n "CN=127.0.0.1" -ss My -sr LocalMachine -a sha256 -len 2048 -cy end -r -eku
1.3.6.1.5.5.7.3.1 c:\temp\authCert.cer
```

Install the certificate to the Trusted Root

To install the certificate, follow these steps:

1. Double-click **authCert.cer** to install the certificate.
2. Select **Install Certificate**.
3. Select **Local Machine** > **Place all certificates in the following Store** > **Browse** > **Trusted Root Certification Authorities** and select **Next** through each screen.
4. Leave the **Password** field blank.
5. In the **Certificate** dialog box, browse to **Details** and look for **Thumbprint**.



6. Copy and save the thumbprint. You will need it to configure the AOS as described earlier in this topic.

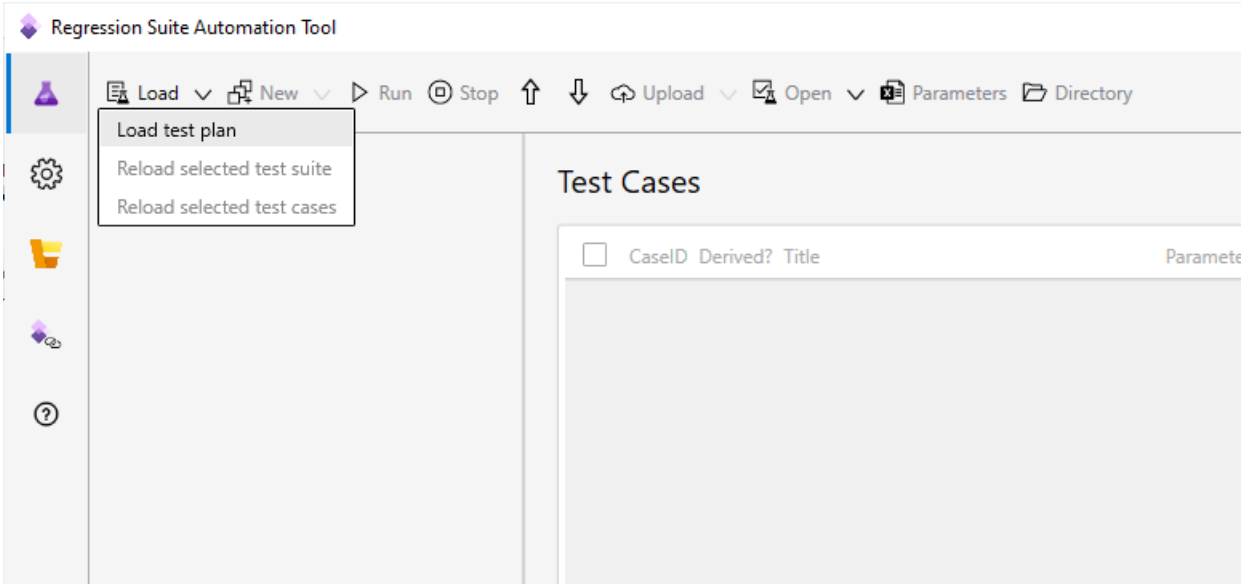
Use the Regression suite automation tool (RSAT)

2/18/2021 • 9 minutes to read • Edit Online

This topic explains how to load test cases from Azure DevOps, generate automation files, modify test parameters, run and investigate results, and save your work back to Azure DevOps.

Load test cases and create automation files

In RSAT, select the **Test Plans** tab and then select **Load** to download test cases and test case automation files. All test cases (and their corresponding attachments) belonging to the test plan specified in the **Settings** tab are downloaded to the local working directory.



Test cases are organized by test suites under a common test plan. These are test suites you created in your Azure DevOps project. Using this tool, you can work with one test suite at a time.

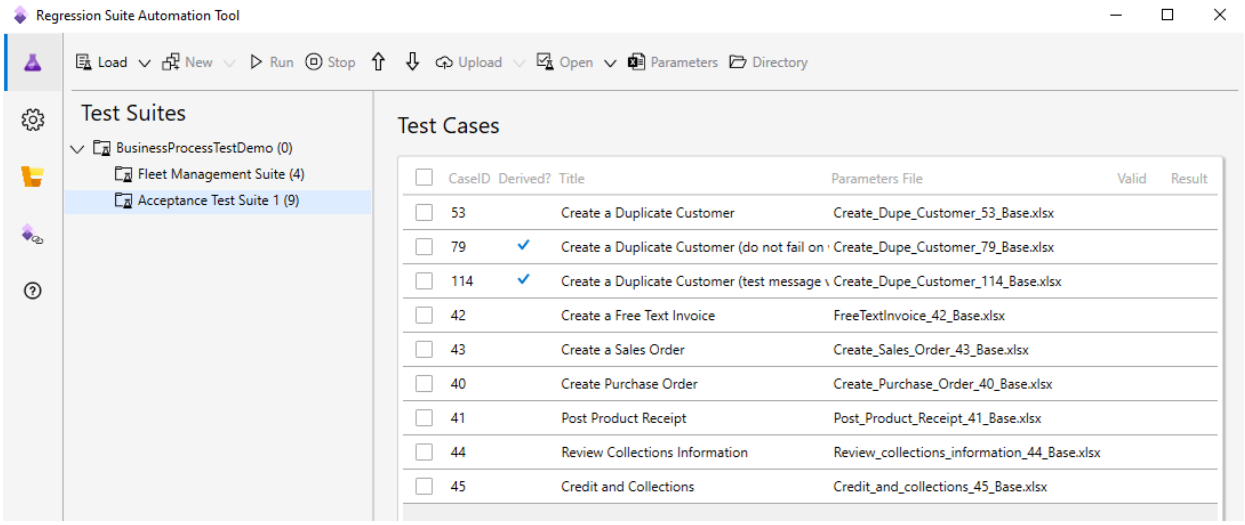
If the tool fails to load any test case, verify that your test plan in Azure DevOps is properly created and contains the desired test suites and test cases.

If this is the first time you are using this test plan, the **Parameters File** column will be blank. You must create test automation files for your test cases.

A test case requires the following attachments for successful execution:

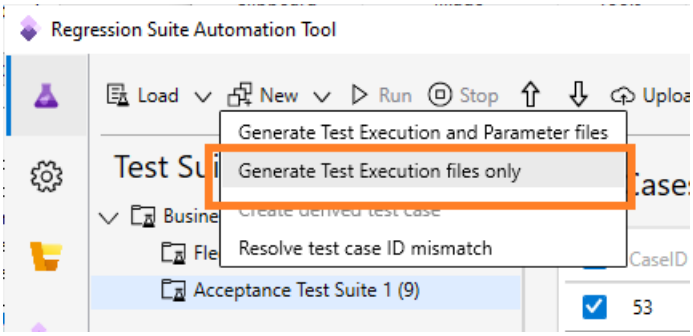
- A **recording file**: This is the recording file created by the Finance and Operations Task recorder. It defines the steps of your test case. It is typically named **recording.xml** or but you can also name it to match the test case title in Azure DevOps. It is attached to the test case in Azure DevOps and downloaded into the **attachment** folder of the local working directory of the test case.
- **Test automation files** consisting of **a test parameter file** (Microsoft Excel file) containing configurable test case parameters and **test execution files**: These files are generated by RSAT to enable automated execution of the test recording. Filenames are suffixed by **_Base.cs**, **_Base.xml**, and **_Base.dll**.

When you select **New**, test automation files are generated in your working directory. The Excel test parameter files will appear on the grid under **Parameters File**.



You can also generate **test execution files** only, without overwriting your parameter files. Select **New** > **Generate Execution Files** to regenerate only execution files and leave Excel files unaffected.

You must generate test execution files when you install a new version of the tool, and when you modify or load a new version of the recording file. In this way, you update your execution files but also preserve the test parameter files.

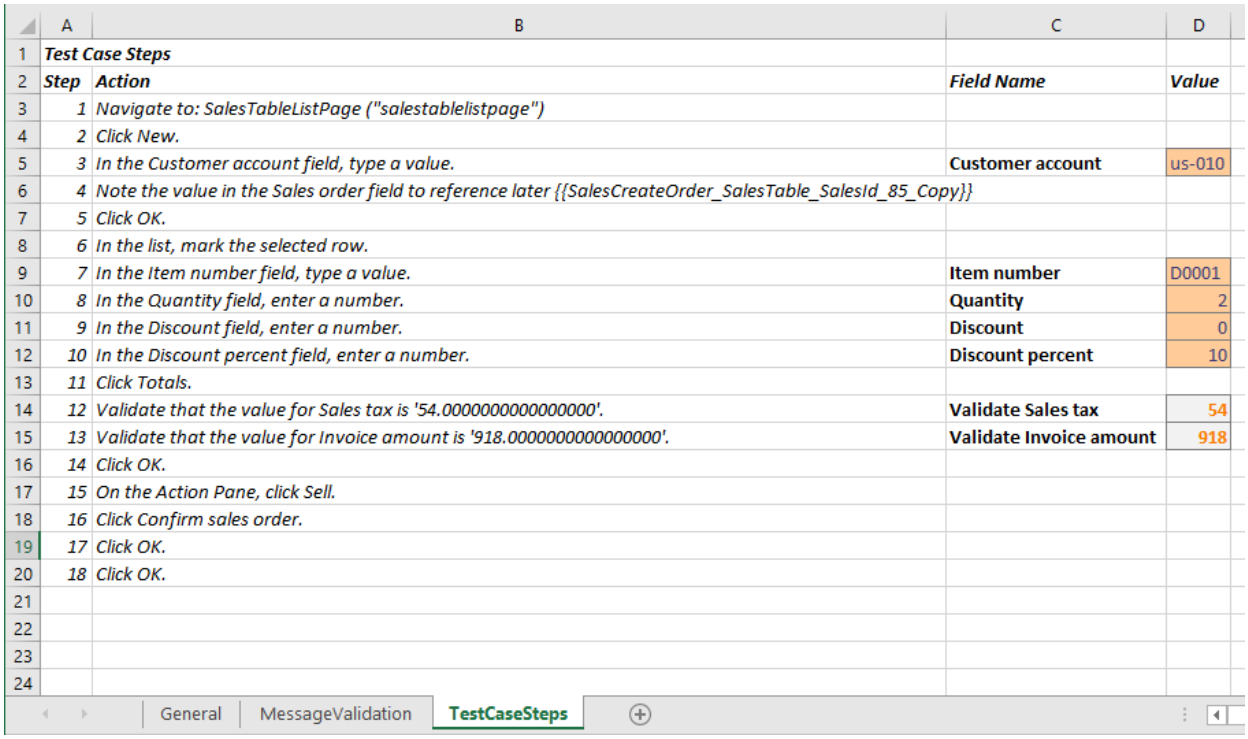


Modify test parameters

This section describes how to modify Excel files to specify input and validation parameters for your test run. Select one or more test cases to modify, and then select the **Parameters** button (Microsoft Excel symbol) on the toolbar. An Excel window opens for each test case that you selected. Alternatively, you can open the Excel files directly from the working directory.

In addition to the **General** tab, the Excel parameter file contains a **MessageValidation** tab and a **TestCaseSteps** tab.

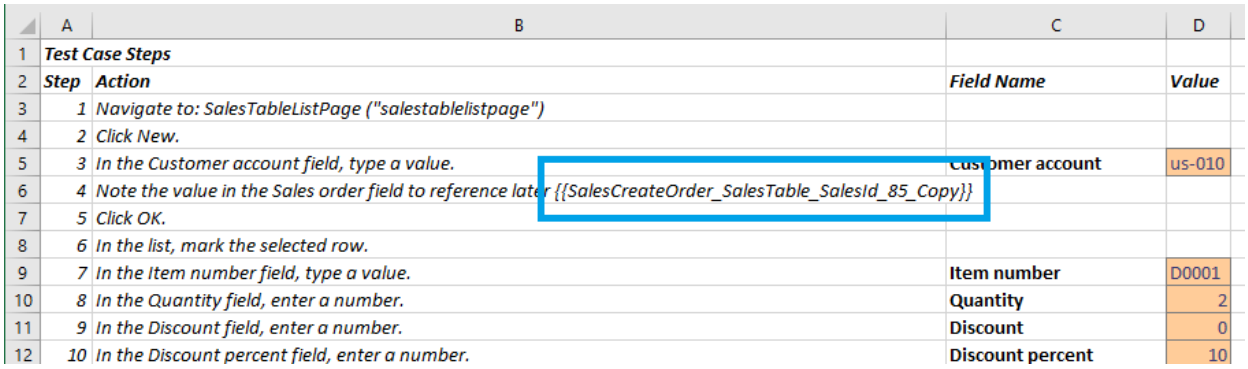
Select the **TestCaseSteps** tab to configure input and validation parameters of your test case. Input and validation parameters are placed directly next to their corresponding test case step, enabling test authors with context and a simple experience. When you modify parameters, it is clear what steps of the test case you are affecting. You can enter values or formulas in context. Color coding differentiates input parameters from validation steps.



	A	B	C	D
1	***Test Case Steps***			
2	***Step***	***Action***	***Field Name***	***Value***
3	*1*	*Navigate to: SalesTableListPage ("salestablelistpage")*		
4	*2*	*Click New.*		
5	*3*	*In the Customer account field, type a value.*	**Customer account**	us-010
6	*4*	*Note the value in the Sales order field to reference later {{SalesCreateOrder_SalesTable_SalesId_85_Copy}}*		
7	*5*	*Click OK.*		
8	*6*	*In the list, mark the selected row.*		
9	*7*	*In the Item number field, type a value.*	**Item number**	D0001
10	*8*	*In the Quantity field, enter a number.*	**Quantity**	2
11	*9*	*In the Discount field, enter a number.*	**Discount**	0
12	*10*	*In the Discount percent field, enter a number.*	**Discount percent**	10
13	*11*	*Click Totals.*		
14	*12*	*Validate that the value for Sales tax is '54.0000000000000000'.*	**Validate Sales tax**	54
15	*13*	*Validate that the value for Invoice amount is '918.0000000000000000'.*	**Validate Invoice amount**	918
16	*14*	*Click OK.*		
17	*15*	*On the Action Pane, click Sell.*		
18	*16*	*Click Confirm sales order.*		
19	*17*	*Click OK.*		
20	*18*	*Click OK.*		
21				
22				
23				
24				

General | MessageValidation | **TestCaseSteps**

Reusable variables that are copied while recording the test case are also shown in context of the test case step. You can easily locate a variable and copy it to use in subsequent steps and formulas. For more information, see Copy variables to chain test cases.



	A	B	C	D
1	***Test Case Steps***			
2	***Step***	***Action***	***Field Name***	***Value***
3	*1*	*Navigate to: SalesTableListPage ("salestablelistpage")*		
4	*2*	*Click New.*		
5	*3*	*In the Customer account field, type a value.*	**Customer account**	us-010
6	*4*	*Note the value in the Sales order field to reference later {{SalesCreateOrder_SalesTable_SalesId_85_Copy}}*		
7	*5*	*Click OK.*		
8	*6*	*In the list, mark the selected row.*		
9	*7*	*In the Item number field, type a value.*	**Item number**	D0001
10	*8*	*In the Quantity field, enter a number.*	**Quantity**	2
11	*9*	*In the Discount field, enter a number.*	**Discount**	0
12	*10*	*In the Discount percent field, enter a number.*	**Discount percent**	10

Save the Excel files when you are done making edits.

Run a test as a specific user

By default, tests are executed using the admin role. If you want to run the test as a specific security role, specify the email address of a user under the **Test User** parameter in the **General** tab of the Excel parameter file. The **Test User** must be a valid user of the environments you are connecting to. The test will run under the security roles that the specific user belongs to. You need version 1.200 or newer for this feature to be functional.

	A	B
1	General	
2	Field Name	Value
3	Test Case ID	1422
4	Recording Name	Update_user
5	Playback Order	0
6	Company	TEST
7	Test User	alicia@contoso.com
8	Pause between test cases (Seconds)	
9	Action Timeout (Seconds)	
10	Abort test suite execution on failure	FALSE
11	Fail on warning message in the Infolog	FALSE
12	Pause between steps (Seconds)	

Run a test in the context of a specific company

The **General** tab of the Excel parameter file also allows you to specify the name of a legal entity (Company). The test will run in the context of this company. You can specify your default company in the **Settings** dialog box of the tool.

Pause after a specific test step

You can insert a pause between specific test steps. Navigate to the **TestCaseSteps** tab of the Excel parameters file and insert a value (in seconds) in the pause column of a test step. This will pause test cases execution after the test step is completed.

	A	B	C	D	E
1	*Test Case Steps*				
2	*Step*	*Action*	*Field*	*Value*	*Pause*
3	1	*Navigate to: CustTable ("custtablelistpage")*			
4	2	*Click New.*			
5	3	*In the Customer account field, enter or select a value.*	**Customer account**	Test-001	10
6	4	*In the Name field, type a value.*	**Name**	RFB	
7	5	*In the Customer group field, enter or select a value.*	**Customer group**	10	
8	6	*In the Country/region field, type a value.*	**Country/region**	usa	
9	7	*Click Save.*			
10					

If you don't see the **Pause** column, you are using an older version of the Excel parameters file and need to regenerate it. Select the desired test case, then go to **New > Generate Test Execution and Parameter Files**. This may override edits you have made to the parameters file, so you should back up the existing Excel file first.

Other notable test case execution settings

You may find the following settings useful. They are available on the **General** tab of the Excel parameter file.

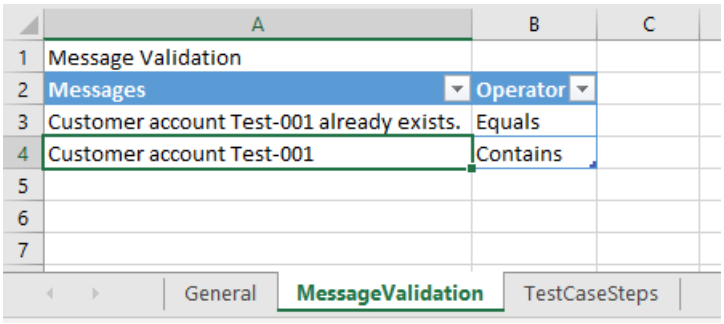
- **Fail on warning message in the Infolog**: By default, test cases fail when an error occurs or a validation step fails. If you also want a test case to fail in response to a warning message, set the **Fail on warning message in the Infolog** option to **True** . This is useful, for example, if a test case adds a duplicate customer record. The default setting is **False**.
- **Abort test suite execution on failure**: If you set the **Abort test suite execution on failure** option to **True**, execution of the test suite is aborted if the test case fails. All the remaining test cases will have a status of **Not Executed**. The default setting is **False**.
- **Pause between steps**: The number of seconds to pause between test steps. This will affect every test step. The default value is **0** (zero).

Infolog and message validation

Excel parameter files that are generated using version 1.200 or newer contain a **MessageValidation** tab.

You can enter messages in this tab under **Message Validation**. After a test case completes execution, it validates that the messages specified here appear in the Infolog. The test case will fail if these messages are not found.

You can specify any expected messages including error messages. Any message specified in this section will cause a test case to fail unless it is found in the Infolog during execution. Two operators are available: **Equals** and **Contains**. If you use **Equals**, then RSAT performs a string comparison with all messages in the Infolog and fails validation if the full message is not found. If you use **Contains**, then RSAT will validate that at least one message in the Infolog contains the string you specify.



You can configure whether string comparison is case sensitive or not in the **Optional** page of the **Settings** tab.

Run

Select **Run** to execute the selected test cases. Only test cases with existing automation files can be run. The tool will open and execute these tests with the data you entered in Excel.

You can modify the order in which test cases are executed using the up and down arrow buttons.

Pause prior to a test case run

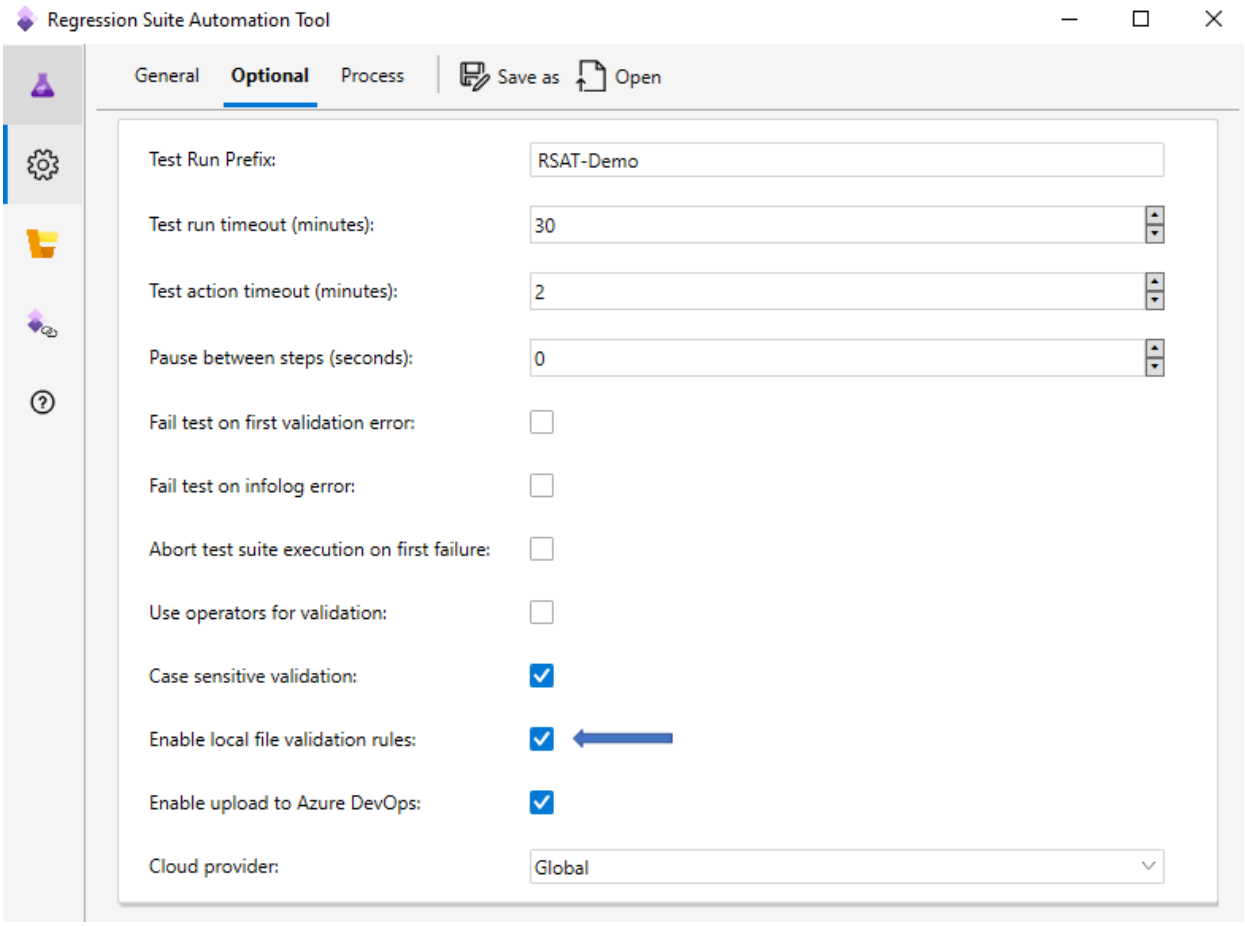
You can add a pause before a test case starts execution. If you want to pause, update the cell **Pause (seconds)** on the **General** tab of the Excel parameters file of the desired test case.

Stop a run

When a test run is in progress, you can select the **Stop** button on the toolbar to cancel the run. Execution stops after the currently running test case completes. The remaining test cases will be marked as **Not Executed** in Azure DevOps.

Validate readiness of test automation files

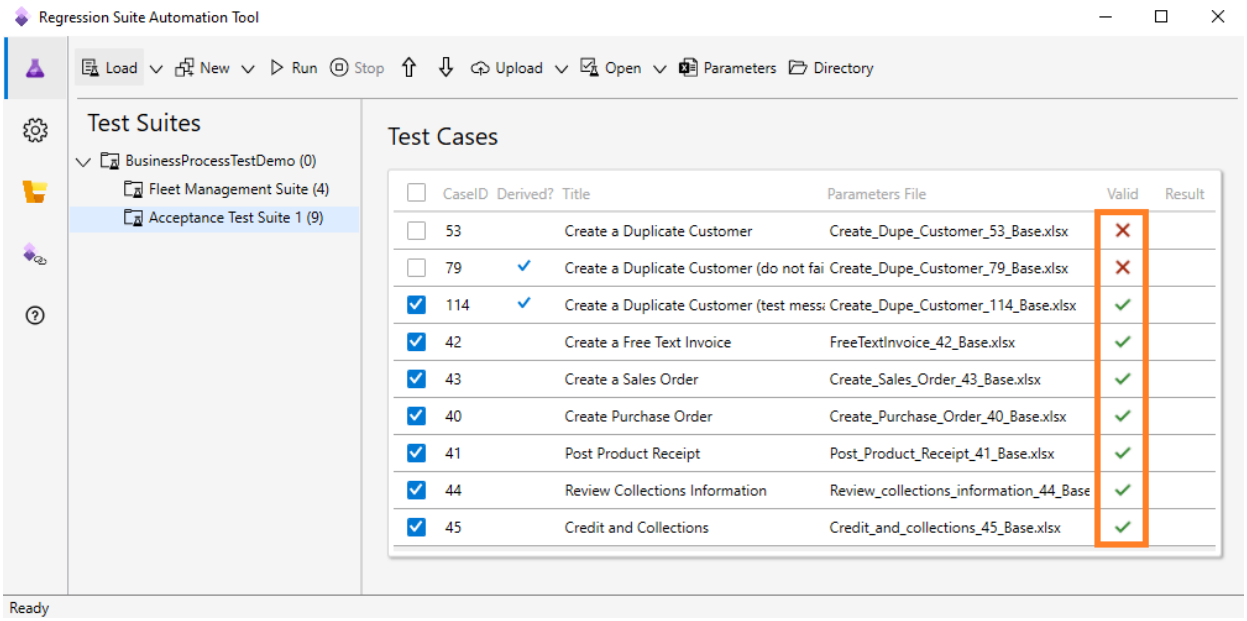
Optionally, you can turn on a setting that validates whether your test cases are ready for execution. This setting prevents unknown errors related to the validity of recordings and test automation files. This option is available as of RSAT version 1.210. You can enable this by selecting the **Settings** tab and then selecting the **Optional** tab.



When enabled, a background process continuously validates the following for each test case.

- The local working directory exists.
- The Excel parameter file exists.
- Test automation files (binary and Xml files) needed for execution exist.
- Test automation files are compatible with current version of RSAT. You must regenerate test automation files when you install a new version of RSAT.
- Test case ID specified in the Excel parameter file matches the test cases ID in Azure DevOps.

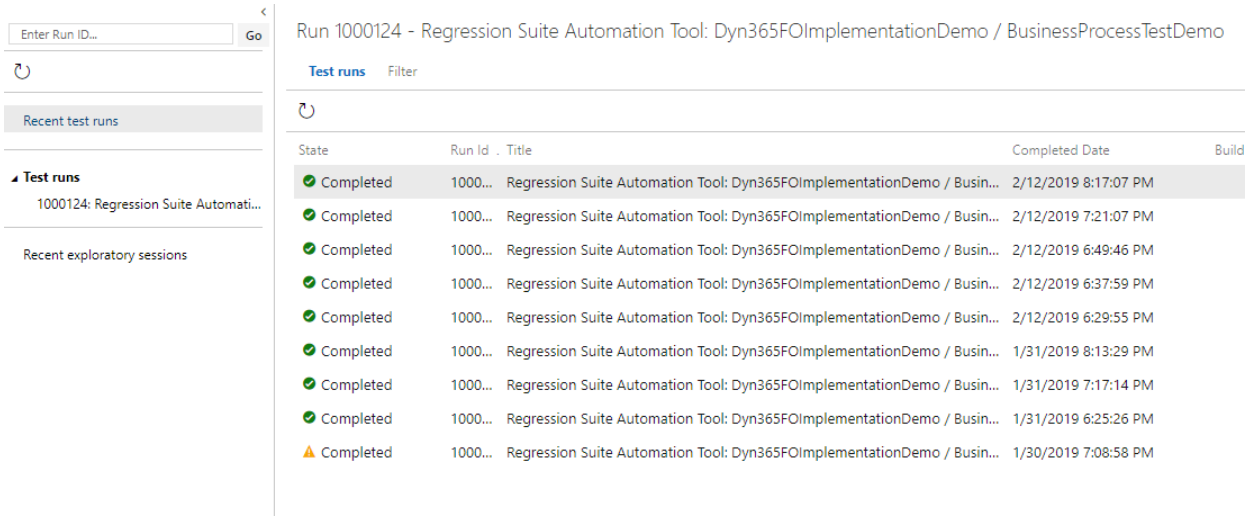
The Valid column in the grid indicates the result of the validation process. If validation fails, click on the **X** in the **Valid** column to view the error and recommended action.



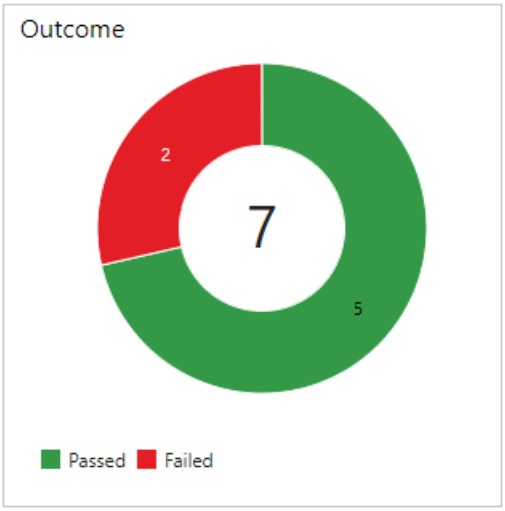
Investigate results

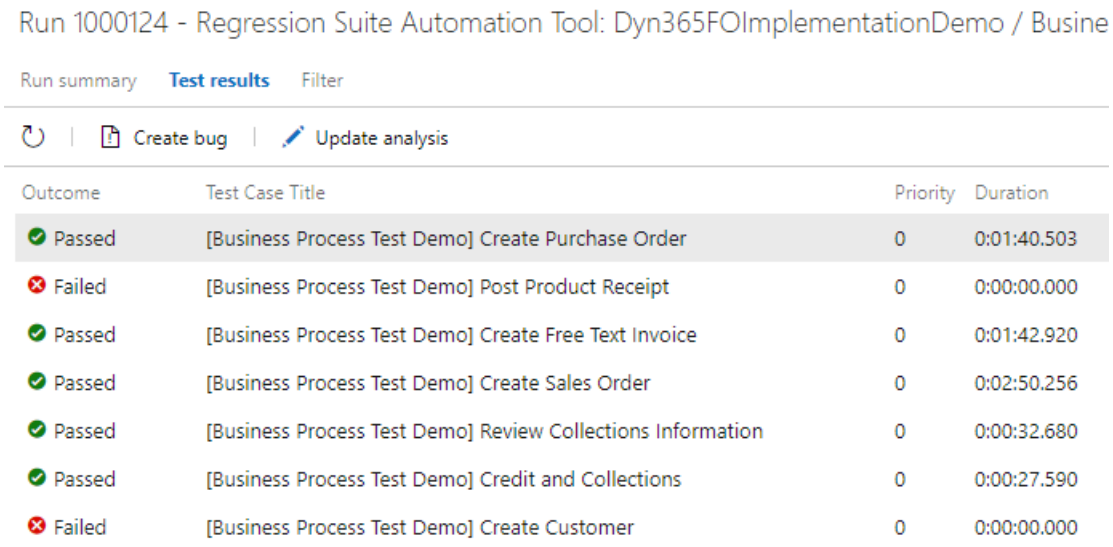
When all test cases complete execution, **Pass** or **Fail** will be populated in the **Result** column. You can click on the result to see error messages.

Additional investigation details are available in Azure DevOps. To view this information, from your Azure DevOps project page, go to **Test > Runs**.

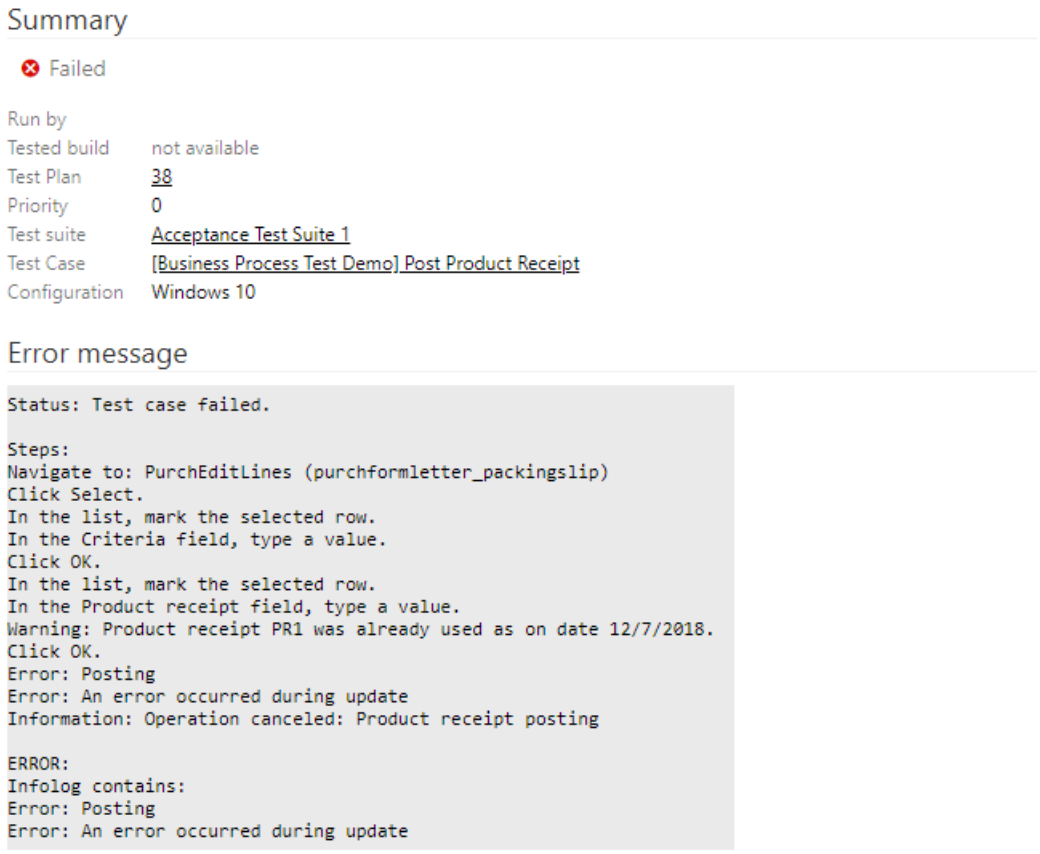


Select the desired test run. It will include the results of all tests that were executed during that run.





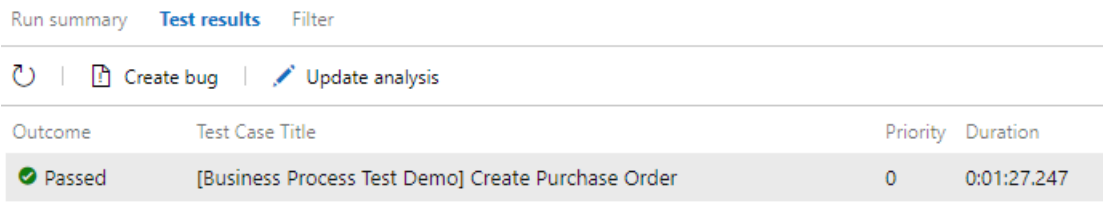
You can open a failed test result and review the **ErrorMessage** section for information about the failure.



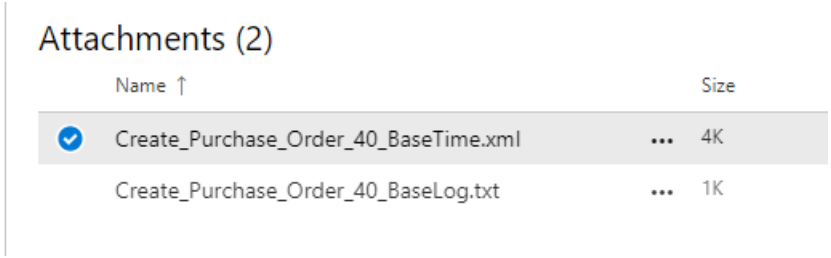
All error messages are also available locally under **C:\Users$YourUserName\AppData\Roaming\regressionTool\errormsg-.txt**.

Test response times

In addition to execution logs, the duration of a test case is also available in the test result.



You can also review the response time of each step of the test case by opening the **BaseTime.xml** file attached to the test result.



You need version 1.200 or newer for response times to be available.

Upload to Azure DevOps to commit your work

To commit your work to Azure DevOps, select **Upload**. This uploads recordings and test automation files, including Excel test parameter files, of all selected test cases to Azure DevOps for future use. After test

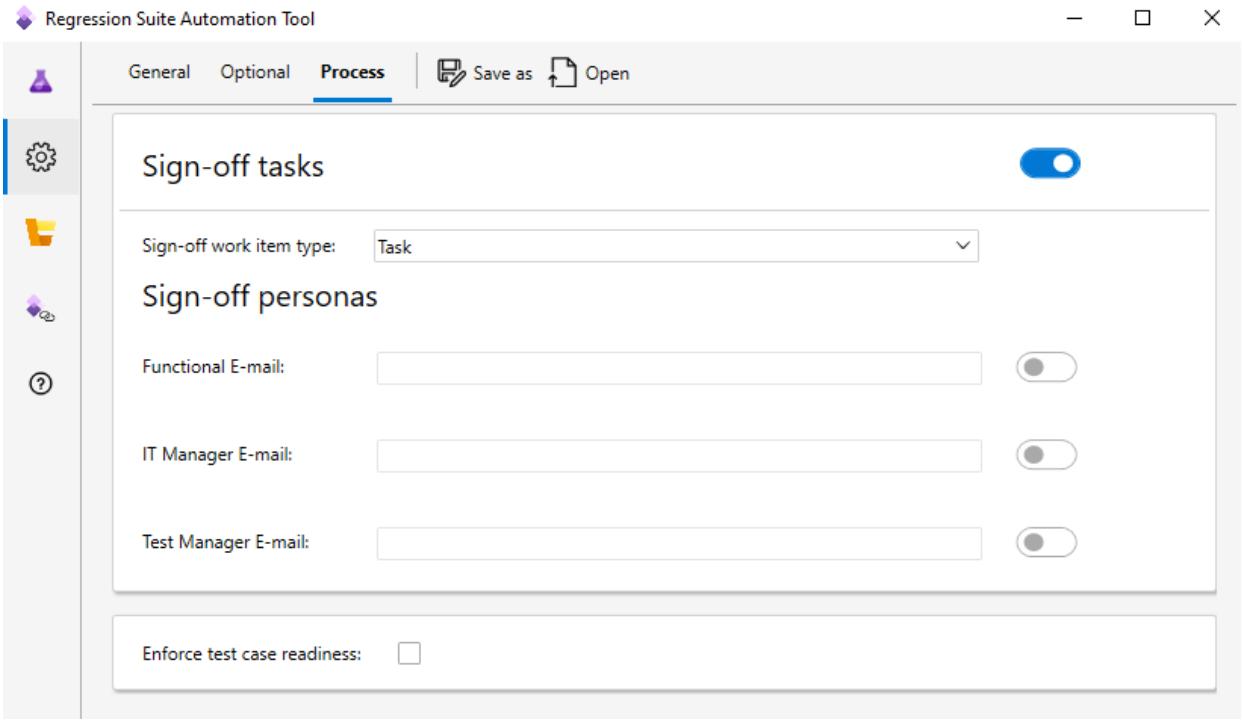
automation files are uploaded to Azure DevOps, the next time you use the Regression suite automation tool, even from a different computer, you can simply use **Load** and then **Run**, without generating test execution files or editing Excel parameter files.

In the upload menu, you also have the option to upload recording files (Task recordings) only.

If you are unsure what test cases to select, and you want to commit all changes (since last load) to Azure DevOps, select **Upload all modified automation files** in the upload menu.

Process compliance

RSAT provides capabilities for managing the readiness of test cases. It also provides a sign-off process for test runs. This is configurable in the **Process** tab under Settings.



Enforce test case readiness

You can set up the test case so that it isn't run unless it has a status of **Ready** in Azure DevOps. Select the **Enforce test case readiness** check box. By default, the check box is cleared.

Signoffs

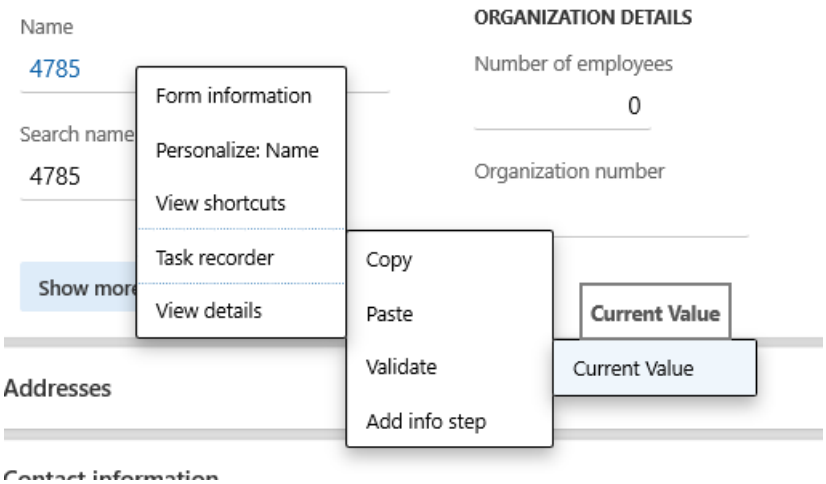
When your test run is complete, RSAT can create sign-off work items in Azure DevOps. Select the **Sign-off tasks** check box. Then set the type of work item that should be created for each person who signs off. You can select the **Functional**, **IT Manager**, or **Team Manager** role for sign-offs, and then specify appropriate email addresses. Work items will then be created in Azure DevOps and assigned to owners for approval.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Validate expected values

2/18/2021 • 2 minutes to read • Edit Online

An important component of a test case is validation of expected values. You can define validation parameters during the authoring of your test cases using Task Recorder. While recording, right-click on a control and select **CurrentValue** under the **Task Recorder > Validate** menu. This action becomes a validation step that you can use with the Regression suite automation tool. The control value will become a validation variable in the automatically generated Excel parameters file. The menu item is shown in the following image.



For more information about how to create task recordings, see Task recorder resources.

When RSAT generates the Excel parameter file for a test case, validation steps are added as shown in the image below. You can enter the expected value to use during execution of the test case.

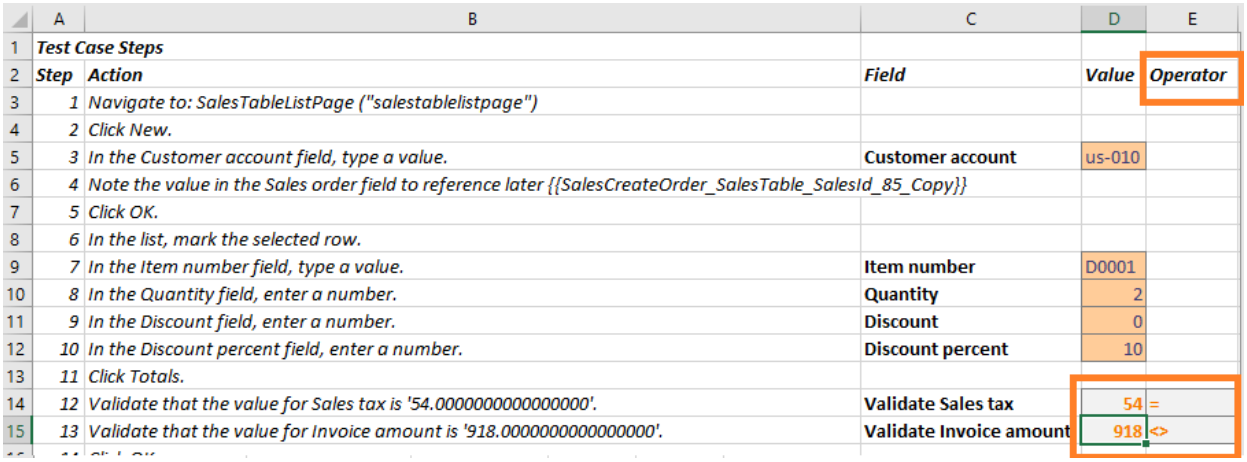
	A	B	C	D
1	***Test Case Steps***			
2	***Step***	***Action***	***Field Name***	***Value***
3	*1*	*Navigate to: SalesTableListPage ("salestablelistpage")*		
4	*2*	*Click New.*		
5	*3*	*In the Customer account field, type a value.*	**Customer account**	us-010
6	*4*	*Note the value in the Sales order field to reference later {{SalesCreateOrder_SalesTable_SalesId_85_Copy}}*		
7	*5*	*Click OK.*		
8	*6*	*In the list, mark the selected row.*		
9	*7*	*In the Item number field, type a value.*	**Item number**	D0001
10	*8*	*In the Quantity field, enter a number.*	**Quantity**	2
11	*9*	*In the Discount field, enter a number.*	**Discount**	0
12	*10*	*In the Discount percent field, enter a number.*	**Discount percent**	10
13	*11*	*Click Totals.*		
14	*12*	*Validate that the value for Sales tax is '54.0000000000000000'.*	**Validate Sales tax**	54
15	*13*	*Validate that the value for Invoice amount is '918.0000000000000000'.*	**Validate Invoice amount**	918
16	*14*	*Click OK.*		
17	*15*	*On the Action Pane, click Sell.*		
18	*16*	*Click Confirm sales order.*		
19	*17*	*Click OK.*		
20	*18*	*Click OK.*		

General | MessageValidation | TestCaseSteps

Validate expected values using operators

You can also use operators in validation steps to validate that a variable is not equal, less than, or greater than a specified value. To use this feature, open the **Settings** tab and select the **Optional** tab. Turn on the setting named **Use operators for validation**. This option is available as of RSAT version 1.210. If you have been using

an older version of the tool, you must regenerate new Excel parameter files to take advantage of this functionality. In the Excel file, a new **Operator** field will appear, as shown in the following image.



	A	B	C	D	E
1	*Test Case Steps*				
2	*Step*	*Action*	*Field*	*Value*	*Operator*
3	1	*Navigate to: SalesTableListPage ("salestablelistpage")*			
4	2	*Click New.*			
5	3	*In the Customer account field, type a value.*	**Customer account**	us-010	
6	4	*Note the value in the Sales order field to reference later {{SalesCreateOrder_SalesTable_SalesId_85_Copy}}*			
7	5	*Click OK.*			
8	6	*In the list, mark the selected row.*			
9	7	*In the Item number field, type a value.*	**Item number**	D0001	
10	8	*In the Quantity field, enter a number.*	**Quantity**	2	
11	9	*In the Discount field, enter a number.*	**Discount**	0	
12	10	*In the Discount percent field, enter a number.*	**Discount percent**	10	
13	11	*Click Totals.*			
14	12	*Validate that the value for Sales tax is '54.0000000000000000'.*	**Validate Sales tax**	54	=
15	13	*Validate that the value for Invoice amount is '918.0000000000000000'.*	**Validate Invoice amount**	918	<>

Validate the state of a control

When recording test cases, Task Recorder supports additional validation action:

- Validate whether a control is enabled or disabled.
- Validate whether a control is editable or read-only.

To take advantage of this validation, you need to be use a Finance and Operations app running on 10.0.13 (or newer) and RSAT 2.0 (or newer). For more information, see Validate.

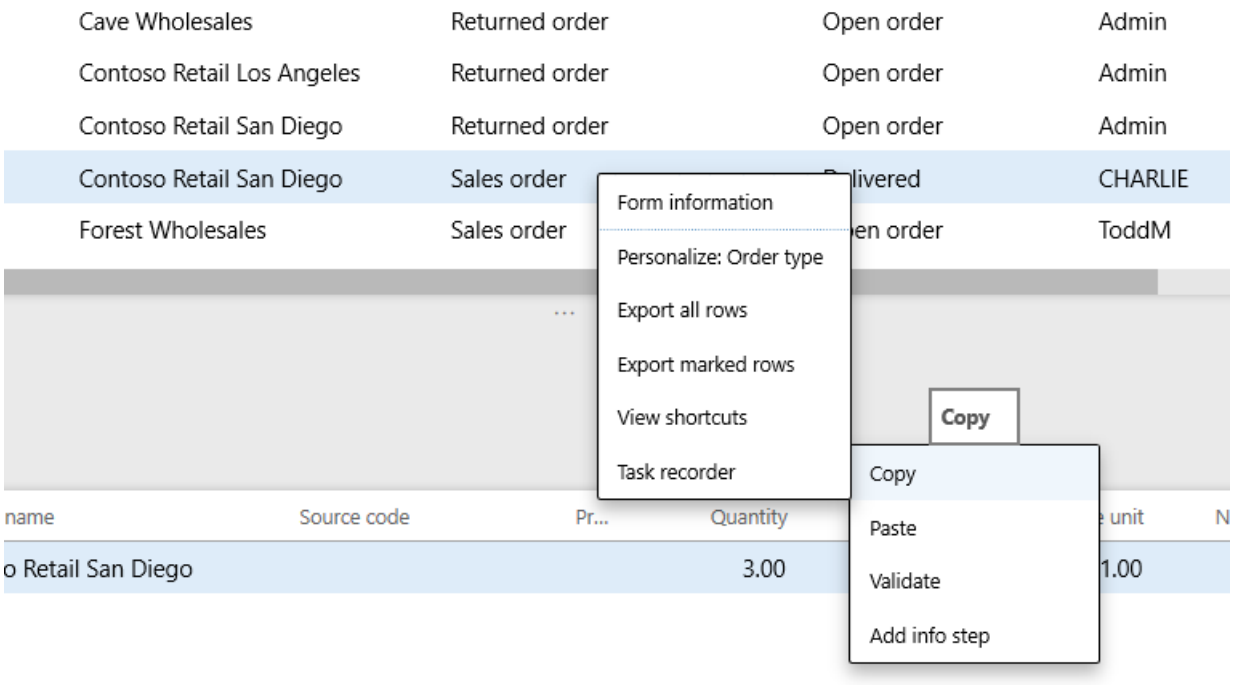
> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Copy variables to chain test cases

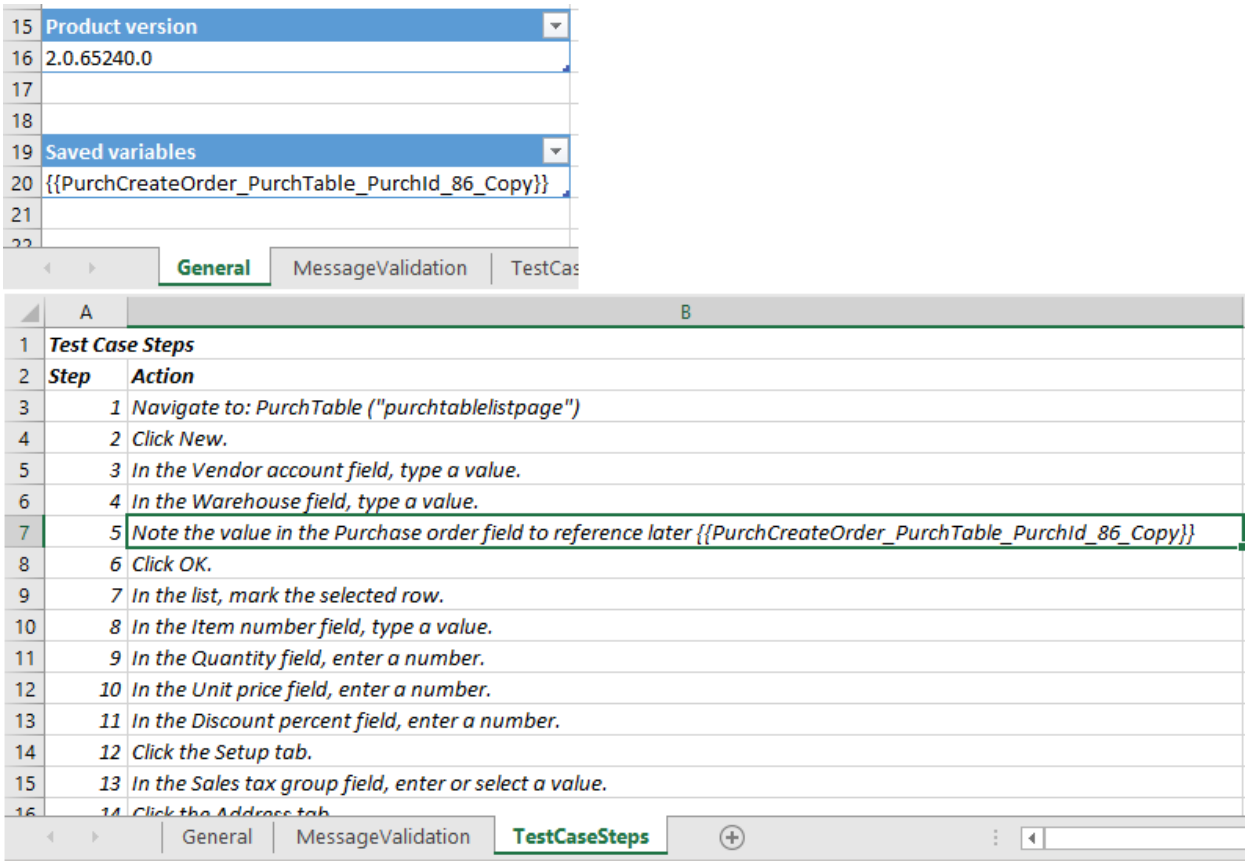
2/18/2021 • 2 minutes to read • Edit Online

One of the key features of the Regression Suite Automation Tool is the chaining of test cases, that is, the ability of a test to pass values to other tests. Test cases are executed according to their defined order in the Azure DevOps test plan, which can also be updated in the test tool itself. It is important to correctly order the tests if you want to pass variables from one test case to the other.

To save the value of a variable while recording the test in Task Recorder, right-click the field and select **Task recorder > Copy**, as shown in the following image. Copying will save the variable in the recording file. This variable can be used in subsequent tests.



When RSAT generates the Excel parameters file, saved variables appear in the **Saved variables** table on the **General** Tab. These variables also appear in the context of the test case steps in the **TestCaseSteps** tab. In the image below, the purchase order ID value was copied during the recording of the test case (step 5). This value is stored in a variable named **{{PurchCreateOrder_PurchTable_PurchId_86_Copy}}**.



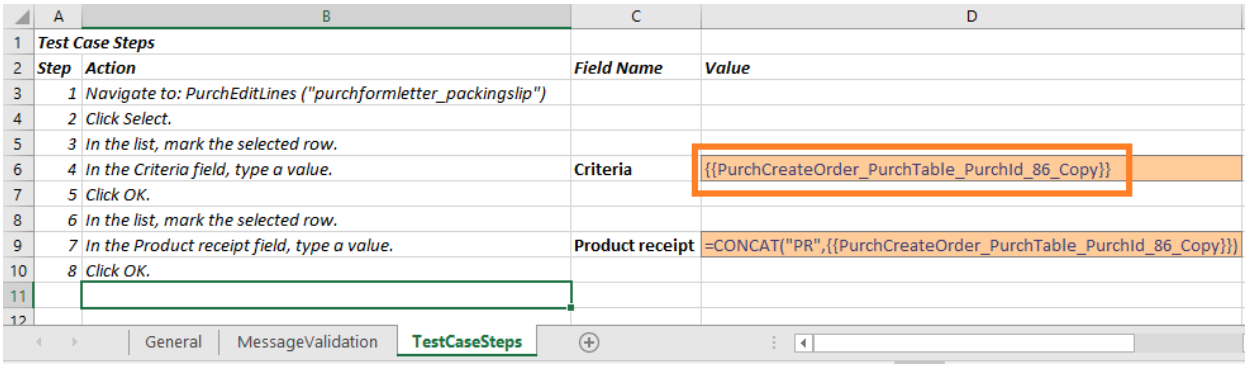
	A	B
15	Product version	
16	2.0.65240.0	
17		
18		
19	Saved variables	
20	{{PurchCreateOrder_PurchTable_PurchId_86_Copy}}	
21		

General | MessageValidation | TestCas

	A	B
1	*Test Case Steps*	
2	*Step*	*Action*
3	*1*	*Navigate to: PurchTable ("purchtablelistpage")*
4	*2*	*Click New.*
5	*3*	*In the Vendor account field, type a value.*
6	*4*	*In the Warehouse field, type a value.*
7	*5*	*Note the value in the Purchase order field to reference later {{PurchCreateOrder_PurchTable_PurchId_86_Copy}}*
8	*6*	*Click OK.*
9	*7*	*In the list, mark the selected row.*
10	*8*	*In the Item number field, type a value.*
11	*9*	*In the Quantity field, enter a number.*
12	*10*	*In the Unit price field, enter a number.*
13	*11*	*In the Discount percent field, enter a number.*
14	*12*	*Click the Setup tab.*
15	*13*	*In the Sales tax group field, enter or select a value.*
16	*14*	*Click the Address tab*

General | MessageValidation | TestCaseSteps

To reuse these variables during test playback, copy the variable name and use it in place of a parameter value in the data file of another test (or the same test), as shown below.



	A	B	C	D
1	*Test Case Steps*			
2	*Step*	*Action*	*Field Name*	*Value*
3	*1*	*Navigate to: PurchEditLines ("purchformletter_packingslip")*		
4	*2*	*Click Select.*		
5	*3*	*In the list, mark the selected row.*		
6	*4*	*In the Criteria field, type a value.*	**Criteria**	{{PurchCreateOrder_PurchTable_PurchId_86_Copy}}
7	*5*	*Click OK.*		
8	*6*	*In the list, mark the selected row.*		
9	*7*	*In the Product receipt field, type a value.*	**Product receipt**	=CONCAT("PR",{{PurchCreateOrder_PurchTable_PurchId_86_Copy}})
10	*8*	*Click OK.*		
11				
12				

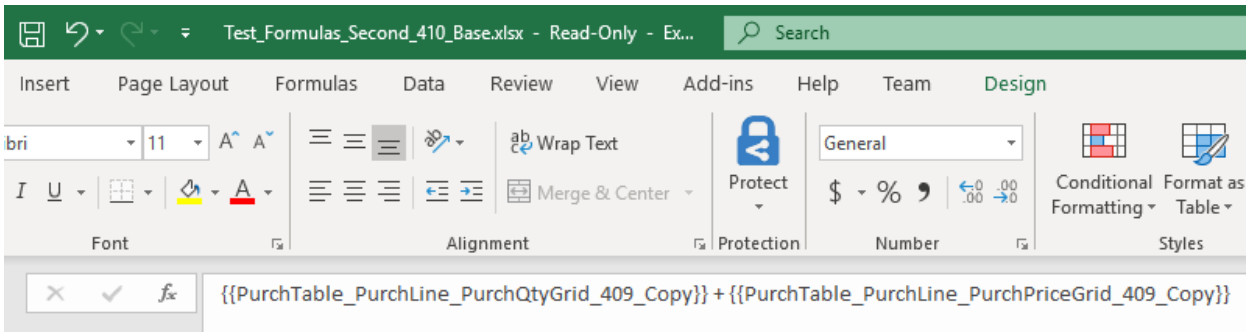
General | MessageValidation | TestCaseSteps

Variables can be used in the same test case where they are defined and can also be passed between tests during the same test run.

Support for formulas of saved variables

You can create formulas that contain saved (copied) variables. If you have been using an older version of the Regression Suite Automation Tool, you will need to regenerate new Excel parameter files to take advantage of this functionality. Supported operators are `+`, `-`, `/` and ''. Only numerical variables can be used in the Regression Suite Automation Tool formulas. Strings or dates are not supported. Always specify variable names within double braces `{{varname}}`. For example, `{{var1}} + {{var2}}`.

In the image below, two different variables are used in a formula.



As of RSAT version 1.220, you can also use Excel functions, such as **ROUND**, **CONCAT**, and **UPPER**, to create formulas with RSAT variables. This feature is implemented using the Excel formula evaluation functionality, so any function supported by Excel is supported by RSAT.

For example,

- To round a value into the nearest whole number, use:

  ```
  =ROUND({{Item_Price_3274_Copy}}, 0)
  ```

- To concatenate strings, use:

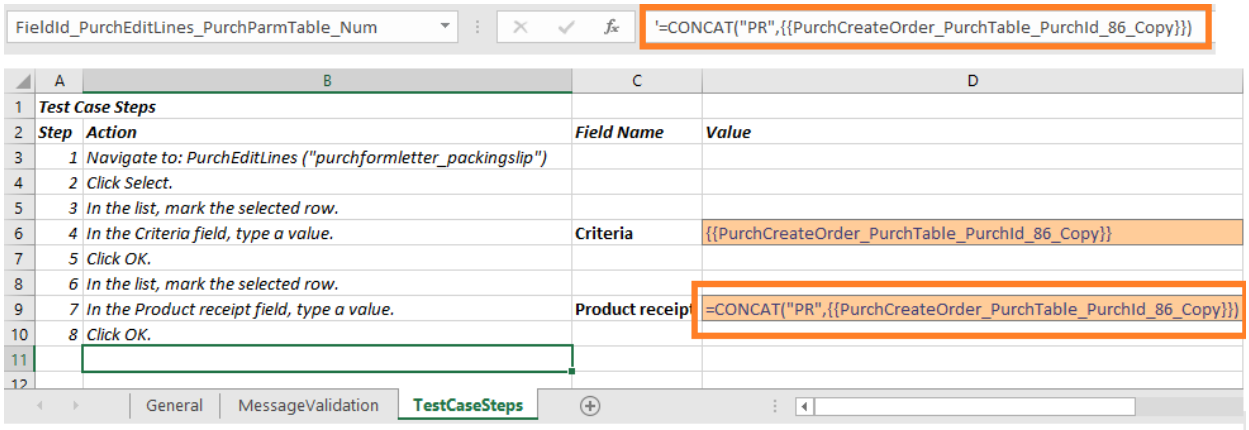
  ```
  =CONCATENATE({{AccountNum_3274_Copy}}, " ", {{ AddressBP_Locator_3274_Copy}})
  ```

- To calculate and format a date and convert it to a string, use:

  ```
  =TEXT(DATEVALUE({{SystemDate_CurrentDate_3276_Copy}}) - 1, "mm/dd/yyyy")
  ```

  Always convert RSAT date values to text for reliable test case execution.

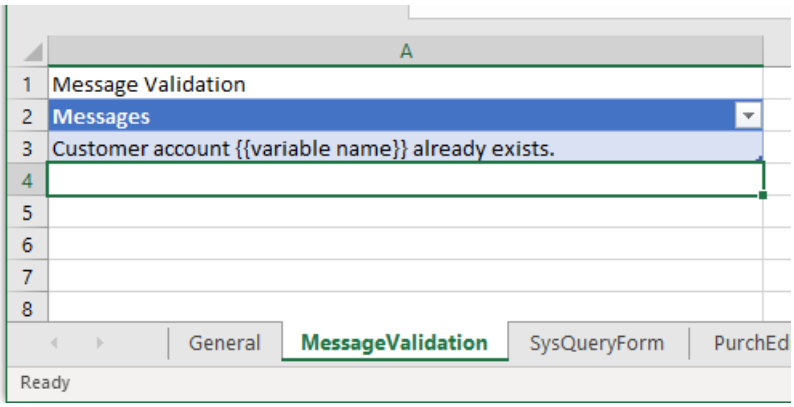
RSAT evaluates these formulas during test execution, so you must precede the formula with a single quote ' to prevent Excel from attempting to prematurely calculate the formula. An example is shown in this image.



Use variables in message validation

You can also use a saved variable as part of a string in the Message Validation tab. Here is an example that validates that the message `Customer account {{variable name}} already exists.`. It appears in the Infolog during test execution. `{{variable name}}` is a variable that is copied during the recording.

Saved (Copied) variables can be used within the same test case or across more than one test case in the same test suite.



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

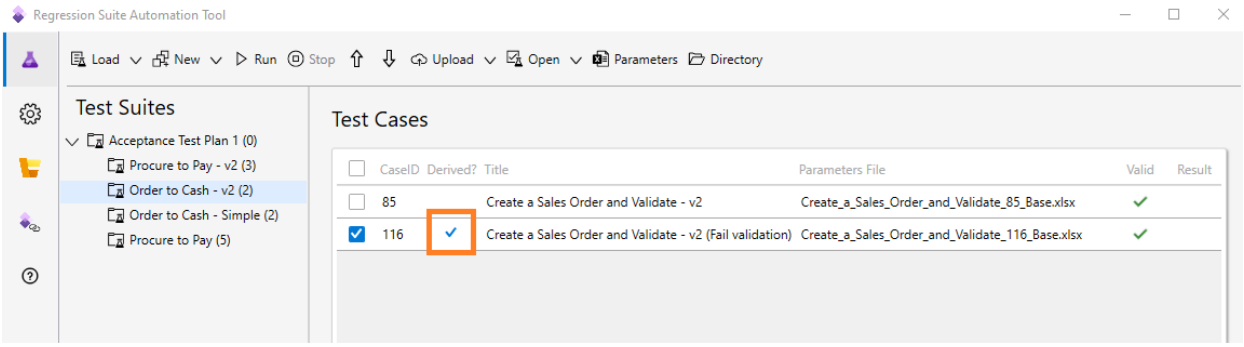
The survey will take about seven minutes. No personal data is collected (privacy statement).

Derived test cases

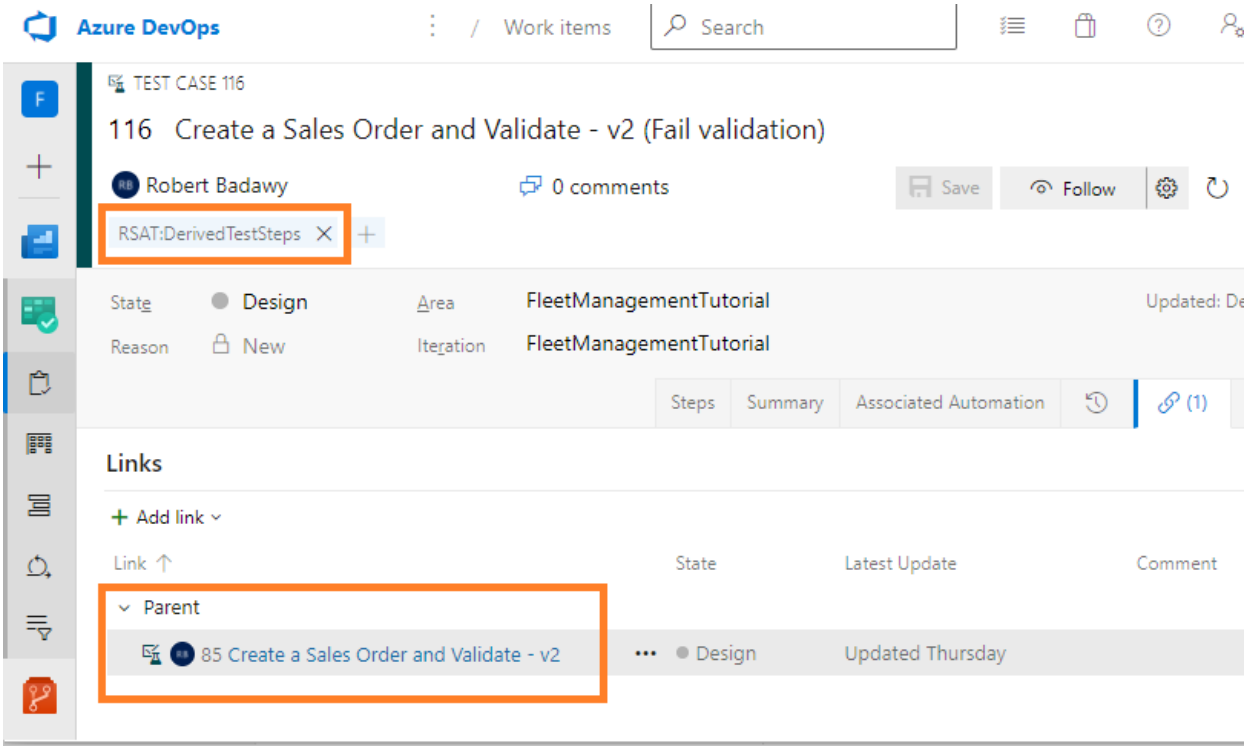
2/18/2021 • 2 minutes to read • Edit Online

The Regression suite automation tool (RSAT) lets you use the same task recording with multiple test cases, so that you can run a task with different data configurations. Select a test case in the Regression suite automation tool and then select **New > Create Derived Test Case**. This creates a child test case in Azure DevOps. The resulting derived test case is linked to its parent test case in Azure DevOps. It has an Excel parameters file attached but no recording file. The derived test case will appear in the Regression suite automation tool grid under the same test suite with the **Derived** column selected. By default, derived test cases are named after their parent test case with a numeric suffix.

In the following image, a derived test case has been created from a test case named **Create a Sales Order and Validate - v2**. The derived test case has been renamed (in Azure DevOps) to **Create a Sales Order and Validate - v2 (Fail validation)**.



In Azure DevOps, a derived test case is a child item of the **Create a Sales Order and Validate - v2** test case and is tagged with the special keyword **RSAT:DerivedTestSteps**.



When you run a derived test case, it will use the recording of its parent test case and its own copy of the Excel parameters file. This will allow you to run the same test with different parameters without the need to maintain more than one recording.

A derived test case does not need to be part of the same test suite as its parent test case, and you can use it in another suite. You can also rename a derived test case. You can edit the Excel parameters file of a derived test case to run it with a different user, a different company, or with different input and validation parameters than its parent test case.

Regression suite automation tool tutorial

2/18/2021 • 13 minutes to read • Edit Online

> **NOTE**
> Use your internet browser tools to download and save this page in pdf format.

This tutorial walks through some of the advanced features of the Regression suite automation tool (RSAT), includes a demo assignment, and describes strategy and key learning points.

Notable Features of RSAT and Task recorder

Validate a field value

RSAT allows you to include validation steps within your test case to validate expected values. For information about this feature, see the article Validate expected values.

The following example shows how you can use this feature to validate whether the on-hand inventory is more than 0 (zero).

1. In the demo data in the **USMF** company, create a task recording that has the following steps:
    a. Go to **Product information management > Products > Released products**.
    b. Use the Quick Filter to find records. For example, filter on a value of **1000** for the **Item number** field.
    c. Select **On-hand inventory**.
    d. Use the Quick Filter to find records. For example, filter on a value of **1** for the **Site** field.
    e. In the list, mark the selected row.
    f. Validate that the value of the **Total available** field is **411.0000000000000000**.
2. Save the task recording as a **developer recording** and attach it to your test case in Azure Devops.
3. Add the test case to the test plan, and load the test case into RSAT.
4. Open the Excel parameter file and go to the **TestCaseSteps** tab.
5. To validate whether the inventory on-hand will always be more than **0**, go to the **Validate Total Available** step and change its value from **411** to **0**. Change the value of the **Operator** field from an equal sign (=) to a greater than sign (>).
6. Save and close the Excel parameter file.
7. Select **Upload** to save the changes that you made to the Excel parameter file to Azure DevOps.

Now, if the value of the **Total Available** field for the specified item in inventory is more than 0 (zero), tests will pass, regardless of the actual on-hand inventory value.

Saved variables and chaining of test cases

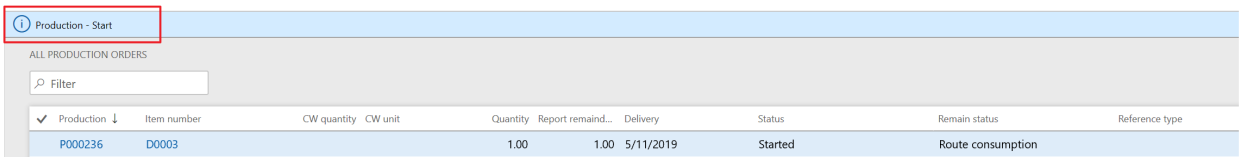
One of the key features of RSAT is the chaining of test cases, that is, the ability of a test to pass variables to other tests. For more information, see the article Copy variables to chain test cases.

Derived test case

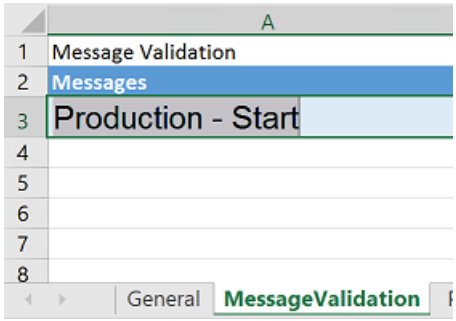
RSAT lets you use the same task recording with multiple test cases, enabling a task to run with different data configurations. See the article Derived test cases for more information.

Validate notifications and messages

This feature can be used to validate whether an action occurred. For example, when a production order is created, estimated, and then started, the app shows a "Production – Start" message to notify you that the production order has been started.



You can validate this message through RSAT by entering the message text on the **MessageValidation** tab of the Excel parameter file for the appropriate recording.



After the test case is run, the message in the Excel parameter file is compared to the message that is shown. If the messages don't match, the test case will fail.

> **NOTE**
>
> You can enter more than one message on the **MessageValidation** tab in the Excel parameter file. The messages also can be error or warning messages instead of informational messages.

Snapshot

This feature takes screenshots of the steps that were performed during task recording. It is useful for auditing or debugging purposes.

- To use this feature, open the **Microsoft.Dynamics.RegressionSuite.WindowsApp.exe.config** file under the RSAT installation folder (for example, **C:\Program Files (x86)\Regression Suite Automation Tool**), and change the value of the following element from **false** to **true**.

  ```
  <add key="VerboseSnapshotsEnabled" value="false" />
  ```

When your run the test case, RSAT will generate snapshots (images) of the steps in the playback folder of the test cases in the working diretory. If you are using an older version of RSAT, the images are saved to **C:\Users\<Username>\AppData\Roaming\regressionTool\playback**, a separate folder is created for each test case that is run.

Assignment

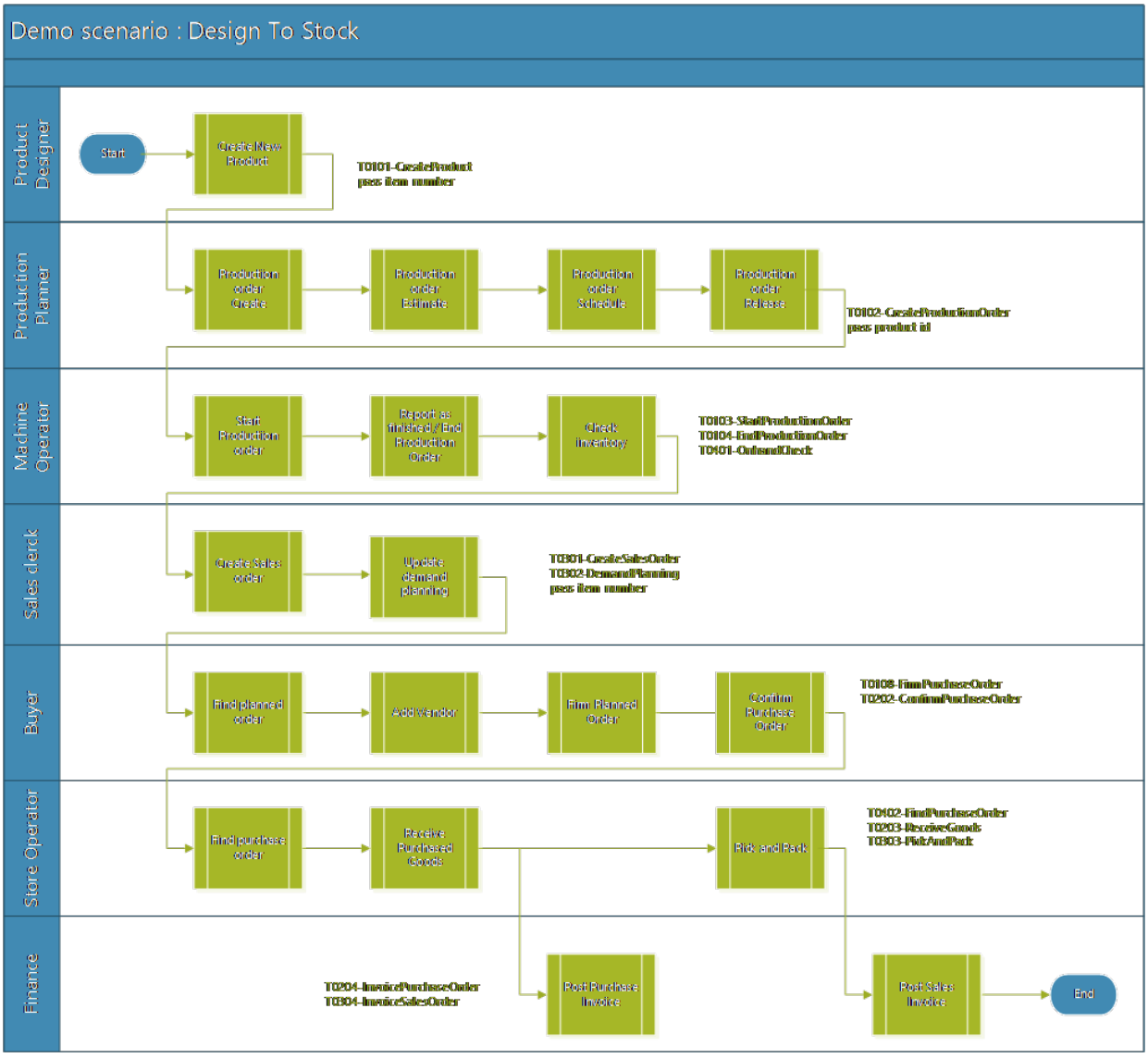
Scenario

1. The product designer creates a new released product.
2. The production manager initiates a production order to bring the stock level to two pieces.
3. Manufacturing starts and ends the production order, and verifies that the on-hand quantity is two pieces.
4. The sales team receives an order for four pieces of the new product. Therefore, the sales team updates the net requirements via the dynamic plan. Because no additional capacity is available, the default order policy is

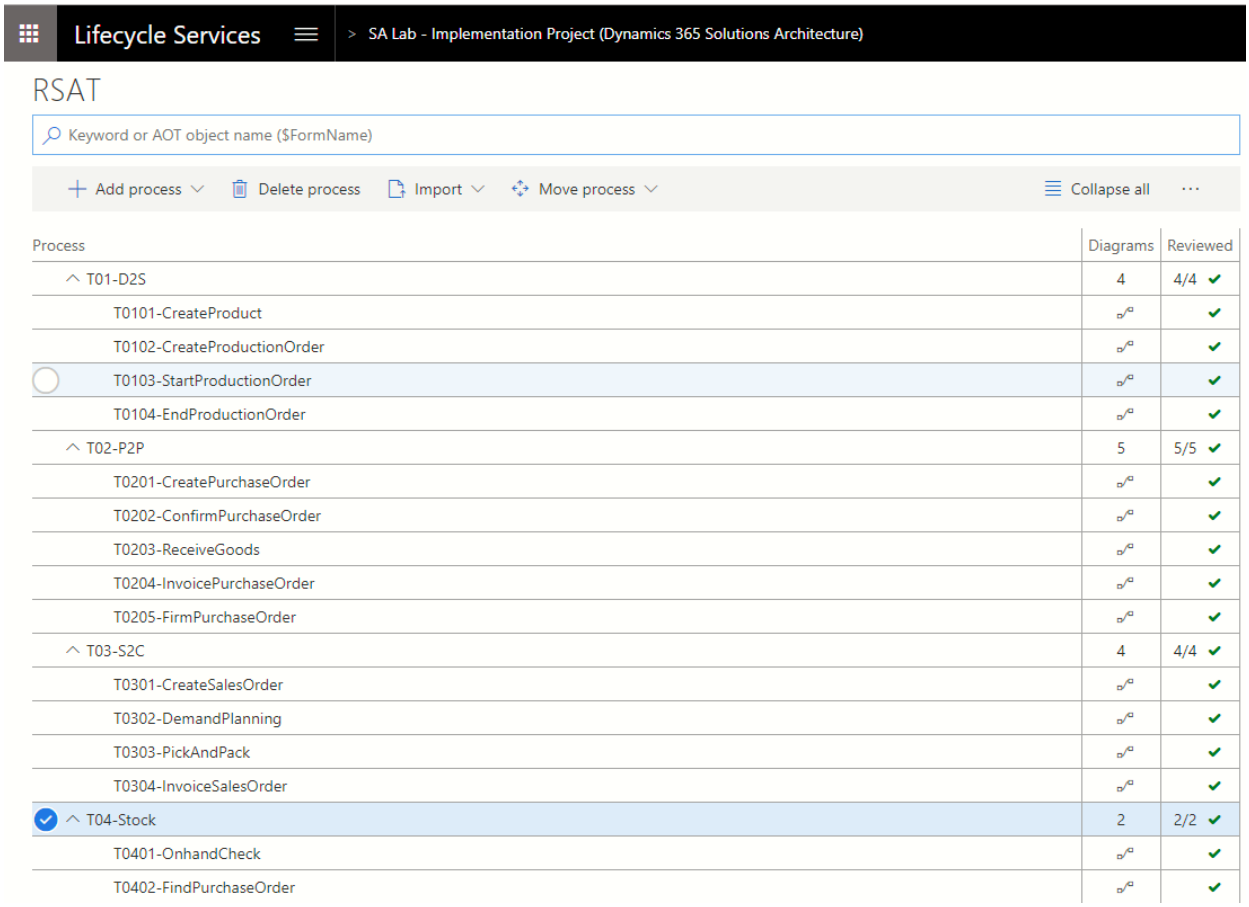
set to "buy instead of make." Therefore, a planned purchase order is created.

5. The buyer adds a vendor, firms the planned purchase order, and then confirms the purchase order.
6. When the goods that were purchased arrive at the store, the store operator searches the related purchase order and receives the goods. Because the order is now completed, goods can be picked and packed against the sales order.
7. Finance posts the purchase invoice and sales invoice.

The following illustration shows the flow for this scenario.



The following illustration shows the business processes hierarchy for this scenario in the LCS Business Process Modeler.



Strategy – Key learning

Data

- Make sure that you have representative data volumes (a copy of production/golden configuration data plus migrated data).
- When you generate new data via Task recorder, create test names that won't conflict with existing names (for example, use a prefix such as **RSATxxx**).
- Use Azure Point-In-Time restore to rerun tests in non-Tier 1 environments.
- Although you can use the **RANDOM** and **NOW** Excel functions to generate a unique combination, the effort is considerably high. Here is an example.

```
product = "AT" &TEXT(NOW(),"yyymmddhhmm")
```

Task recorder

- Define scenarios before you start recording. A well-managed project has predefined test scenarios. To build a test case, consider how predictable the outcome of those test scenarios is.
- Split recordings if they are performed by different roles, or if there is waiting time or an external event before the next step.
- Avoid selecting values in lists. Instead, use text formats, such as **FIFO**, **AudioRM**, and **SiteWH**. When you select in a list, the position of the value in the list is recorded, not the value itself. If items are added to that list, the position of the value can change. Therefore, your recording will use a different parameter, and the rest of the scenario might be affected.
- Think about multi-user behavior. For example, don't assume that your newly created sales order will always be automatically selected. Instead, always use the filter to find the correct order.
- Use the Copy function in Task recorder to save the name of a newly created product so it can be used in

chained test cases.

- Use the Validate function in Task recorder to set checkpoints that verify that steps have been run correctly.

RSAT

- To run the test in another company, you can change the company on the **General** tab of the Excel parameter file. Make sure that settings and data are available in the newly selected company.
- You can change the test user on the **General** tab of the Excel parameter file. Specify the email ID of the user who will run the test case. In this way, the test case can be run by using the security permissions of the specified user.
- To wait before the test is started, you can define a pause on the **General** tab of the Excel parameter file. This pause can be used in a batch job (for example, if a workflow must be run before the next step can be performed.)

Advanced scripting

CLI

RSAT can be called from a **Command Prompt** or **PowerShell** window.

> **NOTE**
>
> Verify that the **TestRoot** environment variable is set to the RSAT installation path. (In Microsoft Windows, open **Control Panel**, select **System and Security > System > Advanced system settings**, and then select **Environment Variables**.)

1. Open a **Command Prompt** or **PowerShell** window as an admin.
2. Navigate to the RSAT installation directory.

   ```
   cd "c:\Program Files (x86)\Regression Suite Automation Tool\"
   ```

3. List all commands.

```
C:\Program Files (x86)\Regression Suite Automation
Tool>Microsoft.Dynamics.RegressionSuite.ConsoleApp.exe help

Usage:
    Microsoft.Dynamics.RegressionSuite.ConsoleApp.exe command
    or
    Microsoft.Dynamics.RegressionSuite.ConsoleApp.exe /settings "C:\Path to\file.settings" command

Available commands:
    ?
    about
    cls
    download
    edit
    generate
    generatederived
    generatetestonly
    generatetestsuite
    help
    list
    listtestplans
    listtestsuite
    listtestsuitenames
    playback
    playbackbyid
    playbackmany
    playbacksuite
    quit
    upload
    uploadrecording
    usage
```

?

Shows help about all available commands and their parameters.

`Microsoft.Dynamics.RegressionSuite.ConsoleApp` `?` `[command]`

?: Optional parameters

`command`: Where `[command]` is one of the commands specified below.

about

Displays the current version.

`Microsoft.Dynamics.RegressionSuite.ConsoleApp` `about`

cls

Clears the screen.

`Microsoft.Dynamics.RegressionSuite.ConsoleApp` `cls`

download

Downloads attachments for the specified test case to the output directory. You can use the `list` command to get all available test cases. Use any value from the first column as a **test_case_id** parameter.

`Microsoft.Dynamics.RegressionSuite.ConsoleApp` `download` `[test_case_id] [output_dir]`

download: required parameters

- `test_case_id`: Represents the test case ID.
- `output_dir`: Represents the output directory. The directory must exist.

download: examples

`download 123 c:\temp\rsat`

`download 765 c:\rsat\last`

edit

Allows you to open parameters file in Excel program and edit it.

```
Microsoft.Dynamics.RegressionSuite.ConsoleApp edit [excel_file]
```

edit: required parameters

- `excel_file`: Must contain a full path to an existing Excel file.

edit: examples

```
edit c:\RSAT\TestCase_123_Base.xlsx
```

```
edit e:\temp\TestCase_456_Base.xlsx
```

generate

Generates test execution and parameter files for the specified test case in the output directory. You can use the `list` command to get all available test cases. Use any value from the first column as a **test_case_id** parameter.

```
Microsoft.Dynamics.RegressionSuite.ConsoleApp generate [test_case_id] [output_dir]
```

generate: required parameters

- `test_case_id`: Represents the test case ID.
- `output_dir`: Represents the output directory. The directory must exist.

generate: examples

```
generate 123 c:\temp\rsat
```

```
generate 765 c:\rsat\last
```

generatederived

Generates a new test case, derived from the provided test case. You can use the `list` command to get all available test cases. Use any value from the first column as a **test_case_id** parameter.

```
Microsoft.Dynamics.RegressionSuite.ConsoleApp generatederived
[parent_test_case_id] [test_plan_id] [test_suite_id]
```

generatederived: required parameters

- `parent_test_case_id`: Represents the parent test case ID.
- `test_plan_id`: Represents the test plan ID.
- `test_suite_id`: Represents the test suite ID.

generatederived: examples

```
generatederived 123 8901 678
```

generatetestonly

Generates only test execution file for the specified test case in the output directory. You can use the `list` command to get all available test cases. Use any value from the first column as a **test_case_id** parameter.

```
Microsoft.Dynamics.RegressionSuite.ConsoleApp generatetestonly [test_case_id] [output_dir]
```

generatetestonly: required parameters

- `test_case_id`: Represents the test case ID.
- `output_dir`: Represents the output directory. The directory must exist.

generatetestonly: examples

```
generatetestonly 123 c:\temp\rsat
```

```
generatetestonly 765 c:\rsat\last
```

generatetestsuite

Generates all test cases for the specified suite in the output directory. You can use `listtestsuitenames` command to get all available test suits. Use any value from the column as a **test_suite_name** parameter.

`Microsoft.Dynamics.RegressionSuite.ConsoleApp` `generatetestsuite` `[test_suite_name] [output_dir]`

generatetestsuite: required parameters

- `test_suite_name`: Represents the test suite name.
- `output_dir`: Represents the output directory. The directory must exist.

generatetestsuite: examples

```
generatetestsuite Tests c:\temp\rsat
```

```
generatetestsuite Purchase c:\rsat\last
```

help

Identical to the ? command.

list

Lists all available test cases.

`Microsoft.Dynamics.RegressionSuite.ConsoleApp` `list`

listtestplans

Lists all available test plans.

`Microsoft.Dynamics.RegressionSuite.ConsoleApp` `listtestplans`

listtestsuite

Lists test cases for the specified test suite. You can use `listtestsuitenames` command to get all available test suites. Use any value from first column as **suite_name** parameter.

`Microsoft.Dynamics.RegressionSuite.ConsoleApp` `listtestsuite` `[suite_name]`

listtestsuite: required parameters

- `suite_name`: Name of the desired suite.

listtestsuite: examples

```
listtestsuite "sample suite name"
```

```
listtestsuite NameOfTheSuite
```

listtestsuitenames

Lists all available test suites.

`Microsoft.Dynamics.RegressionSuite.ConsoleApp` `listtestsuitenames`

playback

Plays back a test case using an Excel file.

`Microsoft.Dynamics.RegressionSuite.ConsoleApp` `playback` `[excel_file]`

playback: required parameters

- `excel_file`: A full path to the Excel file. File must exist.

playback: examples

```
playback c:\RSAT\TestCaseParameters\sample1.xlsx
```

```
playback e:\temp\test.xlsx
```

playbackbyid

Plays back multiple test cases at once. You can use the `list` command to get all available test cases. Use any value from the first column as a **test_case_id** parameter.

`Microsoft.Dynamics.RegressionSuite.ConsoleApp` `playbackbyid`
`[test_case_id1] [test_case_id2] ... [test_case_idN]`

playbackbyid: required parameters

- `test_case_id1` : ID of exisiting test case.
- `test_case_id2` : ID of exisiting test case.
- `test_case_idN` : ID of exisiting test case.

playbackbyid: examples

```
playbackbyid 878
```

```
playbackbyid 2345 667 135
```

playbackmany

Plays back many test cases at once, using Excel files.

```
Microsoft.Dynamics.RegressionSuite.ConsoleApp playbackmany [excel_file1] [excel_file2] ... [excel_fileN]
```

playbackmany: required parameters

- `excel_file1` : Full path to the Excel file. File must exist.
- `excel_file2` : Full path to the Excel file. File must exist.
- `excel_fileN` : Full path to the Excel file. File must exist.

playbackmany: examples

```
playbackmany c:\RSAT\TestCaseParameters\param1.xlsx
```

```
playbackmany e:\temp\test.xlsx f:\rsat\sample1.xlsx c:\RSAT\sample2.xlsx
```

playbacksuite

Plays back all test cases from the specified test suite. You can use `listtestsuitenames` command to get all available test suites. Use any value from first column as **suite_name** parameter.

```
Microsoft.Dynamics.RegressionSuite.ConsoleApp playbacksuite [suite_name]
```

playbacksuite: required parameters

- `suite_name` : Name of the desired suite.

playbacksuite: examples

```
playbacksuite suiteName
```

```
playbacksuite sample_suite
```

quit

Closes the application.

```
Microsoft.Dynamics.RegressionSuite.ConsoleApp quit
```

upload

Uploads all files belonging to the specified test suite or test cases.

```
Microsoft.Dynamics.RegressionSuite.ConsoleApp upload [suite_name] [testcase_id]
```

upload: required parameters

- `suite_name` : All files belonging to the specified test suite will be uploaded.
- `testcase_id` : All files beloning to the specified test case(s) will be uploaded.

upload: examples

```
upload sample_suite
```

```
upload 123
```

```
upload 123 456
```

uploadrecording

Uploads only recording file belonging to the specified test cases.

```
Microsoft.Dynamics.RegressionSuite.ConsoleApp uploadrecording [testcase_id]
```

uploadrecording: required parameters

- `testcase_id`: Recording file belonging to the specified test cases will be uploaded.

uploadrecording: examples

`uploadrecording 123`

`uploadrecording 123 456`

usage

Shows two ways to invoke this application: one using a default setting file, another one providing a setting file.

`Microsoft.Dynamics.RegressionSuite.ConsoleApp` `usage`

Windows PowerShell examples

Run a test case in a loop

You have a test script that creates a new customer. Via scripting, this test case can be run in a loop by randomizing the following data before each iteration is run:

- Customer ID
- Customer name
- Customer address

The customer ID will be in the format *ATCUS<number>*, where <number> is a value between **000000001** and **999999999**.

The following example uses one parameter, **start**, to define the first number that is used. Is uses a second parameter, **nr**, to define the number of customers that must be created. For each iteration, the parameters in the Excel parameter file are changed by using an UpdateCustomer function. Then the RSAT command line is called in a RunTestCase function.

Open Microsoft Windows PowerShell Integrated Scripting Environment (ISE) in admin mode, and paste the following code into the window that is named **Untitled1.ps1**.

```
param ( [int]$start = 1, [int]$nr = 1 )
function UpdateCustomer
{
    param ([string]$paramFilename, [string]$sheetName, [string]$CustId)
    $xl = New-Object -COM "Excel.Application"
    $xl.Visible = $false
    $wb = $xl.Workbooks.Open($paramFilename)
    $ws = $wb.Sheets.Item($sheetName)
    $ws.Cells.Item(3, 2).Value = "ATCUS" + $CustId
    $ws.Cells.Item(4, 2).Value = "Automated Test Customer " + $CustId
    $ws.Cells.Item(8, 2).Value = "Automated Test Street " + $CustId
    $wb.Save()
    $wb.Close()
    $xl.Quit()
    [System.Runtime.Interopservices.Marshal]::ReleaseComObject($xl)
}
function RunTestCase
{
    param ( [string]$filename )
    $cmd = "cd c:\Program Files (x86)\Regression Suite Automation Tool\ &&  "
    $cmd = $cmd + "Microsoft.Dynamics.RegressionSuite.ConsoleApp.exe playback "
    $cmd = $cmd + $filename
    cmd /c $cmd
}
$excelFilename = "full path to Excel parameter file"
l$sheetName = "DirPartyQuickCreateForm"
for ($i = $start; $i -lt $start + $nr; $i++ )
{
    $CustomerId = $i.ToString("000000000")
    Write-Host "customer : " $CustomerId
    UpdateCustomer $excelFilename $sheetName $CustomerId
    RunTestCase $excelFilename
```

Run a script that depends on data in Microsoft Dynamics 365

The following example uses an Open Data Protocol (OData) call to find the order status of a purchase order. If the status isn't **invoiced**, you can, for example, call an RSAT test case that posts the invoice.

```
function Odata_Get
{
    Param ( [string] $environment, [string] $cmd )
    [Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12
    $tenant = "your tenant"
    $creds = @{
        grant_type = "client_credentials"
        client_id = "your client application Id"
        client_secret = "your client secret"
        resource = $environment
    }
    $headers = $null
    $bearer = Invoke-RestMethod https://login.microsoftonline.com/$tenant/oauth2/token -Method Post -Body
$creds -Headers $headers;
    $headers = @{
        Authorization = "Bearer " + $bearer.access_token
    }
    $Odata_cmd = $environment + '/data/' + $cmd
    return (Invoke-RestMethod -Uri $Odata_cmd -Method Get -Headers $headers -ContentType application/json )
}
function PurchaseOrderStatus
{
    Param ( [string] $environment, [string] $purchaseOrderNumber )
    $cmd = 'PurchaseOrderHeaders?$filter=PurchaseOrderNumber eq '
    $cmd = $cmd + "'" + $purchaseOrderNumber + "'"
    $response = Odata_Get -environment $environment -cmd $cmd
    return $response.value.PurchaseOrderStatus
}
$environment = "https://your environment"
$orderStatus = PurchaseOrderStatus -environment $environment -purchaseOrderNumber '000003'
if ($orderStatus -eq $null) {   write-host 'doesn''t exist'}
elseif ($orderStatus -ne 'invoiced') { RunTestCase "PostInvoice" }
```

Regression suite automation tool best practices

2/18/2021 • 5 minutes to read • Edit Online

This topic describes best practices and common use cases of the Regression suite automation tool (RSAT) and Task recorder.

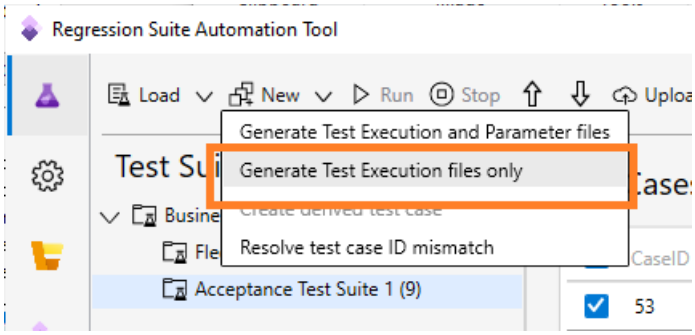
Author test cases using the Task recorder

When you author task recordings for RSAT, follow these practices:

1. Make sure all your recordings start on the main dashboard.
2. Keep individual recordings short and focus on a business task performed by one user, like creating a sales order. This simplifies maintainability and reusability of test cases.
3. Chart controls are not supported. Any task recording actions related to charts will be ignored by RSAT during test case playback.
4. When creating a recording, make sure to select a tab header even if the tab is already open. For example, you can switch to another tab and then select the needed tab again to activate it before using a control on it. This will make your recording more reliable during test case playback.
5. RSAT cannot play back any test step that is not recognized by the task recorder. For example, you cannot upload a file from the local disk during play back of a test case.
6. RSAT cannot play back a **page refresh** step. Avoid refreshing a page while recording your test.

Best practices when using the Regression suite automation tool

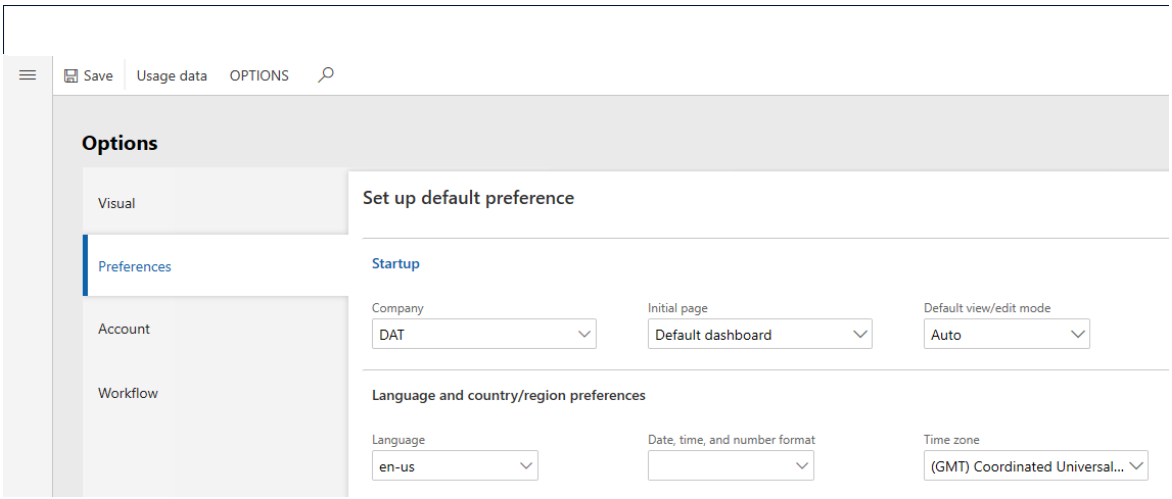
1. Upon opening the tool for the first time, select **Settings** and ensure that you have all the needed settings.
2. Before installing a new version of the tool, it is recommended to close and uninstall the previous version.
3. When you install a new version of the tool, regenerate **all** test execution files.



   It is not necessary to regenerate Microsoft Excel parameter files unless you want to take advantage of new features available in a newer format of parameter files.

4. For test parameters that need a unique value, for example, the product receipt number in the **Product Receipt** form or the invoice number in the **Vendor Invoice** form, use the **RandBetween(a,b)** Excel function to generate a unique number every time the test case is executed.
5. The default values in Excel come from the task recording. For **Reference Group** controls such as storage dimensions or tracking dimensions, it stores the key of the lookup instead of the value, for example, 2 instead of **SiteWH**. We recommend that you update these fields with the actual value in Excel so that the test is more robust and resilient to changes.

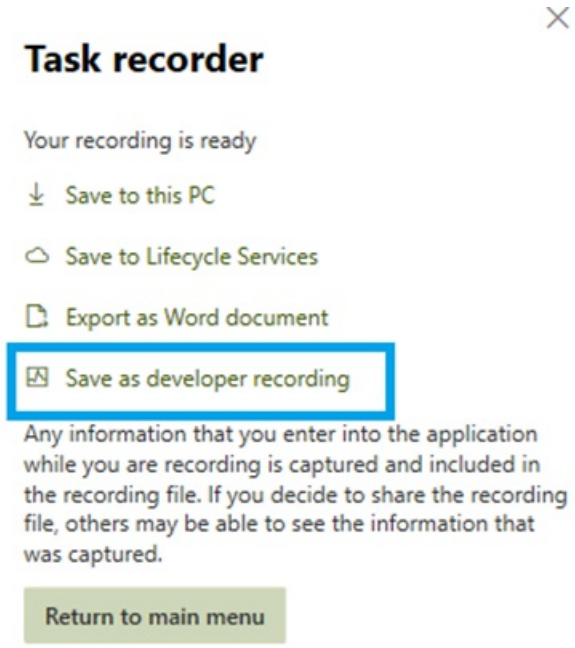
6. It is recommended to set the same locale for **Language** and **Date, time, and number format** settings of your environment prior to running RSAT. If these values are inconsistent, it may result in validation errors.



Manage local recording files

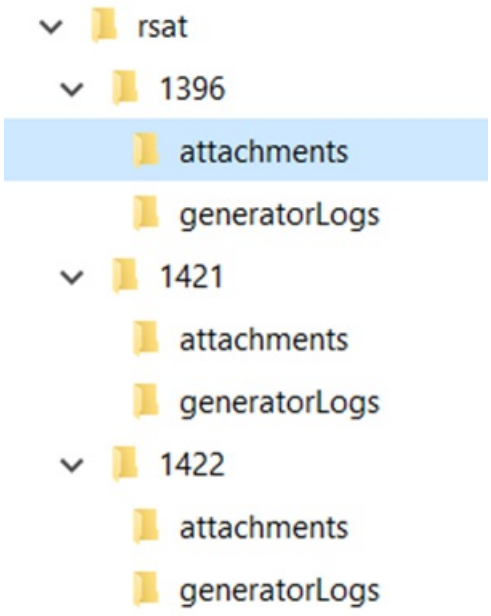
RSAT relies on Azure DevOps to store and manage test recording files (also known as task recordings). When RSAT loads a test plan from Azure DevOps, associated files are downloaded to the current **working directory** on your local computer. (This working directory is defined in RSAT settings.)

In version 1.200.42264.6 and later, it's easier to manage local recording files. You can make changes in Task recorder and then use RSAT to test them, without having to go through the Business process modeler (BPM) or Azure DevOps. When you use Task recorder, after you've finished authoring or modifying a recording, you can save it directly to your local disk as a developer recording.

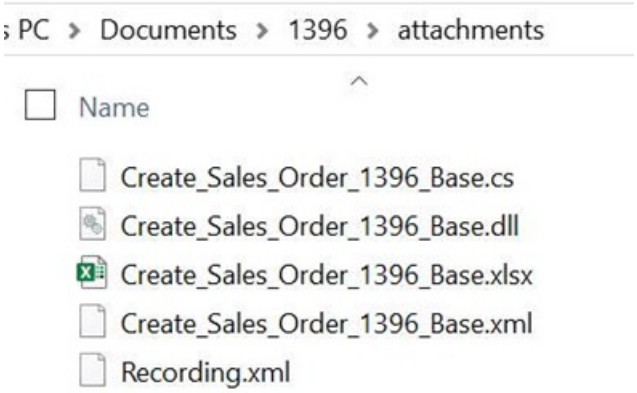


Put the recording file under the working directory that is associated with the test case. For example, if your configured working directory is `C:\Users\<username>\Documents\RSAT`, put the recording file for test case 1234 under `C:\Users\<username>\Documents\RSAT\1234\attachments`. You must name the developer recording file **Recording.xml**. Alternatively, you can name the recording file **-Test Case Title-.xml**, where **-Test Case Title-** is the title of the test case in Azure DevOps.

The following illustration shows an example of a working directory folder structure. You can open the directory directly from RSAT by clicking the folder symbol.



Each test case has its own folder, which is named after the ID of the test case. Test case attachments (recording files, automation files, and Excel parameter files) are downloaded into an attachments folder. Here is an example.



The generatorLogs directory contains log files. It doesn't contain any files that users can modify. You can ignore this directory unless RSAT support explicitly asks you to provide log files from it.

Commit a recording file to Azure DevOps

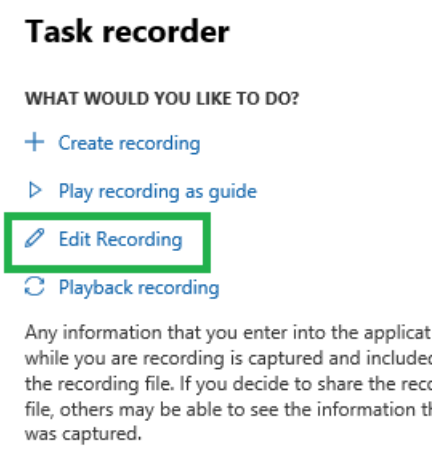
After a recording has been tested and finalized, use RSAT to upload it and commit it to Azure DevOps. The upload button has two options: **Upload automation files** and **Upload recording file**. The second option uploads only your recording file to Azure DevOps.

> **NOTE**
>
> If you're using a version of RSAT that is earlier than 1.200.37255.0, and you upgrade to the latest version, you must reload your test cases from Azure DevOps to download them into the correct directory. Otherwise, RSAT will fail, and you will receive a "File not found" error.
>
> If you're working across several DevOps projects, we recommend that you use a different working directory for each project. Otherwise, attachment files from multiple projects can become commingled in the same directory structure.

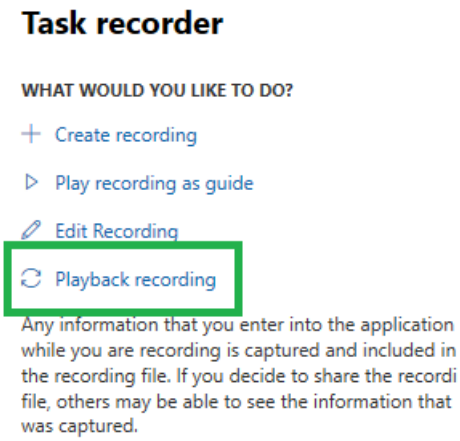
Modify (Edit) a Task recording

If you want to modify an existing task recording, note these best practices.

In the web client, open the Task recorder pane and start editing the recording using the **Edit Recording** option.



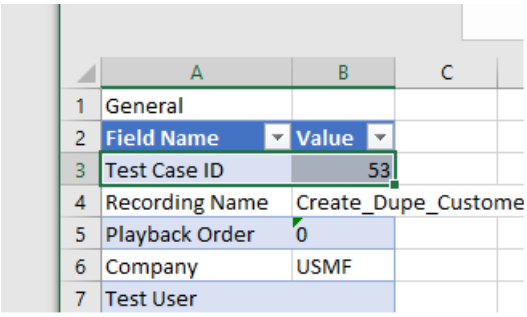
When you've finished editing the recording, play it back in the client, and verify that all the steps work correctly. Playback is required.



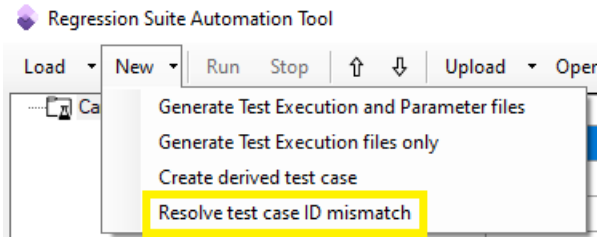
After you've finished playing back an edited recording, save it. It's then ready to be used by RSAT.

Copy test cases in Azure DevOps

As you are building your test suites in Azure DevOps, it is handy and common to duplicate test cases along with their attachments. If a copied test case contains an existing Excel parameter file attached, RSAT cannot execute it without manual edits to the Excel file. The **Test Case ID** in the Excel parameter file must match the Azure DevOps test case ID. You will need to edit all copied Excel parameter files. In the following image, the Excel file is associated with Test Case number 53 in Azure DevOps.



As of RSAT version 1.210, this process is easier. To automatically fix all occurrences of a mismatch, select the desired test cases in the grid, and then select **Resolve test case ID mismatch** in the **New** menu.



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

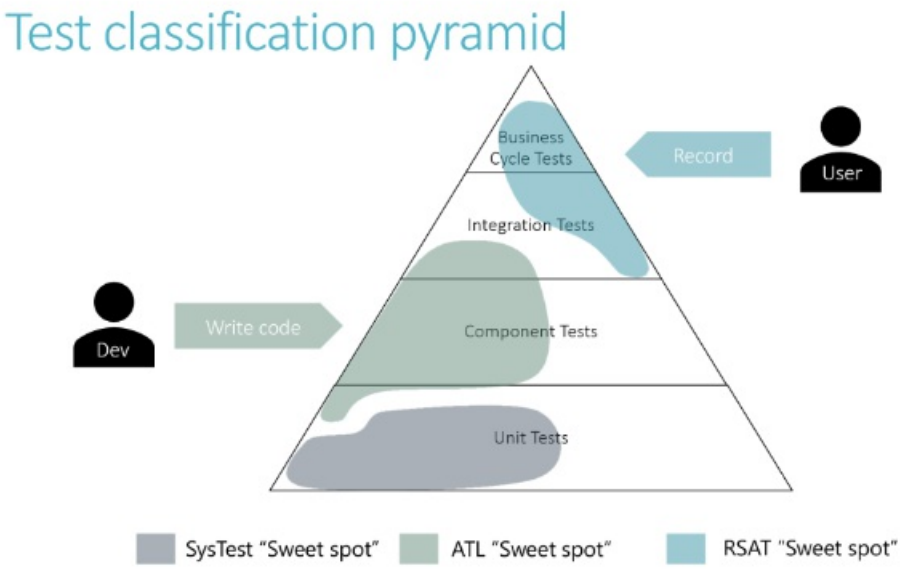
The survey will take about seven minutes. No personal data is collected (privacy statement).

Data agnostic testing using the Regression Suite Automation Tool

2/18/2021 • 2 minutes to read • Edit Online

While the functional validation of an ERP application can't be fully data agnostic, there are multiple phases and approaches for testing. These testing phases include:

- SysTest framework
- ATL frameowrk
- Regression Suite Automation Tool (RSAT)

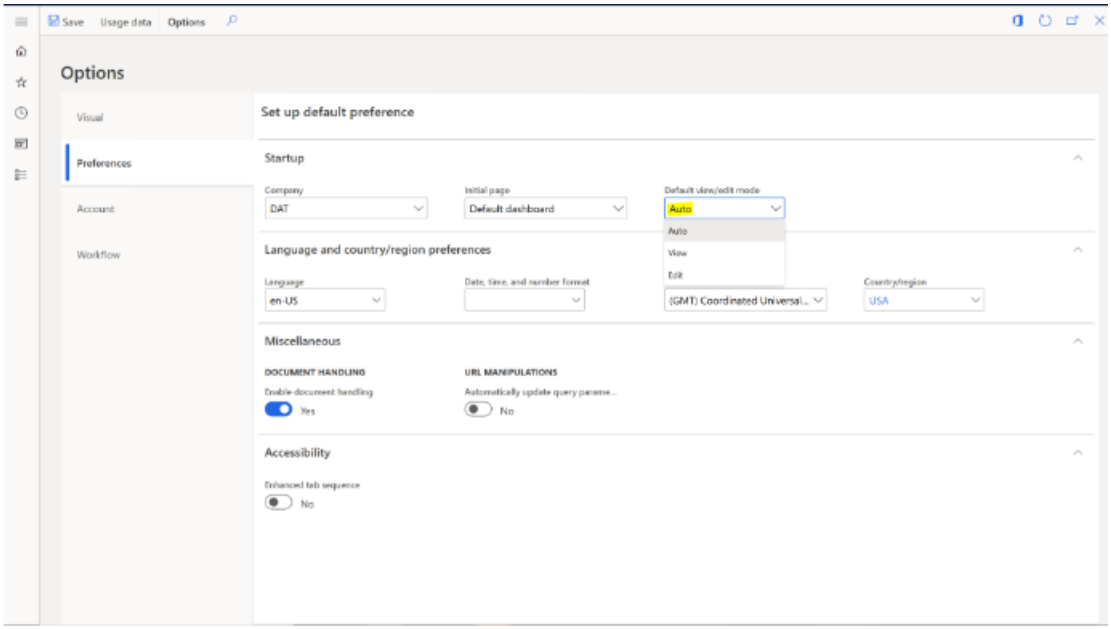


Overview

- **SysTest framework** – The SysTest framework is reliable for writing unit tests. Because unit tests are generally testing a method or function, they should always be data agnostic and dependent only on the input data that is provided as part of the test.
- **ATL framework** – Microsoft has an ATL framework that is an abstraction on the SysTest framework and makes functional test writing much more simple and reliable. This framework should be used for writing component tests or simple integration tests.
- **RSAT** – The RSAT is used for integration tests and business cycle tests. The business cycle tests, also called the regression validation tests, are dependent on existing data. However, these tests can become data agnostic if you consider additional factors.
    - o Where unit tests and component tests are low level and can fully be data agnostic (not dependent on existing dataset), the business cycle or regression validation tests are dependent on some existing data. This data includes setup, configuration settings (parameters), and master data (customer, vendors, items, etc.), but never transaction data. Make sure that during the test, if any of these are being changed, that they are reverted back as part of the final test.
    - Select master data based on certain criteria instead of selecting a particular record. For example, if you

want to select an item based on its dimension values and stock availability, filter the product list with those values, select the first item, and copy the number to be used for future tests. If it's a simple master data line such as customer, vendor, or item, it can be created as part of the automation and used in future tests through chaining.

- o Enter the unique identifiers, such as invoice numbers, through the number sequence or by using Microsoft Excel functions such as =TEXT(NOW(),"yyyymmddhhmm"). This function will provide a unique number every minute, which allows you to track when the action happened. This can be used for variables such as product receipt numbers and vendor invoice numbers. These tests continue to work on the same database again and again, without requiring any restoration.
- Always set the **Edit mode** of the environment to **Read** or **Edit** as the first test case because the default option is **Auto**. The **Auto** options always uses the previous setting and can cause unreliable tests.



- Only validate after you filter on a particular transaction instead of generic validation. For example, for the number of records, filter for the transaction number or the transaction date so that the validation excludes all other transactions.
- If you are checking a customer balance or budget check, save the value first and then add your transaction value to validate the expected result instead of validating a fixed expected value.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Troubleshoot the Regression suite automation tool

2/18/2021 • 5 minutes to read • Edit Online

This topic contains information about how to troubleshoot the Regression suite automation tool (RSAT).

Playback logs

To troubleshoot issues that happen during playback of a test case, open the developer error log located at **[RSAT working directory]\[test case ID]\playback\[TestName]Log.txt**. The RSAT working directory is the directory specified in the RSAT settings dialog. Analyze the error message to determine the possible cause of a failure.

> **NOTE**
>
> For RSAT versions prior to 1.210, the log is located at **C:\Users\[YourUserName]\AppData\Roaming\regressionTool\playback\[TestName]Log.txt**.

Generator logs

To troubleshoot errors when generating test execution and parameter files, enable generator logs.

Open the **Microsoft.Dynamics.RegressionSuite.WindowsApp.exe.config** file under the RSAT installation folder (for example, **C:\Program Files (x86)\Regression Suite Automation Tool**), and change the value in the following element from **false** to **true**.

```
<add key="LogGeneration" value="true" />
```

After test execution files are generated, you can find the log file under **[RSAT working directory]\[test case ID]\generatorLogs**

> **NOTE**
>
> For RSAT versions prior to 1.210, the logs are generated under **C:\Users\[Username]\AppData\Roaming\regressionTool\generatorLogs**.

Authentication certificate and installation

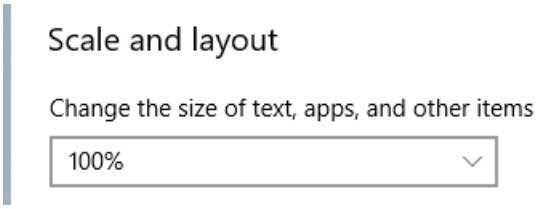
- The authentication certificate must be created and installed by an administrator on the same computer where RSAT is installed. If it is not created by an admin, you will encounter the following error message when you try to run a test case.

```
Cannot access Finance and Operations environment. Verify your settings and make sure the environment is
available.
```

- If you have used a previous version of RSAT on the same computer, close it and uninstall it before installing a new version.

Screen resolution when using Internet Explorer

If you have selected Internet Explorer as your browser, your desktop resolution should be set to 100% to run the tests successfully. To change the settings, use Windows **Display settings > Scale and layout**, as shown in the following image:



Test playback errors

SOAP or HTTP request errors

If you are testing against a Standard Acceptance Test Sandbox environment (Tier2) or any other multi-box environment, and you may receive one of the following errors when you run a test.

```
There was no endpoint listening at
https://<yourURL>soap.sandbox.operations.dynamics.com/Services/AxUserManagement/Service.svc/ws2007FedHttp
that could accept the message. This is often caused by an incorrect address or SOAP action…

An error occurred while making the HTTP request to
<Hostname>/Services/AxUserManagement/Service.svc/ws2007FedHttp. This could be due to the fact that the
server certificate is not configured properly with HTTP.SYS in the HTTPS case. This could also be caused by
a mismatch of the security binding between the client and the server.
```

Assuming that you have specified the correct SOAP hostname in the RSAT settings dialog box, run the following PowerShell scripts on your client computer where the test tool is installed.

```
Set-ItemProperty HKLM:\SOFTWARE\Microsoft\.NETFramework\v4.0.30319 -Name SchUseStrongCrypto -Value 1 -Type
dword -Force -Confirm:$false

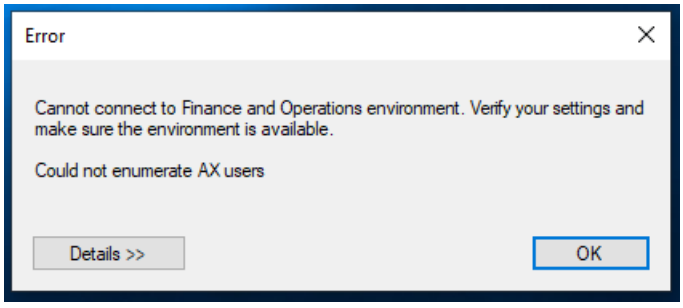
if ((Test-Path HKLM:\SOFTWARE\Wow6432Node\Microsoft\.NETFramework\v4.0.30319))  { Set-ItemProperty
HKLM:\SOFTWARE\Wow6432Node\Microsoft\.NETFramework\v4.0.30319 -Name SchUseStrongCrypto -Value 1 -Type dword
-Force -Confirm:$false}
```

You can also manually set the registry keys.

Cannot enumerate AX users error

You may receive the following error when running a test case, or the error details may contain the following messages.

```
<Message>The type initializer for
'MS.Dynamics.TestTools.CloudCommonTestUtilities.Authentication.UserManagement' threw an exception.</Message>
<Message>Could not enumerate AX users</Message>  (InnerError)`
```



To resolve this error, verify the **Admin user name** specified in the RSAT settings dialog box. The **Admin user**

name must be the email address of a user that belongs to the System Administrator role on the Finance and Operations test environment that RSAT is connecting to. The user account (e-mail address) must also belong to the same tenant as the test environment. For example, if your test environment's tenant is **contoso.com**, the admin user must end with **@constoso.com**.

Unsecured fault exception

If a test case inconsistently fails with the following error, this usually indicates an incomplete configuration of the authentication thumbprints on the AOS virtual machines.

```
<Message>An unsecured or incorrectly secured fault was received from the other party. See the inner
FaultException for the fault code and detail.</Message>
<Message>At least one security token in the message could not be validated.</Message>
```

Typically, this error happens when the test environment has not been configured to trust the certificate that RSAT is using for authentication. (The certificate is identified by the thumbprint specified in your RSAT settings.) For example, the thumbprint could be missing in the wif.config file on the AOS virtual machine of the test environment. If you are running against a standard acceptance test environment (Tier 2 or higher), you might not have configured the authentication thumbprint on all of the AOS virtual machines. Make sure you properly add the thumbprint to the **wif.config** file on all of the AOS machines. For more information, see Configure the test environment to trust the connection.

We have also seen this error when there is a mismatch between the UTC time between the client computer (where RSAT is installed) and the Finance and Operations environment. This is a rare case and only happens if your administrator has incorrectly configured the UTC time. The client computer and Finance and Operations environment can be on different time zones; however, the UTC time on both environments must match because the authentication mechanism relies on this comparison.

Google Chrome Browser

The Google Chrome browser may not work with the Regression suite automation tool due to your Active Directory security settings. In this case, change your RSAT settings to use the new Microsoft Edge or Internet Explorer.

Validating blank dates

If your test case requires validation that a certain control of type Date/Time is blank, you can insert the following value into the Excel cell corresponding to this control: "01/01/1900".

Azure DevOps connectivity

You might you see this error when you select the desired Azure DevOps project in RSAT settings: "The structure path <iteration path> is not valid. Verify your settings and try again". To resolve this error, open the project in Azure DevOps and navigate to the Test Plans. Verify the iteration path defined for each test plan. If the iteration path is similar to what is shown in the error, remove the existing iteration path and add a new one for the test plan and save.

Requirements for publishing apps on AppSource

2/18/2021 • 2 minutes to read • Edit Online

Lifecycle Services (LCS) solution packages for Microsoft AppSource are partner-designed and developed solutions that can be automatically deployed on Microsoft Azure to deliver an end-to-end solution, using industry and vertical-specific content.

This topic points to resources that will help you understand the requirements for creating LCS solutions for Microsoft AppSource. The requirements for creating an LCS solution package fall into the following groups.

- App validation
- Code migration
- Database backup and Data packages
- Business process models
- Methodologies
- Marketing

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Add methodologies to solutions

2/18/2021 • 2 minutes to read • Edit Online

This topic explains how to create and modify methodologies in Microsoft Dynamics Lifecycle Services (LCS). It also provides information about the requirements for methodologies.

Methodology requirements

The first two phases of your methodology are the Learn and Consume phases from the LCS Solutions Consumption methodology.

During the Learn phase, your product description must be aligned with your business process library, the descriptions or summaries in your marketing material, and the functionality that is supported in the current version of your solution. The Learn phase includes the following tasks:

- **Product description** – Take advantage of your marketing description by adding it to the LCS methodology.
- **Get an overview** – Include content for your solution, such as information about the features and architecture.
- **How to get help** – Include numbers, contacts, or direct links to your company's website, for people who help build and maintain the solution.

The Consume phase is **optional**. It includes tasks that are required in order to complete Conference room pilot 1 (CRP1).

After the Learn and Consume phases, you can add any other steps that are required in order to implement your solution. You can either modify the phases and tasks so that they are aligned with your solution, or use your company's implementation methodology.

Create a new methodology

1. On the LCS home page, select the **Manage methodologies** tile.
2. Select the **New methodology** button (the plus sign [+]).
3. Set values for the fields, and then select **Confirm**.
4. Create phases and tasks.
5. Add any linked tools and resources that are required.

Edit a methodology

1. On the LCS home page, select the **Manage methodologies** tile.
2. Select the methodology to edit.
3. Select the **Edit methodology** button (the pencil symbol), and edit the methodology.

Edit a project's methodology

Follow these steps to edit a methodology in a specific project only.

1. On the project's home page, select the ellipsis (...) button above the project phases, and then select **Edit methodology**.
2. Edit the phases and tasks.
3. Edit the linked tools and resources, if any changes are required.

Change a project's methodology

1. On the project's home page, click the ellipsis (...) button above the project phases, and then click **Change methodology**.
2. In the dialog box that appears, set the **Do you want to keep the existing phases and tasks?** option. If you set this option to **Yes**, the phases and tasks from the methodology that you select in step 3 are added to the end of the current methodology. If you set the option to **No**, the current methodology is replaced with the methodology that you select in step 3.
3. Select a methodology to use with your project.
4. Select **Confirm**.

Additional resources

Requirements for publishing apps on AppSource

Set up Business process modeler libraries

2/18/2021 • 5 minutes to read • Edit Online

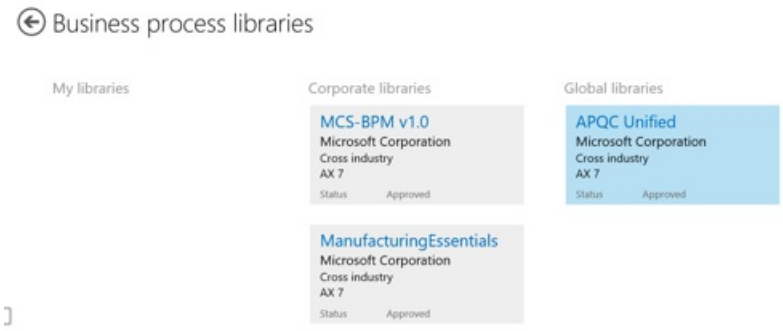
This topic explains how to create and work with Business process modeler (BPM) libraries.

Create a business process library

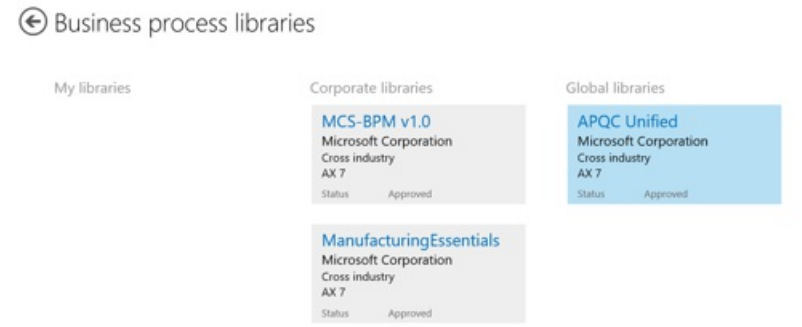
There are two ways to create a BPM library. You can create a new library that has no lines or task recordings, or you can copy an existing library.

1. In Microsoft Dynamics Lifecycle Services (LCS), open a project, scroll to the right until you see the **More tools** section, and then select the **Business process modeler** tile. The **Business process libraries** page that appears has three sections, one for each type of library:

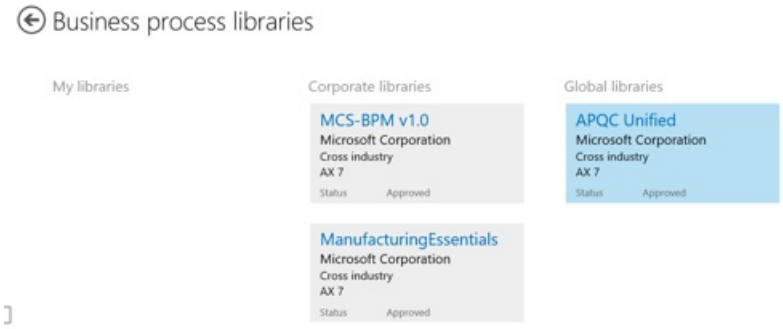
   - **My libraries** – Business processes that users have created or added.
   - **Corporate libraries** – Custom business processes that someone in your organization has uploaded.
   - **Global libraries** – Cross-industry standard business processes.



2. To create a new library, right-click any library, and then, in the lower-left corner of the window, select **Create**. To copy an existing library, right-click that library, and then, in the lower-left corner of the window, select **Copy**.



- **Global libraries** – Cross-industry standard business processes.



3. Enter a name for the library.

The library that you created now appears under **My libraries**.

Update, publish, or delete a business process library

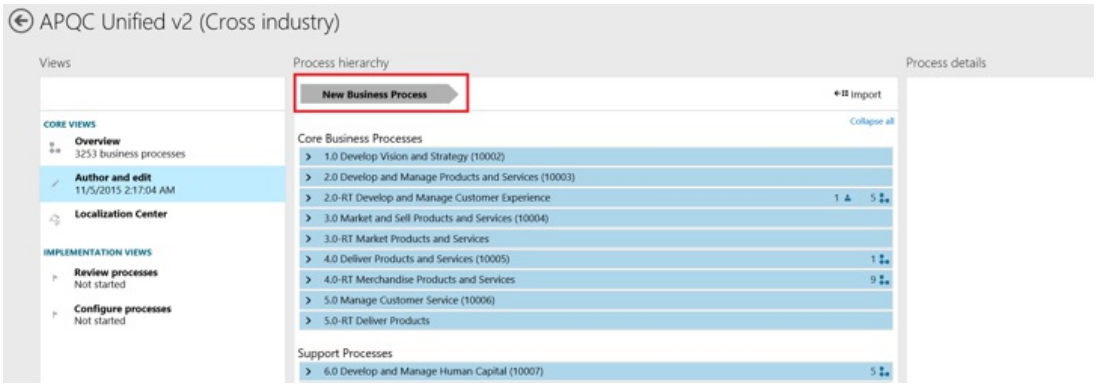
You can change the name or description of a library, publish a library so that other people in your organization can view it, or delete a library.

- On the **Business process libraries** page, right-click the library to update, publish, or delete. Then, in the lower-left corner of the window, select **Edit**, **Publish**, or **Delete**.
  - **Edit** lets you change the name and description of the library.
  - **Publish** creates a copy of the library under **Corporate libraries**. This library will be available to everyone in your organization.
  - **Delete** deletes the library and any information that is stored in it.

Modify a business process library

Follow these steps to change or update the business process lines or hierarchy in your business process library.

1. Select and open the library to update.
2. In the left pane, under **Core views**, select **Author and edit**.
3. Follow one or more of these steps to make the required changes:
   - To rearrange the hierarchy of the business process lines, drag the processes in the hierarchy.
   - To delete a node, select a business process in the center pane, and then select **Delete** in the right pane.
   - To create a new node, drag the **New Business Process** flag at the top of the center pane to the place in the hierarchy where the new node should appear.

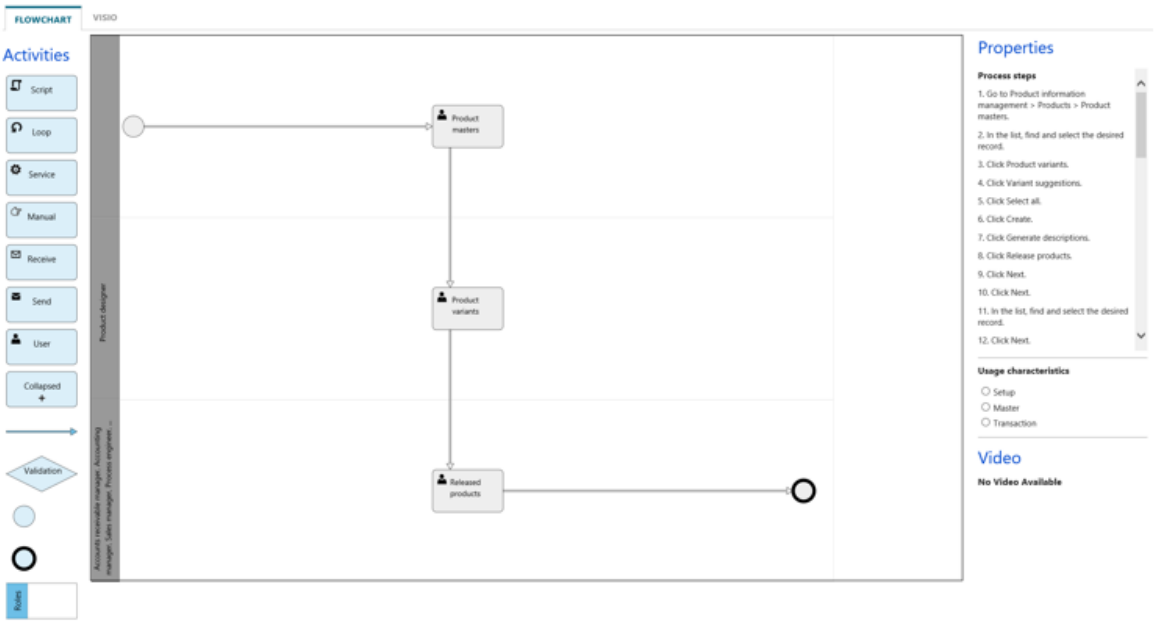


- To import business processes from existing business process libraries, follow these steps:

    a. Select **Import** in the center pane.
    b. On the right side of the page that appears, select the library to import business processes from.
    c. Drag the required business processes into the hierarchy on the left side of the page.

View and modify task recordings in a business process library

1. Open the library that you want to work in.
2. Navigate to the business process line that has a task recording associated with it, and select the link.
3. Drag the tiles under **Activities** to manually modify the flow chart.



4. To upload a Microsoft Visio diagram of the modified flow chart, select the **Visio** tab.

Structure a business process library

There are two sections in business process libraries: **Core Business Processes** and **Support Processes**. The **Core Business Processes** section should include all custom business processes for your solution. All customizations and functionality should be covered in end-to-end scenarios. Task recordings should be created for all processes in this section. Import American Productivity & Quality Center (APQC) processes that are relevant to your solution into the **Support Processes** section. This section should not include any custom business processes. You don't have to create task recordings for processes in this section. Your business process library should be aligned with the descriptions and summaries in your methodology and your marketing material.

Create a task recording and associate it with a business process

Task recordings should be created in an environment that has your custom data and customizations. For reference information about Task recorder, see Task recorder resources.

Create a task recording

1. In your application, select the **Settings** button in the upper-right corner, and then select **Task recorder**.
2. Select **Create recording**.
3. Enter a name and description for the recording.
4. Perform the task that you want to record.

5. When you've completed the task, select **Stop**.
6. If you want to upload your new task recording to LCS, continue to the next section. To save your recording to your local computer or convert the recording to a Microsoft Word document, select the appropriate option.

Associate a task recording with a business process

1. Select **Save to LCS**.
2. Select a business process library.
3. Select the business process line to associate with the recording.
4. Select **OK**.

You can now view the task recording in the business process library in LCS.

Set up and use task guides

Task recordings can be played as task guides. Task guides are used to guide users through the steps for completing business processes. Before you set up and use task guides, create task recordings, and save them to a business process library in LCS.

Set up task guides

1. In your application, select **System administration** > **Setup** > **System parameters**.
2. On the **Help** tab, select the LCS project where the business process library that you want to work with is stored.
3. Select the business process libraries that have the task recordings that you want to play as task guides.
4. Adjust the order of the business process libraries as you require. The order of the business process libraries determines the order that the task guides appear in. Therefore, the task guides for the first business process library appear first when a user uses the task guide functionality.

Use task guides

1. In your application, open the page where you want to run the task guide.
2. In the upper-right corner, select the **Settings** button, and then select **Task recorder**.
3. Select a task guide.
4. Select **Start Task guide**.
5. Follow the steps in the task guide. If you select **Unlock**, you can work without following the task guide.
6. When you've finished, select **Stop Task guide**.

Additional resources

Requirements for publishing apps on AppSource](lcs-solutions-app-source.md)

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Migrate code for Finance and Operations apps solutions

2/18/2021 • 2 minutes to read • Edit Online

To complete your solution package, the first step is to upgrade your code by using the best practices in **Migrate and Create Finance and Operations Apps Solutions** in Microsoft Dynamics Lifecycle Services (LCS). After that step is completed, you must run the Customization Analysis Report (CAR). This report analyzes your customization and extension models, and runs a predefined set of best practice rules.

To generate the CAR, run the following command on a development environment.

```
xppbp.exe -metadata=<local packages folder> -all -model=<ModelName> -xmlLog=C:\BPCheckLogcd.xml -module=
<PackageName> -car=<reportlocation>
```

Here is an example of this command.

```
xppbp.exe -metadata=C:\Packages -all -model=MyAppSuiteCustomizations -xmlLog=C:\temp\BPCheckLogcd.xml -
module=ApplicationSuite -car=c:\temp\CAReport.xlsx
```

The xppbp.exe file is located in c:\packages\bin or I:\AosService\Packages\LocalDirectory\bin. You must resolve any warnings or errors that appear on the **Issues** tab of the report. You must then submit a copy of the CAR to Microsoft before your validation meeting. For more information, see Customization Analysis Report (CAR). For information about issues and exceptions, see the Customization Analysis Report: Exceptions and known issues post on the Dynamics 365 Community blog.

Extensibility

In Microsoft Dynamics 365 for Finance and Operations version 8.0 (April 2018), all product models are sealed. Therefore, only extension-based customizations are currently supported. For more information about extensibility, see Extensibility.

The first step in completing your solution package is to upgrade your code using the best practices in **Migrate and Create Finance and Operations Apps Solutions** in LCS. After this step is complete, you must run the Customization Analysis report. This report analyzes your customization and extension models, and runs a predefined set of best practice rules.

To generate the Customization Analysis report (CAR), run the following command on a development environment.

```
xppbp.exe -metadata=<local packages folder> -all -model=<ModelName> -xmlLog=C:\BPCheckLogcd.xml -module=
<PackageName> -car=<reportlocation>
```

Here's an example of how this command might look.

```
xppbp.exe -metadata=C:\Packages -all -model=MyAppSuiteCustomizations -xmlLog=C:\temp\BPCheckLogcd.xml -
module=ApplicationSuite -car=c:\temp\CAReport.xlsx
```

The xppbp.exe file is located in *c:\packages\bin* or *I:\AosService\Packages\LocalDirectory\bin)*. Any warnings or

errors that appear on the **Issues** tab of the report must be resolved. A copy of the CAR report must be submitted to Microsoft prior to your validation meeting. For more information, see Customization Analysis Report (CAR) or refer to the Dynamics Community blog for issues and exceptions.

Additional resources

Requirements for publishing apps on AppSource

Develop and customize home page

Validate applications for Finance and Operations apps

2/18/2021 • 4 minutes to read • Edit Online

This topic provides information about the requirements that are used to verify that custom code meets Microsoft guidelines, and that a solution package can be successfully bundled and delivered in a Finance and Operations apps environment.

Microsoft requires specific reviews in order to validate the following requirements:

- A partner's custom code meets Microsoft guidelines.
- A Microsoft Dynamics Lifecycle Services (LCS) solution package can be successfully bundled and delivered.
- Core independent software vendor (ISV) business scenarios can be transacted.

Currently, partners must demonstrate that these requirements have been met by doing test deployments and then sharing the results with Microsoft. No code will be deployed on a customer environment that Microsoft hasn't validated. Partners must complete the following curation artifacts and tests:

- Code analysis report (CAR)
- Business process modeler (BPM)/test scripts
- Business database backup
- Project name and description
- Data packages
- Methodology
- Binaries (optional)
- Deployable packages
- Models (code and tests)
- Marketing content

Curation meeting

The following table describes the steps that must be completed before the validation meeting.

PHASE	STEP	ACTIVITY	PROCESS STEPS	SUCCESS CRITERIA
1	1	Validate code.	Run all customer model files by using the CAR tool, and then generate the report.	Successfully create a CAR without any localization, accessibility, performance, or security issues.
1	2	Verify user experience (UX) guidelines.	Follow UX guidelines to implement the workspace correctly.	Reference best practice information in the Migrate and Create methodology section of LCS.

PHASE	STEP	ACTIVITY	PROCESS STEPS	SUCCESS CRITERIA
2	3	Validate the solution package in LCS.	Create a solution package in LCS that includes all the required artifacts.	A solution package that has all the required artifacts has been published in LCS, and a globally unique identifier (GUID) has been created for the solution.
2	4	Deploy an environment.	Deploy a standard environment that has partner code, based on the package contents (code, binaries, and configuration).	Successfully deploy at least one Finance and Operations environment without any errors. The environment configuration (including components and the configuration) is the same as the partner's reference environment. A user can successfully sign in to this environment without any errors.
2	5	Configure and deploy data.	Deploy partner-supplied data in the environment without any errors.	Demonstrate that partner-supplied master and reference data was successfully pushed into the environment without any errors.
2	6	Do a sanity check.	After data has been loaded into the environment, users should be able to complete business transactions (as defined in the scope of the solution).	Users can sign in to the data-loaded environment without any errors. Business transactions can be completed, as defined in the package scope, without any errors.

Detailed curation requirements

The following table provides more information about each curation requirement.

REQUIREMENT	DESCRIPTION
CAR	All major issues that the CAR highlights should be addressed after you upgrade. The CAR must be submitted to Microsoft before the validation meeting.
BPM/test scripts	All task recordings should be completed for the industry vertical that the solution package is designed for and should include end-to-end scenarios.

REQUIREMENT	DESCRIPTION
Business database backup	A business database of your upgraded environment and best practice configurations should be loaded into the Asset library in LCS.
Project name and description	The project name and description should be incorporated into the beginning of the implementation methodology for the solution package.
Data packages	All data packages should be loaded into LCS before the validation meeting. Create data entities for any additional custom fields or tables for your custom functional features. You should be able to modify the data packages and load them into an empty environment, and then consume the data packages in Data Management Framework.
Methodology	The methodology should incorporate an overview of the product. A guided experience to Conference Room Pilot 1 (CRP1) and any other implementation methodology that is specifically tailored to your solution are optional.
Binaries (optional)	Incorporate any required binary files.
Deployable packages	Incorporate the deployable packages that are required in order to bring your custom features and functionality into your environment.
Models (code and tests)	Incorporate any model files that are required for your solution.
Marketing content	Add your marketing content, such as logos, descriptions, and screen shots of your solution package. The solution logo should be app-specific and should not include your company name. The description should be aligned with your custom business processes.

Update and maintenance requirements

If you have a curated solution that is published on AppSource, you must keep the solution up to date. After each major Spring and Fall release, you will have eight weeks to upgrade your code. You must update and test the following artifacts:

- CAR
- Models (code and tests)
- Deployable packages
- BPM/test scripts
- Data packages
- Business database backup

Maintenance process steps

PHASE	NUMBER	ACTIVITY	PROCESS STEPS	SUCCESS CRITERIA

PHASE	NUMBER	ACTIVITY	PROCESS STEPS	SUCCESS CRITERIA
1	1	Validate customer code.	Run all customer model files by using the CAR tool, and generate the report.	Successfully create a CAR without any localization, accessibility, performance, or security issues. All major issues that the CAR highlights should be addressed after you've upgraded to the latest major release. The CAR must be submitted to Microsoft within eight weeks after each major Spring and Fall release.

Additional resources

Requirements for publishing apps on AppSource

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

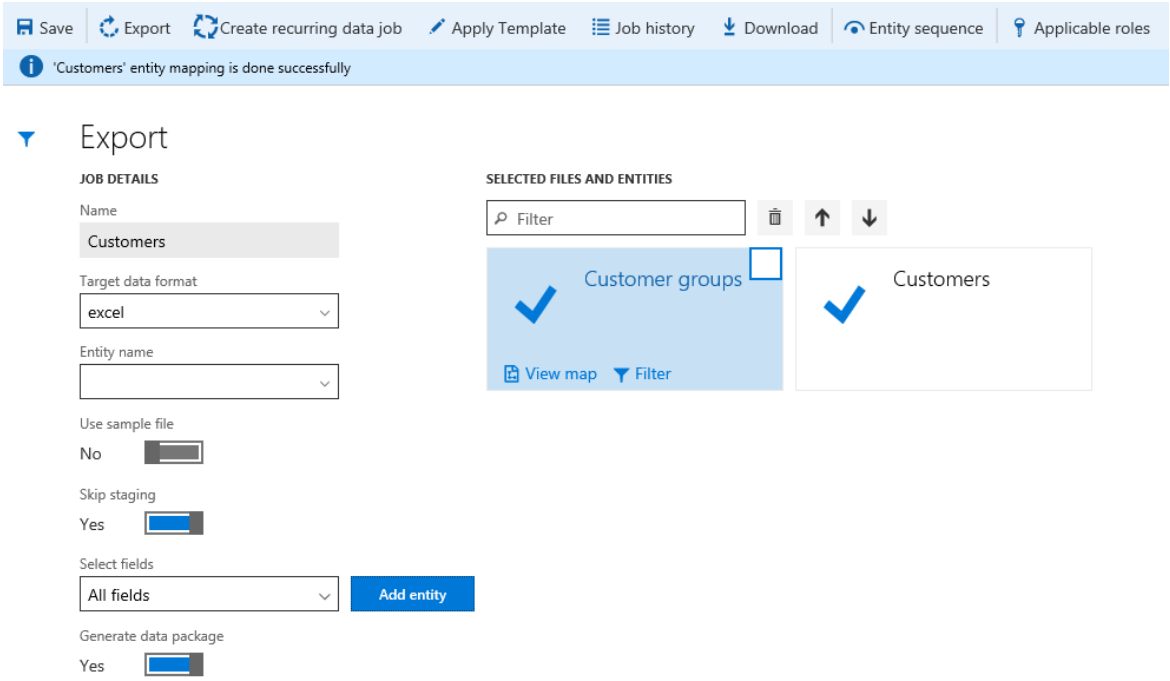
Process and consume data packages in Dynamics 365 Finance and Operations apps solution

2/18/2021 • 10 minutes to read • Edit Online

A data package for a Dynamics 365 Finance and Operations app can consist of one or many data entities. A typical data package consists of a group of entities for a specific task, process, or function. For example, the data entities that are required for general ledger setup might be part of one data package. The format of a data package is a compressed file that contains a package manifest, a package header, and any additional files for the data entities that are included.

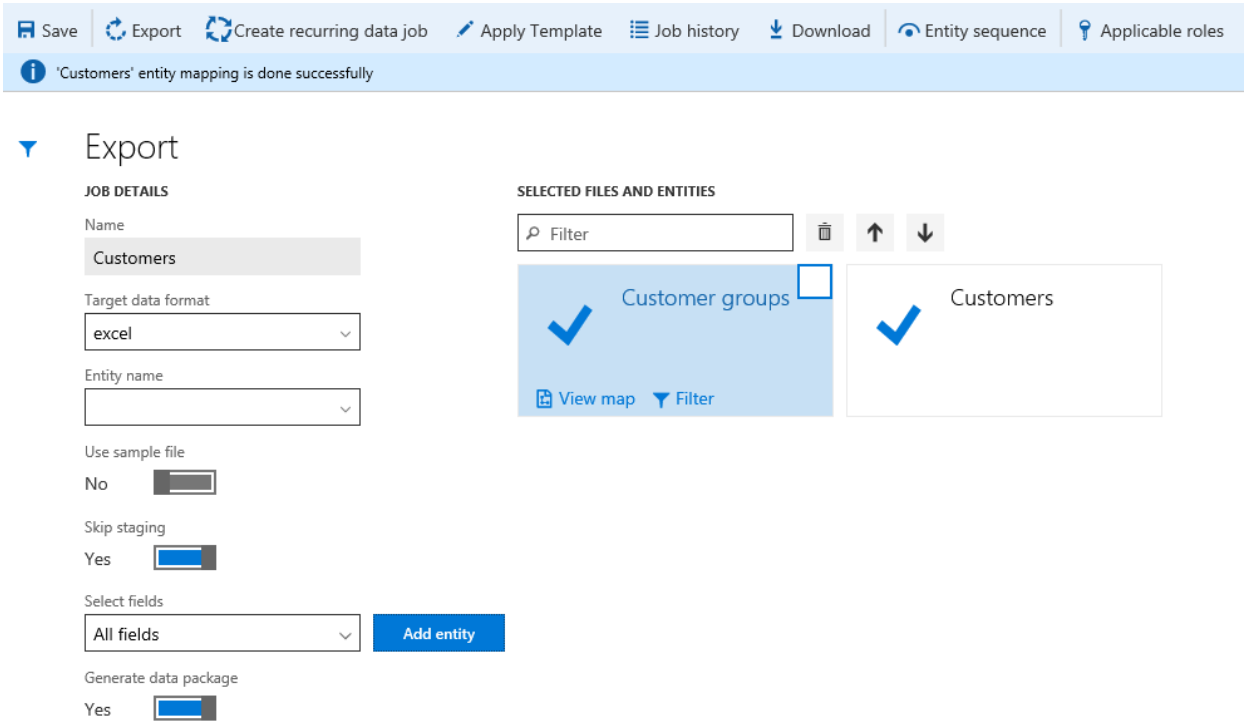
Before you create your data package, plan out what it should include. In this way, you help guarantee that the correct entities, entity sequence, and fields are included. You create a data package by using the **Data management** workspace in your application. Follow these steps to create a data package.

1. In the application, select **System administration** > **Workspaces** > **Data Management IT**.
2. Select the **Export** tile, and then, in the **Name** field, enter **Data project**.
3. In the **Target data format** field, select the format for the export. The data formats that are available include comma-separated value (CSV) format and Microsoft Excel format.
4. In the **Entity name** field, enter or select an entity. You can add multiple entities, but you must add each entity separately.
5. In the **Select fields** field, select a field setting, and then click **Add entity** to add the entity to the project.
6. Repeat steps 4 and 5 to add more entities to the project. **Note:** As you add each entity to the project, a tile appears that contains the entity's name and two buttons: **View map** and **Filter**. To automatically create a data package when you export the data project, set the **Generate data package** option to **Yes**. If you don't set this option, you can create a data package at the time of export.
7. On the Action Pane, click **Export**.
8. Click **Download**. The package is saved to the **Downloads** folder of the computer where the browser session is running. When you work with data packages, you must plan for and consider any prerequisites for the entities that will be included in the packages. For example, the customer groups are required in order to create customers. Therefore, you should either import the customer groups into a package before you import customers, or sequence customer groups within a data package that will be completed before customers are imported. For example, in the following illustration, sequencing is set within the data package. As you can see, the Customer groups entity and Customers entity are part of the Customers data project.

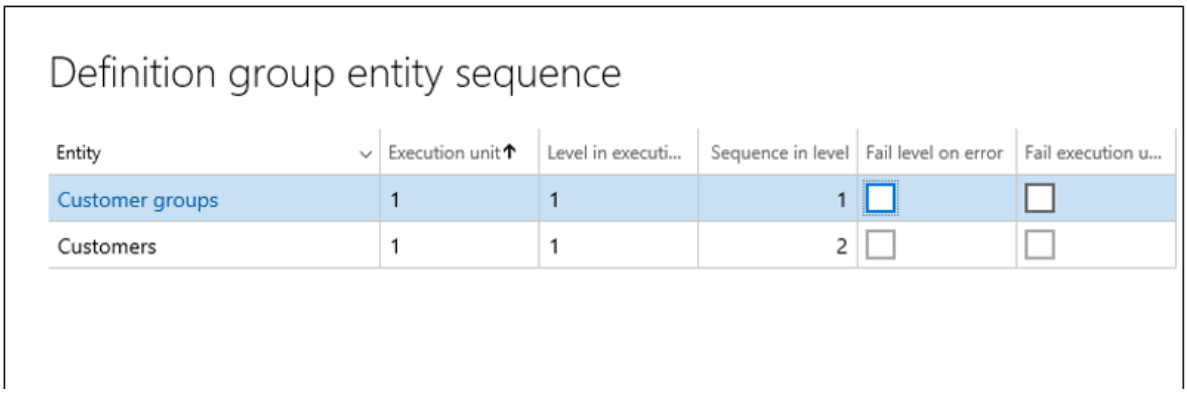


To automatically create a data package when you export the data project, set the **Generate data package** option to **Yes**. If you don't set this option, you can create a data package at the time of export.

9. On the Action Pane, select **Export**.
10. Select **Download**. The package is saved to the **Downloads** folder of the computer where the browser session is running. When you work with data packages, you must plan for and consider any prerequisites for the entities that will be included in the packages. For example, customer groups are required in order to create customers. Therefore, you should either import the customer groups into a package before you import customers, or sequence customer groups within a data package that will be completed before customers are imported. For example, in the following illustration, sequencing is set in the data package. As you can see, the Customer groups entity and Customers entity are part of the Customers data project.



11. On the Action Pane, select **Entity sequence** to open the **Definition group entity sequence** page. Based on the current setup, the Customer groups entity and Customers entity are run at the same level. However, this sequence might not be ideal.



12. To create a better sequence, select the **Customers** entity, and then update the value of the **Execution unit** field from **1** to **2**. This change helps guarantee that customer groups are imported before the Customers entity is run.

Microsoft Dynamics Lifecycle Services (LCS) contains multiple base data packages that you can use to reduce the implementation time. These packages contain the elements that are required in each module/area in order to meet the minimum requirements. For advanced business processes, you might have to add more entities to the list of packages. The data packages that Microsoft publishes on LCS use a numbering sequence that is based on the module, data type, and sequence. Here is an example:

- Module/area number

Module	Module Reference
System administration	01
General ledger	03
Public Sector	04
HRM	05
Accounts payable	10
Accounts receivable	11
Budgeting	12
Cash and bank management	13
Compliance and internal controls	14
Cost accounting	15
Fixed assets	16
Inventory management	19
Master planning	20
Organization administration	21
Payroll	22
Procurement and sourcing	23
Product information management	24
Production control	25
Project management and accounting	26
Retail	27
Sales and marketing	28
Service management	29
Trade allowance management	31
Transportation management	32
Travel and expense	33
Warehouse management	34

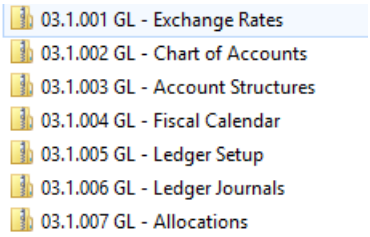
- Data type numbering

Data Type	Data Type Reference
Setup	1
Master	4
Transaction	8

- Numbering format

Numbering Format	
Module # . Data Type Reference . 001 (Sequence Number)	
01.1.001	System / Setup Data / Seq 001
01.1.002	System / Setup Data / Seq 002
01.4.001	System / Master Data / Seq 001
01.4.002	System / Master Data / Seq 002
03.1.001	General Ledger / Setup Data / Seq 001

The names of data packages include the numbering format, which is followed by the module abbreviation and then a description. For example, the following illustration shows the General ledger data packages.



Process data packages

A process data package (PDP) consolidates Data import/export framework (DIXF) data packages into a unified bundle. The PDP is then used to configure a business process or a group of business processes in one business process library. Together, DIXF data packages, dependencies between those packages, and business processes that require the packages for their configuration make up a PDP. This section describes how to create a PDP for your LCS solution package. To create a PDP, you must have the following items and knowledge:

- An implementation business process library for the solution in your working LCS project.
- DIXF data packages that have configurations that follow best practices. These packages should include master and reference data for the whole solution.
- An understanding of the dependencies between the data in the packages.
- An understanding of the data dependencies between the data packages.

Follow these steps to create a PDP.

1. In LCS, select the **Asset library** tile.
2. Select **Data package** as the asset type, and then select the plus sign (+) to add a data package.

Follow these steps to consolidate the data packages that you uploaded to LCS into a single PDP.

1. In LCS, select the **Asset library** tile.
2. Select **Process data package** as the asset type, and then select the plus sign (+) to add a PDP.
3. Enter a name and description for the PDP, and then select **Confirm**.
4. For step 1, "Add business process library," select the implementation business process library that your solution is built for, and then select **Continue**.
5. For step 2, "Add data packages," select **Edit**, and then add the data packages that are required in order to configure the system and enable the business process library transactions to be run. When you've finished adding data packages, select **Select**. The order in which data packages in the PDP are loaded into the system might be important. For example, before a chart of accounts (COA) can be loaded, the legal entity setup and currencies are required. Those data entities are in 01.1.002 System Setup. Therefore, 03.1.1001 COA Setup depends on 01.1.002 System Setup. In the next step, you must make a note of this dependency.
6. For step 3, "Add data package dependencies," select a data package, and then select **Edit**. Select the dependent data packages that must be loaded into your target environment before the data package that you selected, and then select **Select**.
7. For step 4, "Associate business process to data packages," select the business process that should be used to configure the process nodes.

Consume a PDP

> **IMPORTANT**
>
> For the PDP consumption requirement, you have the option to consume data packages directly via the Data Management Framework in your environment. However, note that only the consumption of data packages via the LCS PDP tool is optional. You must still create the PDP and upload it to your Asset library.

The Consume flow lets you review a business process, and apply the configuration and data that are required in order to implement the business process in your environment. To consume PDPs, you must have the following items:

- An implementation business process library for the solution in your working LCS project
- PDPs
- A target environment that includes an existing legal entity

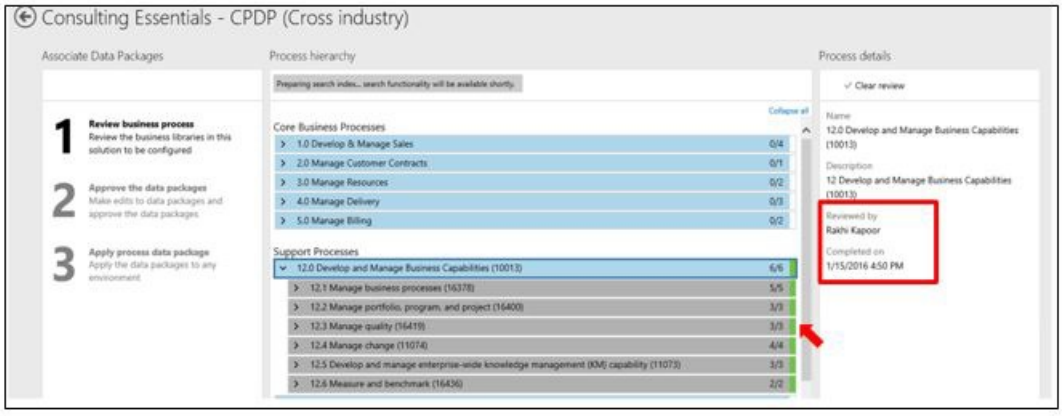
Follow these steps to consume the PDP.

1. In LCS, select the **Asset library** tile.
2. Select **Process data packages** as the asset type, and then select **Consume**.
3. The **Consume process data package** page shows a list of the existing PDP assets. Select the plus sign (**+**) to create a new Consume PDP.
4. Enter a name for the Consume PDP, select the PDP that you created and saved in the Asset library, select a target environment, and then select **Create**.
5. The Consume PDP that you created appears in the list of PDP assets. Select the package.

> **NOTE**
>
> The asset that you created can be consumed only in the target environment that was linked to it.

Review and approve BPMs

- For step 1, "Review business process," review the business process models (BPMs), and then select **Mark as reviewed**. The review status is updated for all dependent processes, and a green bar appears to the right of them. The **Reviewed by** and **Completed on** fields are also updated for each business process.



Review and approve data packages that are associated with a BPM

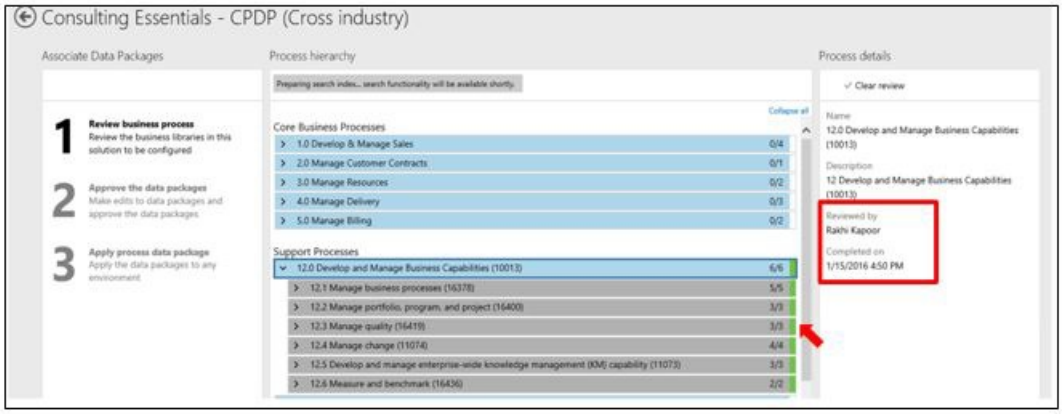
1. In the BPM library that you just reviewed, in the left pane, select **2 Approve the data packages**.
2. Select the business process that is associated with the data packages. The data packages appear in the right pane, under **Process details**. In the right pane, select **Review and Approve**.

3. On the **Consume process data package** page, select a package, and then select **Download** to download the data package. Save the package locally. You can then update the data files in the data package with data that is specific to your target environment. When you've finished updating the data files, select **Upload data package** to upload your changes.
4. After the data package has been updated and completed, select **Approve**. The status is changed to **Approved**. You can approve as many data packages as you require.
5. Select the **Back** button, select **2 Approve the data packages** again, and then select the business process. You should see the **Approved** status in the right pane, under **Process details** > **Dependent packages**.

Apply a process data package

1. In the BPM library, in the left pane, select **3 Apply process data package**.
2. Select the business process, and then, in the right pane, select **Apply Data Packages**.
3. On the **Consume process data package** page, select a package, and then select **Apply**.
4. Select the destination company in the target environment that is linked to the PDP, and then select **Apply**.

- For step 1, "Review business process," review the business process models (BPMs), and then click **Mark as reviewed**. The review status is updated for all dependent processes, and a green bar appears to the right of them. The **Reviewed by** and **Completed on** fields are also updated for each business process.



Review and approve data packages that are associated with a BPM

1. In the BPM library that you just reviewed, click **2 Approve the data packages** in the left pane.
2. Select the business process that is associated with the data packages. The data packages appear in the right pane, under **Process details**. Click **Review and Approve** in the right pane.
3. On the **Consume process data package** page, select a package, and then click **Download** to download the data package. Save the package locally. You can then update the data files in the data package with data that is specific to your target environment. When you've finished updating the data files, click **Upload data package** to upload your changes.
4. After the data package has been updated and completed, click **Approve**. The status is changed to **Approved**. You can approve as many data packages as necessary.
5. Click the **Back** button, click **2 Approve the data packages** again, and then select the business process. You should see the **Approved** status in the right pane, under **Process details** > **Dependent packages**.

Apply a process data package

1. In the BPM library, click **3 Apply process data package** in the left pane.
2. Select the business process, and then, in the right pane, click **Apply Data Packages**.
3. On the **Consume process data package** page, select a package, and then click **Apply**.
4. Select the destination company in the target environment that is linked to the PDP, and then click **Apply**.

View the data package history

- On the **Consume process data package** page, select a package, and then select **History**. You can review the status of the data package. The available information includes the target environment, company, package

name, start and end times, status by data entity, and overall status of the data package. To see the details of any errors that occurred, you can sign in to the target environment.

Additional resources

Requirements for publishing apps on AppSource

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

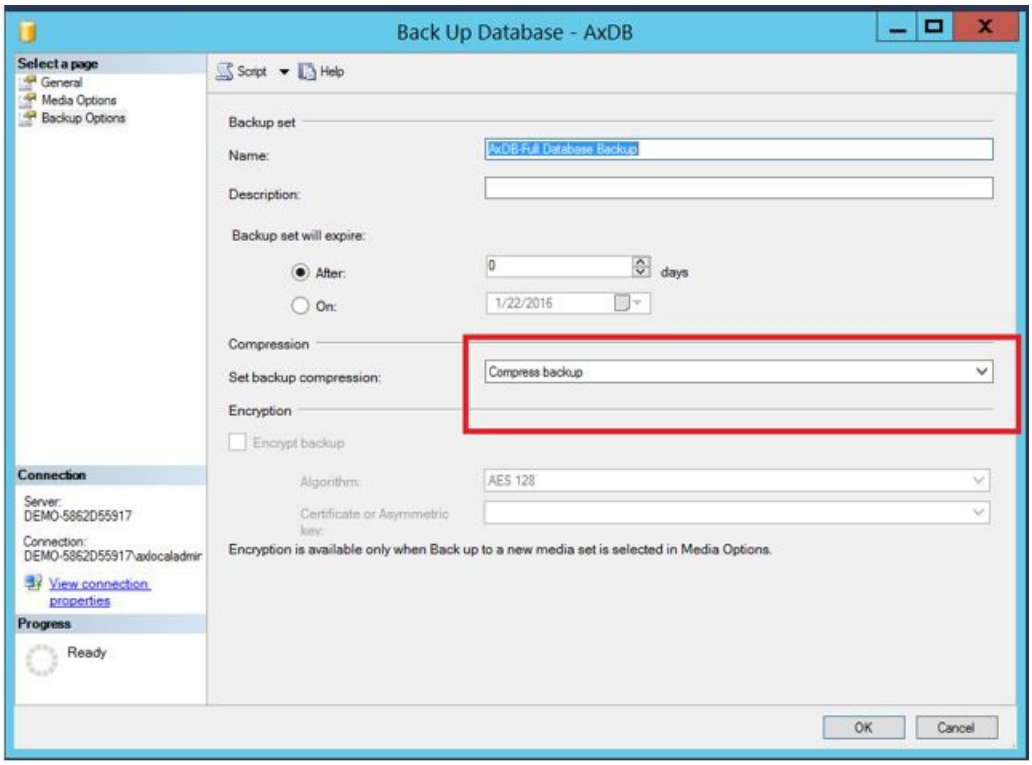
Back up the databases for Finance and Operations apps

2/18/2021 • 2 minutes to read • Edit Online

A backup of theFinance and Operations apps database is required for your Microsoft Dynamics Lifecycle Services (LCS) solution package. When you back up the database, you must include the master, reference, and transactional data that is specific to your solution and industry. This data will be used for your pre-sales demo deployments.

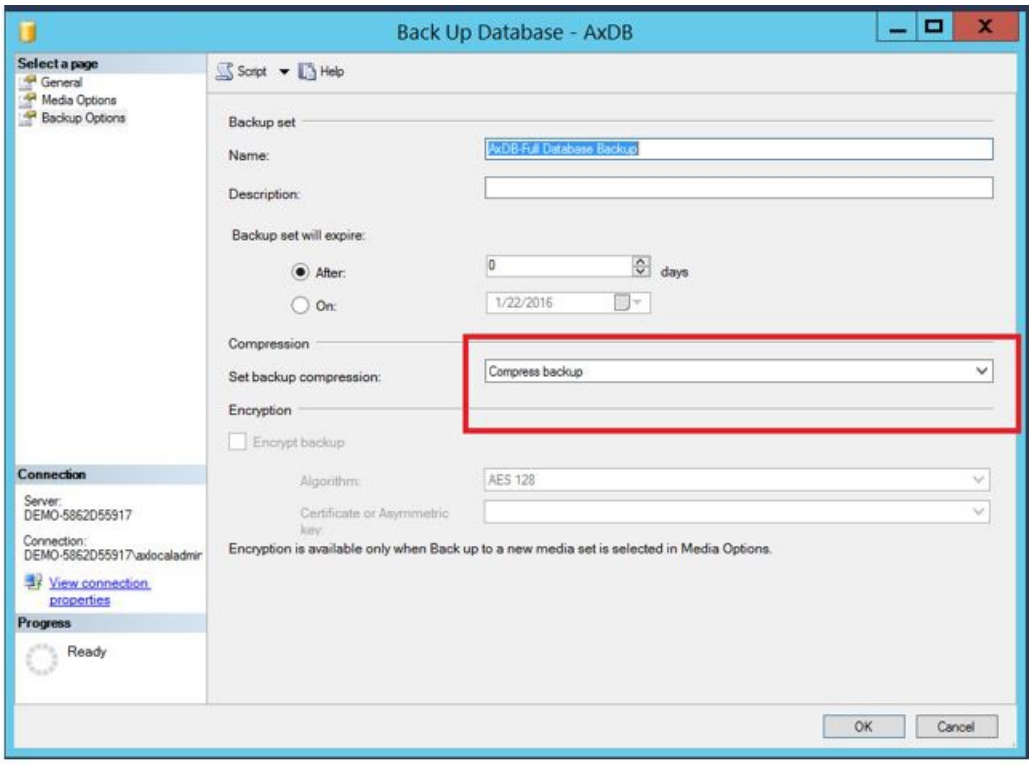
On demo or development environments, the database is typically named AXDBRain. Your database backup should be no larger than 15 gigabytes (GB). Otherwise, a time-out error might occur when you try to upload the database to the Asset library in LCS.

To compress your database backup, in Microsoft SQL Server Management Studio, on the **Back Up Database** page, in the **Set backup compression** field, select **Compress backup**.



On demo or development environments, the database is typically called AXDBRain. Your database backup should be no larger than 15 gigabyte (GB). If your database is larger, a timeout error may occur when you try to upload the database to the Asset library in Lifecycle Services (LCS).

To compress your database backup, in SQL Server Management Studio, on the **Back Up Database** page, in the **Set backup compression** field, select **Compress backup**.



Additional resources

Requirements for publishing apps on AppSource

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Globalization resources

2/18/2021 • 2 minutes to read • Edit Online

Local and regional deployments

If your government regulations require data to be stored differently or serviced differently than is required for other countries/regions, there might be country/region requirements you must consider during deployment. Consider the following resources that might be relevant to you:

Finance and Operations apps operated by 21Vianet in China

Localization and regulatory features

Finance and Operations apps include functionality for the country/regions documented in the Product localization and translation availability guide. This functionality is enabled based on the primary address of the active legal entity.

This topic includes lists of resources that can help you do the following:

- Learn more about developing country/region-specific solutions.
- Get country/region specific updates.
- Submit and review regulatory alerts.
- Learn how to use country/region specific functionality.

Developing localized solutions

The following resources provides guidance and information that can help developers and ISVs who are creating country/region-specific customizations or are creating a solution for a country that Microsoft does not support.

- Separation of localization models
- Apply country/region context
- Regulatory certification information in feature titles
- Classification of localization features
- Country Codes - ISO 3166

Regulatory updates and communication

The following resources provide information about planned and new localization features.

Regulatory updates

- Regulatory updates
- Localization portal (Updated weekly)
- Issue search in Lifecycle Services (LCS) (Updated daily)

Communication and alerts

- Regulatory watch and communication of regulatory updates
- Submit alerts about country/region-specific regulatory features

Dynamics 365 release plans

The Dynamics 365 release plans provide descriptions of new and enhanced capabilities that are planned for Dynamics 365 business applications and application platforms.

Finance and Operations apps what's new

The What's new or changed in Finance and Operations home page lists the features that are included in specific

releases of the Finance and Operations apps.

Electronic reporting

The Electronic reporting (ER) tool allows you to configure formats for electronic documents in accordance with the legal requirements of various countries/regions. ER lets you manage these formats during their lifecycle. For more information, refer to one of the following topics:

- Electronic reporting (ER) overview
- Manage the Electronic reporting (ER) configuration lifecycle
- Create Electronic reporting (ER) configurations
- Extend the list of Electronic reporting (ER) functions
- Electronic reporting (ER) destinations
- Download Electronic reporting configurations from Lifecycle Services
- Import Electronic reporting (ER) configurations
- Configure Electronic reporting (ER) to pull data into Power BI
- Generate electronic documents and update application data by using ER

Task guides

Task guides are available from the product help pane and they provide a guided walk-through of key business processes. You can open a task guide to read the steps of a business process or you can play a task guide to walk through a business process and enter data.

To find task guides, navigate to a page in in the application and click Help. Task guides that use the page are listed in the help pane. You can also use the help pane to search for task guides by title.

To learn more, see Help system.

Country/region specific help content

- Australia
- Austria
- Belgium
- Brazil
- China
- The Czech Republic
- Estonia
- Europe
- France
- Germany
- Hungary
- India
- Italy
- Japan
- Latvia
- Lithuania
- Mexico
- Malaysia
- Netherlands
- Norway
- Poland

- Russia
- Saudi Arabia
- Singapore
- Spain
- Sweden
- Switzerland
- Thailand
- United Kingdom
- United States

Classification of localization features

2/18/2021 • 2 minutes to read • Edit Online

As part of the requirements for LCS solutions for localization and translation, localization features must be classified as either regulatory or competitive in the business process modeler (BPM) library in Microsoft Dynamics Lifecycle Services (LCS). This article explains the difference between these two types of feature and shows how the feature type is used in the title of the feature.

Localization features must be classified as either regulatory or competitive in the Business process modeler (BPM) library in Microsoft Dynamics Lifecycle Services (LCS). The following definitions will help you distinguish the two types of features:

- **Regulatory features** – Organizations that do business in a particular country/region must comply with country/region-specific laws and regulations as they handle their daily business transactions and operations, and meet their legal obligations for activities that are conducted in that country/region. These laws and regulations are enforced, and non-adherence can lead to severe consequences for an organization that does business in that country/region.
- **Competitive features** – This category includes all other features that aren't considered regulatory features according to the preceding definition.

When the BPM library is constructed, the various feature types should be distinguished through the title of the localization solution feature. The label for this title will conform to the following naming convention that indicates the country/region and feature type through prefixes, as shown in the following table.

FEATURE TYPE	FORMAT	EXAMPLE
Regulatory	XX-REG-Feature title	PT-REG-Direct sales Tax report
Competitive	XX-COMP-Feature title	PT-COMP-Country bank payment format

The following table explains the components of the naming convention.

COMPONENT NAME	FORMAT	DESCRIPTION	EXAMPLE
Country/region code	XX	The two-letter ISO country/region code (from the ISO 3166 standard)	PT (= Portugal)
Feature type	REG or COMP	The type of feature	REG
Feature name	Text	A short feature title that describes what the feature is used for	Direct sales tax report

For more information about BPM, see Upload custom business processes to Business process modeler (BPM).

Additional resources

- Globalization resources

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Apply country/region context

2/18/2021 • 3 minutes to read • Edit Online

As part of the requirements for LCS solutions for localization and translation, localization ISV solution providers must implement all country-specific or region-specific functionality so that it can be controlled by country/region context. This article describes how to apply country/region context to meet these requirements. In this article you can find information how you should use country context property and what application objects control user interface elements.

Country/region-specific functionality

You use country/region-specific functionality to help meet the legal, regulatory, and business requirements of individual geographies. A geography is any country or region that is identified by an International Organization for Standardization (ISO) country or region code. The following table highlights the main elements that you use to configure country/region-specific functionality.

ELEMENT	DESCRIPTION
Controlled entity	The controlled entity is a UI element that is hidden or shown, depending on whether its country/region context matches the country/region context of the controlling entity. To enable menus, menu items, and form controls that are based on country/region context to be hidden, a controlled entity includes a **CountryRegionCodes** property on some elements. You use this property to specify the country or region where the element is shown. You find the **CountryRegionCodes** property on the following Application Object Tree (AOT) elements: • Extended data type • Menu and menu items • Enum and enum value • Table and table field • Data entity and data entity field • View and view field • Map and map field • Form control • Tile
Controlling party	The controlling party's role is used to determine whether country/region-specific functionality or UI elements are enabled. The controlling party is defined by the Organization model. Examples include legal entity, customer, vendor, bank, or worker. By default, the legal entity is used as the controlling party. If the country/region context of the controlling party matches the country/region context of the controlled entity, the functionality or UI elements are enabled. You set the country/region context of the controlling party. Any controlled entities that have a matching country/region context are shown.

Using the CountryRegionCodes property

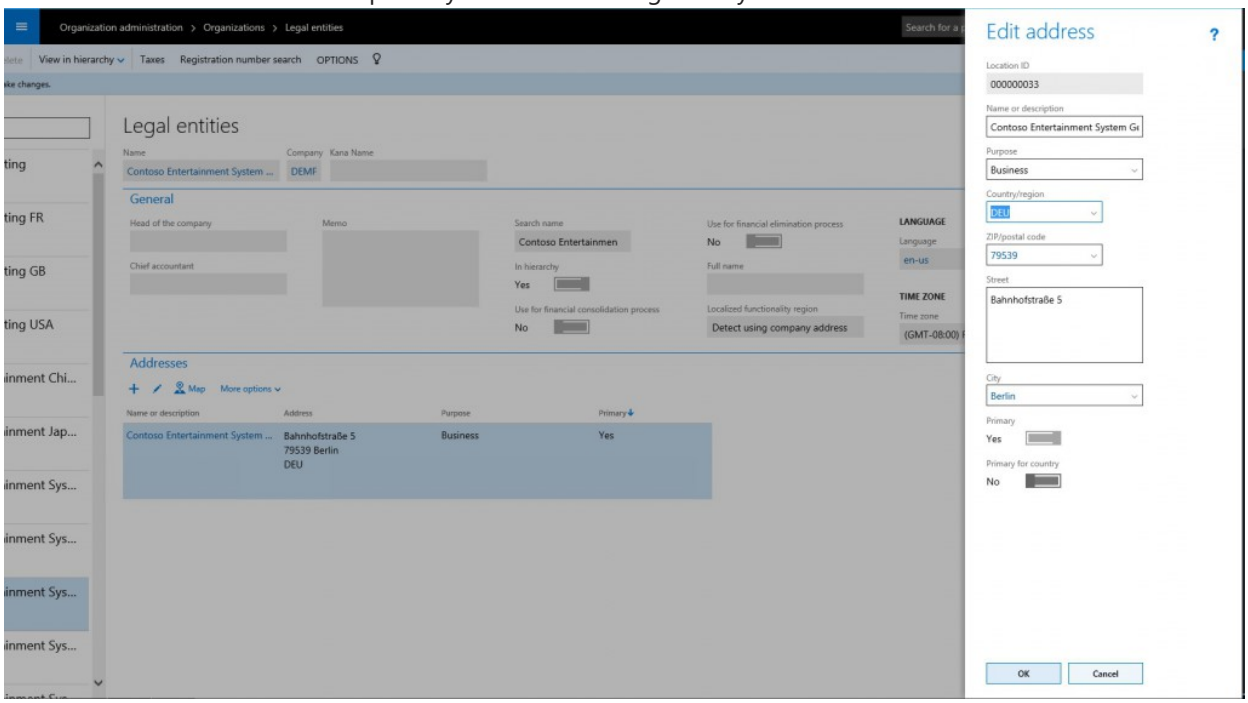
You create country/region context on a controlled entity by setting the ISO code value on the

CountryRegionCodes property. You can find the list of ISO country and region codes on the ISO website. The values of the **CountryRegionCodes** property are compared to the country/region context of the controlling party. If the values match, the element is shown. Otherwise, it's hidden.

> **TIP**
>
> To add more than one ISO country and region code to the **CountryRegionCodes** property, use a comma-separated list.

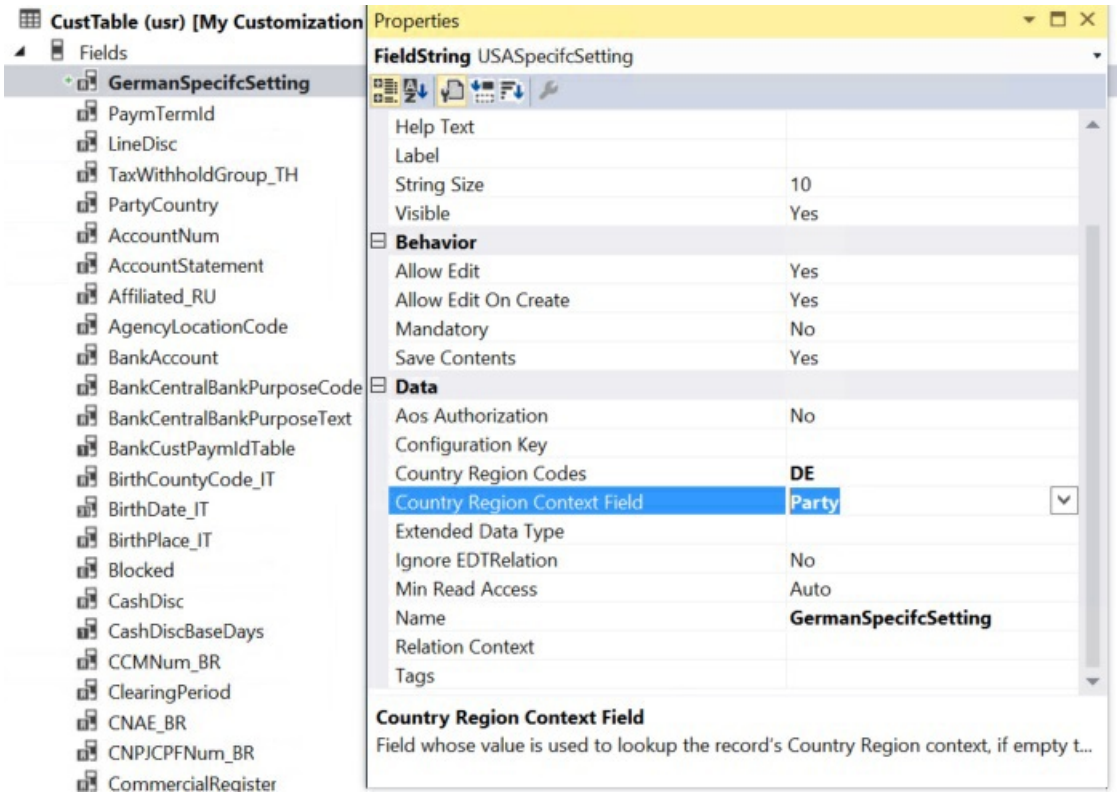
Using the legal entity as the controlling party

The **Country/region** value in the primary address of the legal entity determines the country/region context of the controlling party. The default value of the **Country/region** field is the locale of the system. The following illustration shows how to set the primary address of the legal entity.



Setting another party as the controlling party

You can use another party, such as a customer, bank, or vendor, as a controlling party. For example, you can enable targeted functionality for customers of a specific country/region or require specific validation of vendors from a specific country/region. To set the controlling party, use the **CountryRegionContextField** property of the form, control, or other element. This property lets you select the entity that is the controlling party. The default value is the legal entity. The following illustration shows how to set the **CountryRegionContextField** property for a field.



In this example, the customer becomes the controlling entity. The customer's address is compared with the value of the **CountryRegionCodes** field to determine whether the **GermanSpecifcSetting** field is displayed.

Additional resources

ISO codes

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Regulatory certification information in feature titles

2/18/2021 • 2 minutes to read • Edit Online

As part of the requirements for LCS solutions for localization & translation, localization ISV solution providers must include details about any regulatory certifications that the solution requires in order to be legally compliant for sale in the intended market. This article shows how information about certifications is used in the title of the feature.

Regulatory certification can take various forms, from data privacy to certification of compliance with specific regulations. However, one thing that all regulatory certifications have in common is that they are required by the laws and regulations of the country/region of operation. These certifications are enforced, and non-adherence can lead to severe consequences for an organization that does business in that country/region. When the business process library is constructed, regulatory certifications must be identified as regulatory requirements in the Microsoft Dynamics Lifecycle Services (LCS) Business process modeler (BPM) library through the title of the localization solution feature. The label for this title will conform to the following naming convention that indicates the country/region and certification type through prefixes, as shown in the following table.

FEATURE TYPE	FORMAT	EXAMPLE
Regulatory	XX-REG-Certification for xx	PT-REG-Certification for Fiscal printers

The following table explains the components of the naming convention.

COMPONENT NAME	FORMAT	DESCRIPTION	EXAMPLE
Country/region code	XX	The two-letter ISO country/region code (from the ISO 3166 standard)	PT (= Portugal)
Feature type	REG	The type of feature	REG
Certification name	Text	A short title that describes the certification and its application	Certification for Fiscal printers

In the BPM business process library, certifications should be located under **APQC level 8.0 Manage Financial Resources (10009)**.

Example

- APQC level 8.0 Manage Financial Resources (10009)
  - PT-REG-Certification for Fiscal printers

For more information about BPM, see Flowcharts in Business process modeler (BPM).

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

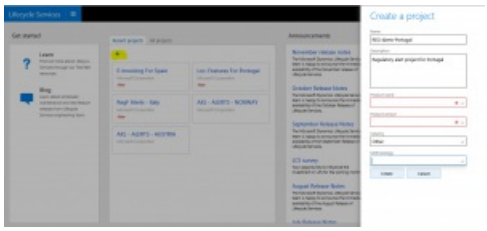
Regulatory watch and communication of regulatory updates

2/18/2021 • 7 minutes to read • Edit Online

As part of the requirements for LCS solutions for localization & translation, localization ISV solution providers must undertake their regulatory watch by taking advantage of localization tools in Microsoft Dynamics Lifecycle Services (LCS).

Set up an alerting project in LCS

Localization independent software vendor (ISV) solution providers must create a new project in Microsoft Dynamics Lifecycle Services (LCS) to record regulatory alerts.



Follow these steps to set up the project.

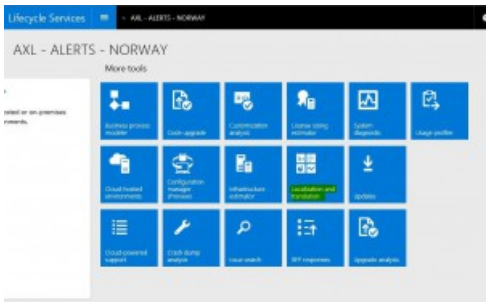
1. Add a new project by clicking the plus sign (+).
2. Enter a name that uses the following project naming convention: **REG-Alerts-Country/region name**
3. Enter a project description.
4. For the product name, specify the latest version of your Dynamics 365 Finance and Operations app.
5. For the product version, specify the latest version.
6. Specify the industry:
   - Select **Other** if the solution is related to all industries.
   - Select an appropriate specific industry.
7. For the methodology, specify **Sure step**.
8. Click **Create**.

Invite participants to the project

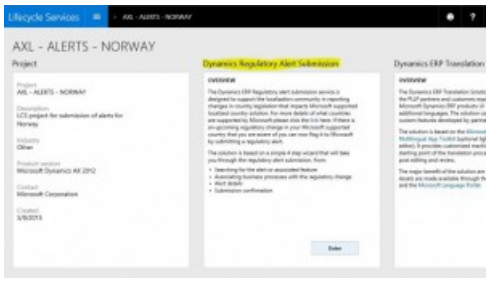
Invite participants that should have access to the project, so that they can submit and review regulatory alerts. For information about how to invite users, see Configure Lifecycle Services (LCS) security.

Access the regulatory alert submission service

1. In your LCS project, scroll to the right side of the page, and then, under **More tools**, click **Localization and translation**.



2. Select **Dynamics Regulatory Alert Submission**, and then click **Enter**. The **Dynamics Regulatory Alert Submission** page opens. Use this page to view any previous alerts that have been submitted by you or your organization.



Submit a regulatory alert

To enter a new regulatory alert, click the plus sign (+) at the top of the form, above the filter. The **Submit regulatory alert** wizard starts. Follow these steps to complete the wizard.

1. On the **Search for existing items** page, follow these steps:

    a. Use Issue search to identify whether a regulatory feature that is related to the alert already exists. Enter search terms to specify criteria such as a keyword, country/region, Microsoft Knowledge Base (KB) number, or Application Object Tree (AOT) object. For example, you can search for the payment term "SEPA" to return related items.

    b. After you've finished entering the search terms, click the **Search** button. Search returns all items that meet search criteria as it searches across both product issues and regulatory features.

    c. You can narrow down the search results by using the filter/criteria that are available.

    d. If you don't find the regulatory feature that you're searching for, you can submit a regulatory alert by clicking **Submit regulatory alert** on the bottom ribbon.

2. On the **Attach business processes** page, follow these steps:

    a. In the **Global business process libraries** field, you can select business process libraries.

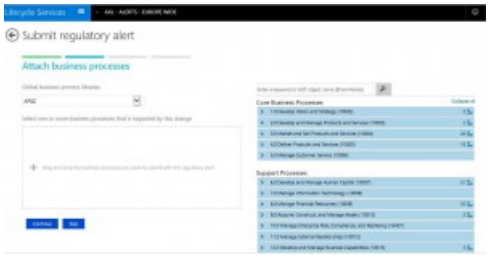
    b. You can enter search criteria to return business processes that are related to a search term. The tool highlights these business processes in yellow.

    c. You can drag business processes into the marked area on the left side of the page. You can select one or many related business processes in the process list. After you've dragged business processes to the marked area on the left, you can edit them further. Deselect business processes by clicking the **x**.

    d. When you've finished, click **Continue** to go to the next page in the wizard. When you receive a

confirmation message, click **Yes**. If no relevant business processes are found, you can skip this page in the wizard.

e. You receive a message that asks whether you want to add the selected business processes to the alert. You can click either **Yes** or **Cancel**.



3. On the **Describe the alert** page, follow these steps:

a. Enter information for the alert in the appropriate fields. Required fields are indicated by a red asterisk (*). The following table provides more information about the submission fields.

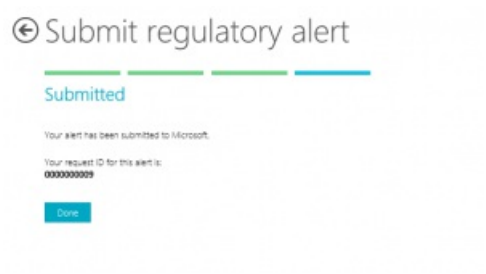
FIELD NAME	FIELD TYPE	DESCRIPTION
Title	Text	A descriptive title to identify the area of impact. For example, you might enter **Changes in invoice document per Jan. 1st 2014**.
Description	Text	A brief, pragmatic overview of the law. The description should focus on issues that are relevant to enterprise resource planning (ERP). It should have three to ten lines of details, so that users gain a high-level understanding of the requirement without having to read the legislation first.
Country	Valid values list	The country/region that the legislation applies to.
Industry	Valid values list	A list of industries, if the requirement applies only to selected industries (for example, Public sector, Retail, or Communication).
Link to legislation	Text (URL format)	Add links to the published law, interpretation guideline, implementation guidance, or any other documentation that will be useful for understanding and implementing the requirement.
Feature reference	Text	The feature reference ID (if it's known).
Law enforcement date	Date	Date from which impacted customers must comply with the law
Government announcement date	Date	The date when the authority announced the change.

FIELD NAME	FIELD TYPE	DESCRIPTION
Latest filing date	Date	The deadline for the first submission of the new/changed report.
Company name	Text	The company name for the person who is submitting the alert.
Contact name	Text	The contact name of the person who is submitting the alert.
Contact email	Email address	The contact email address of the person who is submitting the alert. This value must be in a valid email address format.
Business processes	Default	The business processes are selected through the **Submit regulatory alert** wizard and entered automatically on the submission page.
Comments	Text	Enter any additional information that is related to the alert, and that might be useful for understanding or implementing the requirement. You can add multiple comments. Click **Submit** to save comments separately to the submission page. Comments are saved in order of date.
Attachments	Upload	Add alert attachments by using the attachment tool. Click the **Upload** button to open File explorer. After you select and upload a file, the file is appears as a linked file. You can upload up to three files, each of which can be up 5 MB. To delete a file that you've uploaded by mistake, click **Remove** under the file title. **Important:** Attachments must be publicly available materials. They can't be propriety or customer/partner specific.
Consent check box	Check box	Give appropriate consent to being contacted. The **Submit** button doesn't become available until you select this check box.

b. When you've finished adding all the required fields, you must provide appropriate consent by selecting the consent check box: **By creating this alert I allow Microsoft to contact me in the future for any further information related to this alert. Read the Microsoft Dynamics Lifecycle Services privacy statement for more information.**

c. After you select the check box, the **Submit** button becomes available. Click **Submit** to submit the alert. If you've partially completed an alert, you can save the information that you've already entered for later completion or review. Click **Save** before you submit the alert.

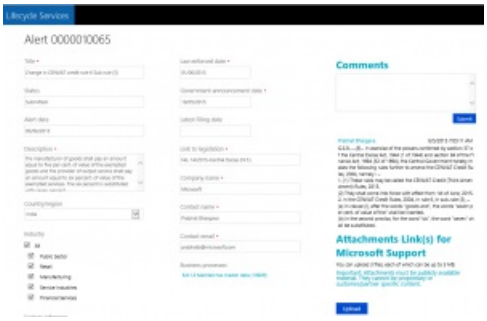
d. When you receive a confirmation message that states that the alert has been successfully submitted, click **Done** to exit the wizard. If you chose to save the alert before you submitted it, an alert ID is generated, and you are notified that the alert has been saved.



Review regulatory alerts that have been entered into the project

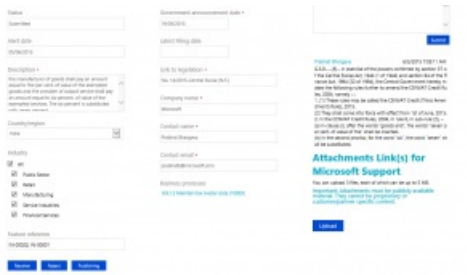
To view the regulatory alerts that have been entered into your alerting project, use the alert grid. This grid provides a high-level list view of the alerts that have been submitted, and shows the alert title, country/region, law enforced date, and so on.

You can search the contents of the grid by using the filter/search field and then selecting from the default search options. You can drill into the detail of an alert by clicking the alert ID, which is a hyperlink. The completed alert submission page opens, where you can review the alert details, and also any comments and attachments.



Process submitted alerts options

After an alert has been submitted to the LCS alert project, you can process it from the grid view by clicking the alert ID hyperlink. Project owners can then change the status of the alert to notify project members whether action will continue to be taken for the alert. Options appear when you drill into details of the alert.



The following table describes the processing options that are available for alerts.

STATUS	STATE	ACTIONS
Submitted	The alert has been submitted to (entered into) LCS by a Loc. Community member.	The alert can be either received or rejected.
Received	The alert has been reviewed by an internal Sol. Partner resource and has been received for further review.	The alert is marked **Received**. A comment and feature reference can be added.
Rejected	The alert has been reviewed by an internal Sol. Partner resource and has been rejected.	The alert is marked **Rejected**. A comment must be added to explain why the alert was rejected.
Reopened	The alert was rejected but has been reopened by a Loc. Community member, and new/additional information has been entered (a comment is mandatory).	The alert can be reviewed again to assess whether it should be received or rejected.

NOTE

Submitted alerts can be rejected for various reasons. Here are some examples:

- The alert is too vague to identify the underlying localization feature.
- The alert is related to an area where no features are localized.
- The alert is related to an area that isn't currently supported by Finance and Operations functionality.

Alerts can be stored in LCS as references.

Further processing, such as potential engineering of a related feature, will be handled in the ISV solution provider's existing systems.

Separation of localization models

2/18/2021 • 2 minutes to read • Edit Online

As part of the requirements for LCS solutions for localization and translation, if a localization solution for previous versions of Dynamics 365 Finance and Operations apps contains both regulatory and competitive features, localization ISV solution providers must split the solution into separate models for each feature type. This article provides information about this requirement.

Multinational customers must comply with regulatory requirements in all countries/regions where they deploy Finance and Operations apps. At the same time, these customers want to minimize the cost of code maintenance. Therefore, if a localization solution for previous versions contains both regulatory and competitive features, the solution must be split into separate models, so that customers can adopt and deploy the features that they require. An effort has been made to split the Application foundation and Application suite stack into multiple models.

The number of models is expected to grow over time. Splitting a monolithic code base provides many benefits, such as better scalability, manageability, and serviceability. The localization requirement to split a localization solution into more granular models builds on this effort. The goal is to provide the same benefits to multinational customers.

After you've classified features as either regulatory or competitive, as described in Classification of localization features, split the code for these features into at least two models, one model for the regulatory features and at least one model for the competitive features. If the competitive features can be split further (for example, into features that are related to Commerce and features that are related to Fixed assets), it's a good idea to split them. However, further splitting isn't mandatory. For more information about how to split the stack into multiple models, see Model split.

Submit alerts about country/region-specific regulatory features

2/18/2021 • 4 minutes to read • Edit Online

This topic describes how to use Microsoft Dynamics Lifecycle Services (LCS) to submit alerts through the Dynamics regulatory alert submission service. This topic also explains how to track planned and released regulatory features through LCS Issue search.

Accessing the regulatory alert submission service

In Dynamics Lifecycle Services (LCS), in your project, scroll to the right side of the page, and then, under **More tools**, click the **Alert service** tile.

The **Dynamics regulatory alert submission** page appears. You can use this page to view any alerts that have previously been submitted by you or your organization.

Submitting a regulatory alert

To enter a new regulatory alert, click the plus sign (+) at the top of the **Dynamics regulatory alert submission** page, above the filter. The **Alert submission** wizard starts. You can complete the following tasks in this wizard:

- Search for existing regulatory items.
- Attach business processes.
- Describe an alert.
- Confirm submissions.

Search for existing regulatory items

Use Issue search to identify whether a regulatory feature, that is related to the alert, already exists.

1. Enter a search term, such as a keyword, country/region, Microsoft Knowledge Base (KB) number, or Application Object Tree (AOT) object. Click the search button. Any items that include the search term, in either product issues or regulatory features, appear in the search results. You can narrow the search by using the filters that are available.
2. If you don't find the regulatory feature that you're looking for, you can submit a regulatory alert by clicking **Submit regulatory alert** at the bottom of the browser window.

Attach business processes

1. In the **Global business process libraries** list, select business process libraries.
2. Enter search criteria to find business processes that are related to the search term. These business processes are highlighted in yellow.
3. In the list on the right side of the page, select one or more related business processes, and drag them into the field on the left . After you've finished, you can edit them further by clearing the selection of business processes.
4. Add the selected business processes to the alert.

Describe the alert

1. Enter information about the alert in the appropriate fields. Required fields are indicated by a red asterisk. The following table provides more information about the fields on the **Describe the alert** page.

Field	Description
Title	Enter a descriptive title to identify the area of impact. For example, enter **Changes in invoice document as of January 1, 2018**.
Description	Enter a brief overview of the law. Your description should focus on issues that are relevant to enterprise resource planning (ERP), so that users can understand the requirements at a high level without having to read the legislation first.
Country	Select the country or region that the legislation applies to.
Industry	Select the industry, if the requirement applies only to specific industries. For example, select **Public sector**, **Commerce**, or **Manufacturing**.
Feature reference	Enter the feature reference, if you know it.
Law enforcement date	Select the date when affected customers must start to comply with the law.
Government announcement date	Select the date when the authority announced the change.
Latest filing date	Select the deadline for the first submission of the new or changed report.
Link to legislation	Enter one or more links to the published law, interpretation guideline, implementation guidance, or any other useful documentation that will help users understand or implement the requirement.
Company name	Enter the company name for the person who is submitting the alert.
Contact name	Enter the name of the person who is submitting the alert.
Contact email	The email address of the person who is submitting the alert.

Business process	The business processes that you selected through the **Alert submission** wizard.
Comments	Enter any additional information that might be help users understand or implement the requirement. Click **Submit** to save your comment. Multiple comments can be added and should be submitted separately. Comments are saved in the order that they are added.
Attachments	Click the **Upload** button, and then browse to select a file to add as an attachment. After you select the file, it's uploaded and appears as a linked file. You can add up to three files that have a size of 5 MB each. To delete files that have been attached, click **Remove** under the title of the file. **Note** Attachments must be publicly available materials. They can't be propriety or customer-specific/partner-specific.

2. After you've finished entering all the information, select the consent check box (**By submitting this regulatory alert, I consent to Microsoft contacting me for additional information about this alert. Microsoft Privacy Statement.**). When you select the check box, the **Submit** button becomes available.

3. Click **Submit** to save and submit the alert.

If you don't have all of the required information, or if you're not yet ready to submit the alert, you can save a partially completed alert.

Confirm your submission

- When the alert is successfully submitted, you receive a confirmation message. Click **Done** to exit the wizard.
- If you save the alert before you submit it, an alert ID is generated, and you receive confirmation that the alert has been saved.

Track the status of regulatory features in Issue search

You can use Issue search in LCS to find planned and released regulatory features, and any associated localization documentation, certifications, and reports. To narrow your search to regulatory features, use the following filters:

- **Category** - Select **Regulatory feature** only.
- **Country/region** - Click **>** to select the country/region that you're interested in.

To narrow the search even more, you can apply the following additional filters

- **Product** - Select the products and product versions that you're interested in.
- **Status** - Select specific statuses.

Get support for Finance and Operations apps or Lifecycle Services (LCS)

2/18/2021 • 2 minutes to read • Edit Online

This topic explains how to get help with Finance and Operations apps or Microsoft Dynamics Lifecycle Services (LCS).

TASK	MORE INFORMATION
Ask the community.	Go to the Dynamics 365 Community page to get help with your questions from the Microsoft Dynamics community.
Get help with questions about licensing.	Contact your partner or a Microsoft sales representative.
Use the **Issue search** tool.	In LCS, use the **Issue search** tool to quickly search for Microsoft Knowledge Base (KB) articles, hotfixes, and workarounds for reported issues. You can see which reported issues are in the process of being fixed for a specific functional area, and which issues have already been fixed. For more information, see Issue search (Lifecycle Services, LCS).
Get in-app support.	Select the **Help** button (**?**) in the upper-right corner of the app, and then select **Support**. Issues are reported on the **Active issues** tab in LCS. There, admins can determine whether they should provide in-house support or submit the issues to Microsoft.
Open a support ticket with the Microsoft Support team.	In LCS, the **Support** tile opens a tool that helps you manage support incidents. To submit issues directly to Microsoft, select the **Support** tile in your LCS project. You can then submit issues in two ways: • On the **Active issue** tab, select your issue, and then select **Submit to Microsoft**. • On the **Submitted to Microsoft** tab, select **Submit an incident**, and then follow the on-screen instructions to submit the incident. After you submit an incident, you will receive an email message from the Microsoft Support engineer who is assigned to your case.
Request new features and functionality.	Visit Dynamics 365 Application Ideas to view, search, or vote for existing ideas, or to add new ideas.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Support for Finance and Operations operated by 21Vianet in China

2/18/2021 • 2 minutes to read • Edit Online

Finance and Operations apps provides many self-service support options and support through 21Vianet.

Self-help resources

- Finance and Operations application documentation
- Help resources for Supply Chain Management
- Finance and Operations apps - operated by 21Vianet in China
- Dynamics community
- Microsoft Learn

Assisted support

- Open a support request

Presales support

Pre-sales support phone number: +86 400-886-6134

Pre-sales support provides assistance on subscription features and benefits, plan comparisons, pricing and licensing, and helps to identify the right solution to meet your business needs. In addition, pre-sales support can help you find a Partner, and purchase and sign up for a trial. You can call during local business hours, Monday through Friday.

Billing and subscription management support through 21Vianet

Billing and subscription support telephone number: +86 400-089-0365.

Assistance for billing and subscription management issues is available online or by telephone Monday through Friday during local business hours 9:00 to 18:00 China Standard Time (CST). Billing and subscription management support can be accessed using the same phone number and online service request process as with technical support.

Here are some examples of billing and subscription management issues:

- Signing up for a trial or purchasing a subscription.
- Converting from a trial subscription to a paid subscription.
- Understanding the bill.
- Renewing a subscription.
- Adding or removing licenses.
- Canceling a paid subscription.

Assisted Technical support through 21Vianet

When you experience a technical issue with your deployment, report it to 21Vianet through the LCS portal or by calling the support number at +86 400-089-0365. Technical support hours of operations are Monday through Friday during local business hours 9:00 to 18:00 China Standard Time (CST).

A support request (SR) is handled within hours, depending on the severity of its impact to your business:

- **Critical business impact** - You will receive an initial response within 1 hour or less, and a support representative will work continuously, all day, until the problem is resolved. You will be expected to allocate appropriate resources to work on the request until the problem is resolved and provide accurate contact information to the support personnel handling your case.
- **Non-critical business impact** - You will receive an initial response within 8 hours or less. You will be expected to provide accurate contact information to the support personnel handling your case.

Get Premier support

If you run mission-critical solutions, Premier support offers additional value:

- Proven advisory services designed to maximize your Dynamics 365 investment.
- A designated service delivery manager committed to improving your Dynamics 365 experience.
- Top priority reactive support to help ensure service continuity.

For details about purchasing Premier support, contact your Microsoft Account team. If you have a Premier support plan you can contact support via My Premier Online.

Additional resources

- Dynamics 365 support site for 21Vianet (Chinese)
- Finance and Operations apps - operated by 21Vianet in China
- Model-driven apps in Dynamics 365 - operated by 21Vianet in China
- Dynamics 365 Privacy statement (Dynamics 365 隐私声明)
- Dynamics 365 Service Level agreement (世纪互联在线服务的服务级别协议)
- Dynamics 365 Legal information (Dynamics 365 法律信息)
- Service terms for Dynamics 365 Lifecycle Services
- OSPT of Dynamics 365 (世纪互联在线服务的服务级别协议)
- Azure Docs (in Chinese)
- Azure China 21Vianet

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Set up technical support for Finance and Operations apps

2/18/2021 • 9 minutes to read • Edit Online

Prerequisites

Before you can set up technical support, you must acquire a Microsoft Azure Active Directory (Azure AD) account. This account is created when you set up a subscription for one of the Microsoft Dynamics 365 Finance and Operations apps.

Create an Azure DevOps project

The **Support** tile in a Lifecycle Services (LCS) project uses Azure DevOps to store issues that are submitted through the client and issues that are manually created from the **Support** tile in LCS. This functionality requires that a Azure DevOps project be configured in the LCS project that you want to use for support. All users who need to use the **Support** tile to submit an issue must have access to the Azure DevOps project, and must authorize LCS to access Azure DevOps on their own behalf. Most users don't have access to LCS or Azure DevOps. Therefore, in the Azure DevOps project, you should create a special system account that can be used to submit issues.

Create a new Azure DevOps project

1. Go to https://www.visualstudio.com/.
2. Click **Sign in** in the upper-right corner.
3. Sign in by using an AAD account that is in the tenant that your subscription is linked to. If the browser already has your credentials, you won't see the sign-in page and should instead click your name in the upper-right corner.
4. On the right side of the page, under **Accounts**, click **Create a free account now**.
5. Specify an account URL, and then click **Create Account**.
6. Name your project, and specify a process template. Your project should now be created.

Add users to the Azure DevOps project

1. In the upper-left corner, click **Team Services**.
2. On the **Users** tab, click **Add**, and invite users who will use the Support experience to the Azure DevOps account. For each user that you invite, select either **Basic** or **Stakeholder**.
3. In the upper-left corner, click **Team Services**.
4. Click **Browse**, and browse to the project that you created in the previous procedure.
5. In the **Members** section of the project home page, click **Add**, and add the users that you invited in step 2.

Create the Support system user

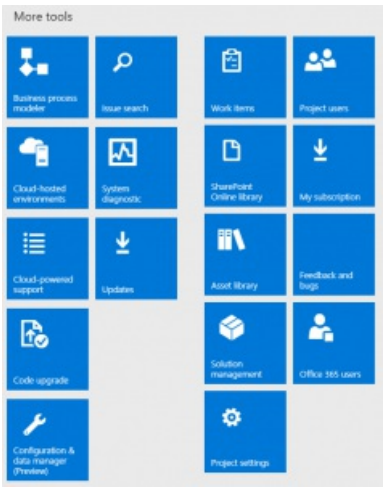
1. Create a new user in your Azure AD tenant, and enter a descriptive name, such as **LcsCpsSystemAccount**.
2. In the upper-left corner, click **Team Services**.
3. On the **Users** tab, click **Add**, and invite the system user that you created in step 1. For this user, select **Stakeholder**.
4. In the upper-left corner, click **Team Services** again.
5. Click **Browse**, and browse to the project that you created earlier.
6. In the **Members** section of the project home page, click **Add**, and add the system user.

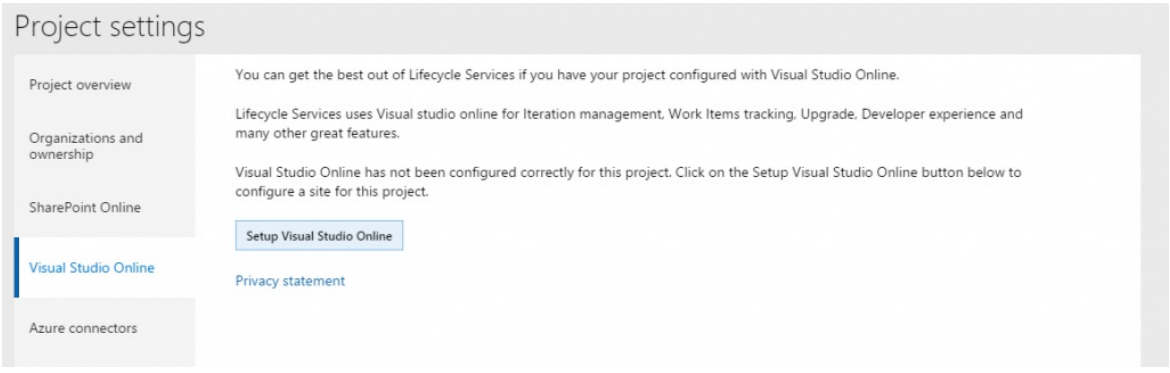
Retrieve the personal access token for the Support system user

1. Sign out of Team Services by clicking the user name in the upper-right corner and then clicking **Sign out**.
2. Sign in to Team Services by using the Support system account that you created in the previous procedure.
3. In the upper-right corner, click the user name, and then click **My profile**.
4. On the **Security** tab, on the **Personal access tokens** tab, click **Add**.
5. Enter a description, such as **LCS Support system account**.
6. Select an expiration date of one year.
7. Click **Selected scopes**, and the select **Work items (read and write)**.
8. Click **Create token**.
9. Copy the token and paste it in a safe location because it won't be accessible after you move away from the page.

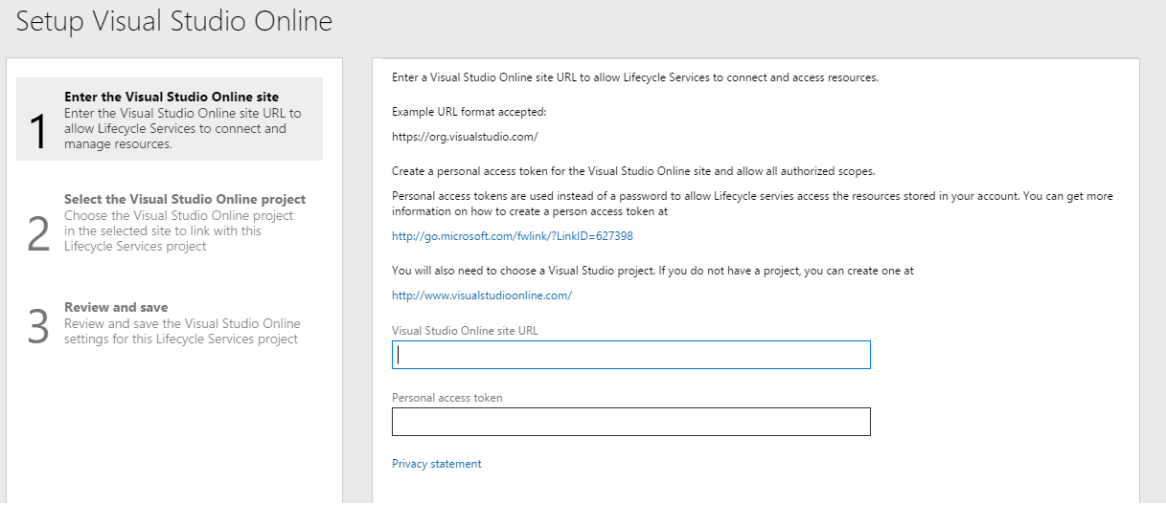
Configure LCS

1. Sign in to LCS by using an account that has the **Owner** role for the LCS project that the application is deployed in.
2. Open the project in LCS.
3. Click **Project settings**, and then click the **Azure DevOps** link.





4. Click **Setup Azure DevOps**.
5. In the **Azure DevOps site URL** field, enter the URL of the Azure DevOps project that you created in the previous section.
6. In the **Personal access token** field, enter the personal access token that you created in the previous section.

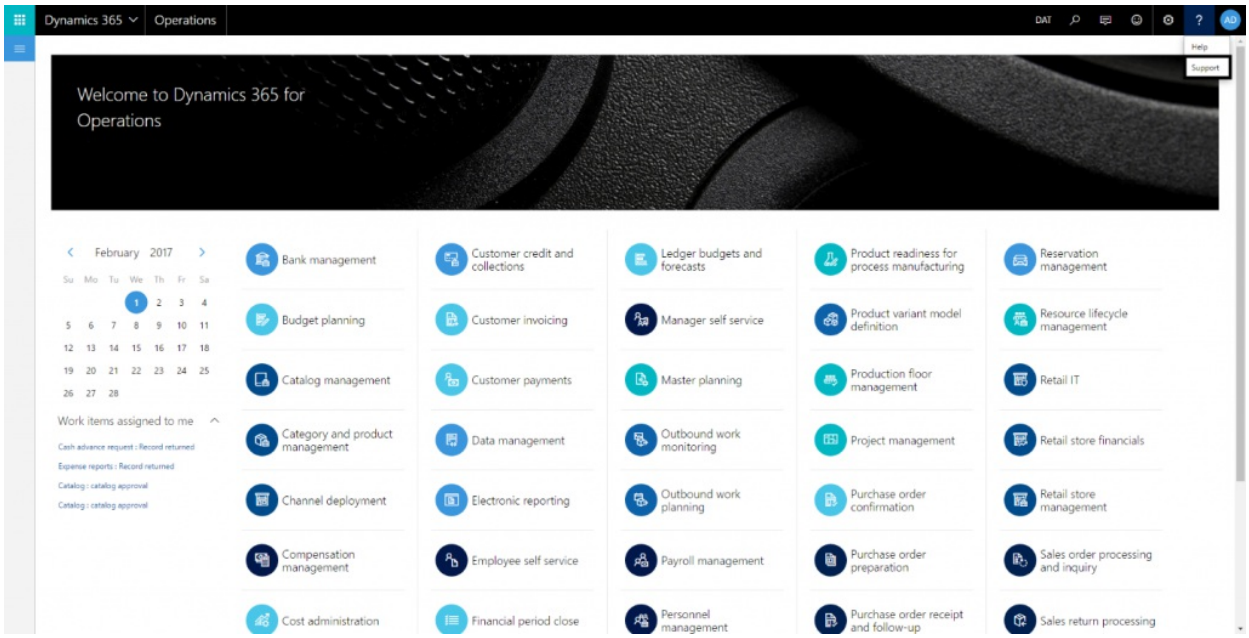


7. Click **Continue**.
8. Select the VSO project to use, and then click **Continue**.
9. Click **Save**.
10. Click **Authorize**.
11. In the confirmation message box, click **OK**.
12. Sign in to Visual Studio Online.
13. Click **Accept**.

Create an issue

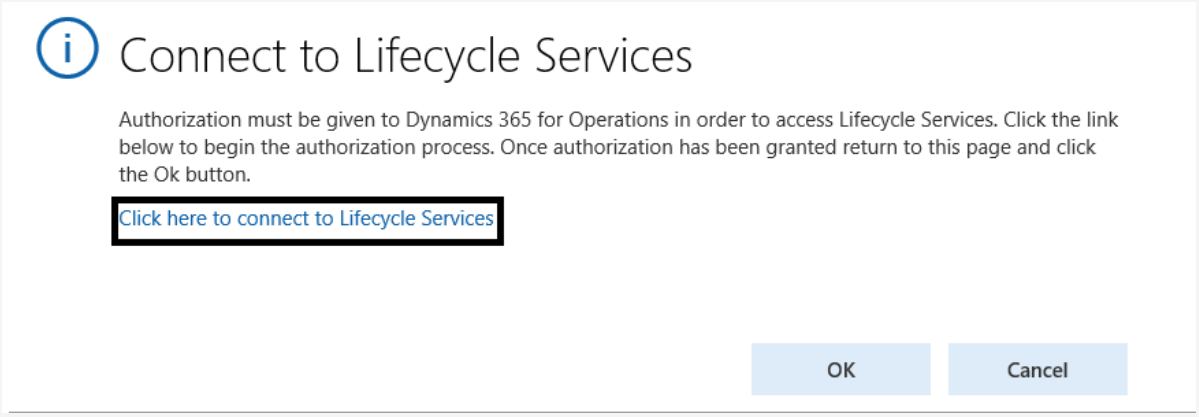
The Support experience has been updated to show updates that are published by Microsoft. In the client, on the top bar, click **?**, and then click **Support**.

> **WARNING**
> If you have an on-premises deployment, the option to search for existing issues and submit a support incident from the on-premises client to your Azure DevOps project is not available.



NOTE

If you haven't already connected to Lifecycle Services (LCS), a dialog box will display where you can connect. Click the link to connect before proceeding.



Search for a fix

After you connect to LCS, you can search for existing Microsoft published updates and fixes. Enter your issue in the **Search** box and press **Enter**.

NOTE

If you don't want the functionality to search for existing fixes enabled for all users, you can remove the **SearchExistingFixes** duty from the System user role and add it to only those roles which you want to have this functionality. Search results are based on the Microsoft Issue Search data that is relevant to your environment. Fixes that you have already installed will not be included in your search results. To view a specific result, click the link to view the details.

Based on the duties assigned to you, you will see either the **Download view** or the **Request view**.

- **Download view** - By default, this view is only available to system administrators. From this view, you can directly download the hotfix.

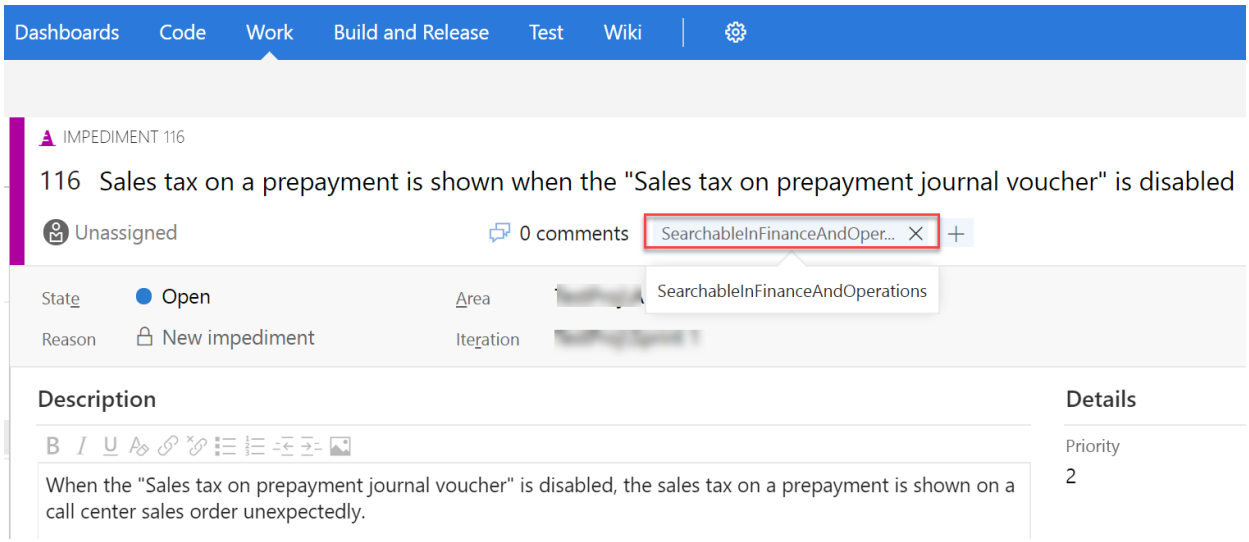
  NOTE

  The duty **DownloadHotfix** controls the ability to directly download fixes from LCS rather than requesting them. Only system administrators will have access to it by default. If you want to assign this duty to users other than system administrators, you can do so by adding the duty to the selected roles.

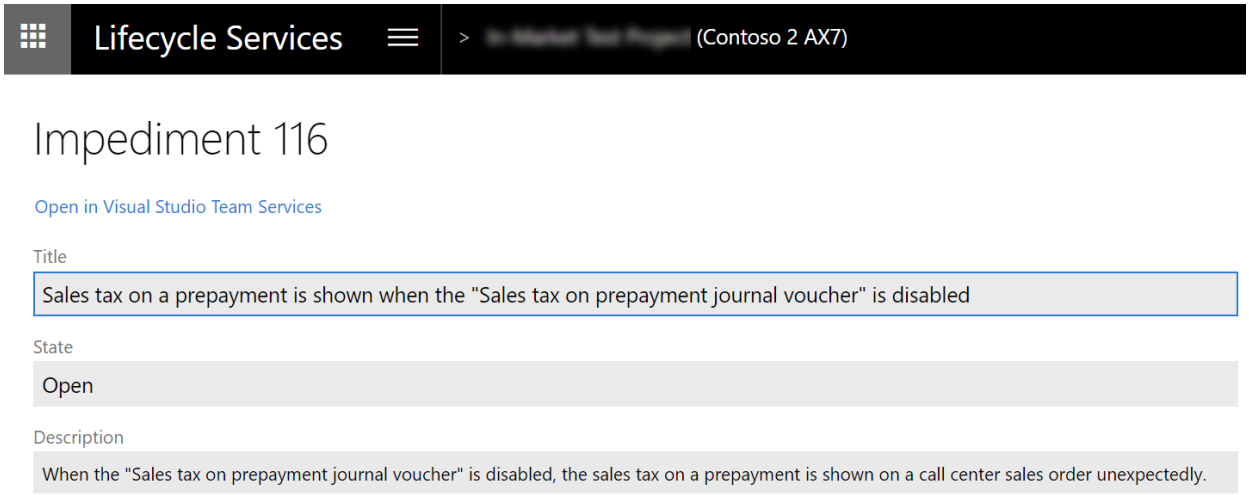
- **Request view** - By default, this view is available to all users who are not system administrators. From this view, you can make a request to download the hotfix. After you submit your request to download the hotfix, a work item will be created in the Azure DevOps project that is associated to your LCS project. The customer IT admin can view all requested hotfixes by clicking the **Support** tile in LCS and then clicking the **Hotfix requests** tab.

Search for project work items in Azure DevOps

The Azure DevOps administrator can publish project work items to your organization users by tagging the work items with **#SearchableInFinanceAndOperations**. The tagged work items will be searchable for users from the client support search box. The search result will include tagged Azure DevOps work items in addition to Microsoft published updates and fixes. The following graphic shows a tagged Azure DevOps work item for publishing.



When you search for published Azure DevOps work items using the support search box, search results show the work item's type, title, state, and description in a new browser tab with **view** mode. Users with proper permissions can edit the work item in Azure DevOps. The following graphic shows the search result of a published Azure DevOps work item.



> **NOTE**
>
> The published Azure DevOps work items are only visible to your organization's users.

Create and submit a new issue

If you don't see a fix in the search results, you can create a new issue by clicking **Create**. This is the same functionality that is available for previous releases and is documented in earlier procedures.

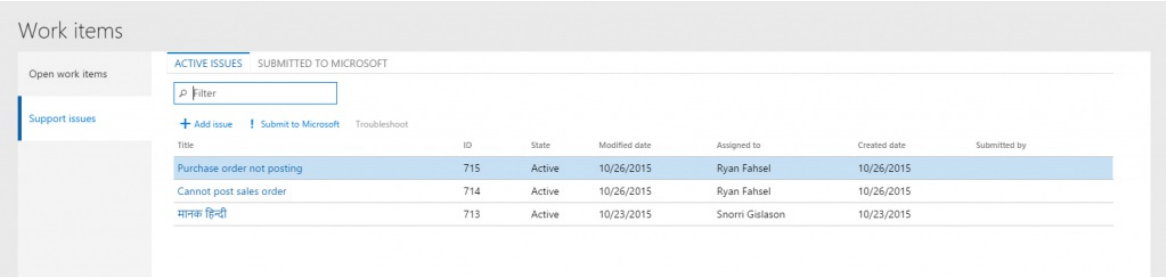
Work with issues in LCS

View issues

In the LCS **Support** tile, issues are stored as work items in the Azure DevOps project that is associated with the LCS project. Specifically, issues are stored as work items of the **Issue** or **Impediment** type, depending on the type of Azure DevOps project, in the **AxAndLcsGeneratedIssues** area. Every work item of one of those types in that area will be included in the list of issues in the **Support** tile. If an issue is modified in Azure DevOps, the changes will be reflected in Support issues. Issues can be assigned to any user in the Azure DevOps project. Users don't need to have access to LCS to work with issues in Azure DevOps.

1. Go to lcs.dynamics.com, and sign in.

2. Open the LCS project that is associated with the environment that you want to view issues for.
3. Click the **Support** tile. A list of the issues that have been created appears.



Edit issues

1. In the **Issues** grid, click the title of an issue.
2. If necessary, sign in to Azure DevOps by using an account that has access to the Azure DevOps project that you set up in the first section of this topic, **Create an Azure DevOps project**.

> **NOTE**
> There is an issue in Azure DevOps, where the link to edit work items doesn't work correctly if sign-in is required. If you see the **Assigned to me** query after you sign in to Azure DevOps, go back to LCS, and click the title of the issue in the issue grid again.

3. The Azure DevOps editor opens. Edit the issue, and then save your changes. The changes will be reflected in the **Support** tile.

Submit an issue to Microsoft

You can submit issues to Microsoft support. When you submit an issue to Microsoft, the information and attachments in the issue can be included in the Microsoft support incident.

LCS users must have a valid Microsoft support plan to submit issues to Microsoft. If you have trouble submitting issues to Microsoft, work with your administrator to make sure that your LCS credentials are added to or associated with your organization's support plan with Microsoft Partner Source Business Center.

1. In the **Issue** grid, select the issue to submit to Microsoft, and then click **Submit to Microsoft**.
2. If your account is associated with multiple support organizations, select the organization to use to create the Microsoft support incident.
3. Use **Issue search** to verify that your issue hasn't already been solved.
4. If **Issue search** doesn't provide a solution to your issue, click **Create incident** at the bottom of the page.
5. Share diagnostic data with the Microsoft support team. By providing version information, your issues can be resolved more quickly.
6. Describe your issue, provide your contact information, and then click **Submit**.

Support settings

> **NOTE**
> The information in this section is not applicable to on-premises deployments.

When you deploy your application from Lifecycle Services, no configuration is required, because the Support tool automatically saves any issues to the same LCS project that Finance and Operations was deployed from. To verify the LCS project that Support uses, go to **System administration** > **Setup** > **System parameters**, and

then click **Help** > **Support Contact**.

Prevent users from creating issues from the client

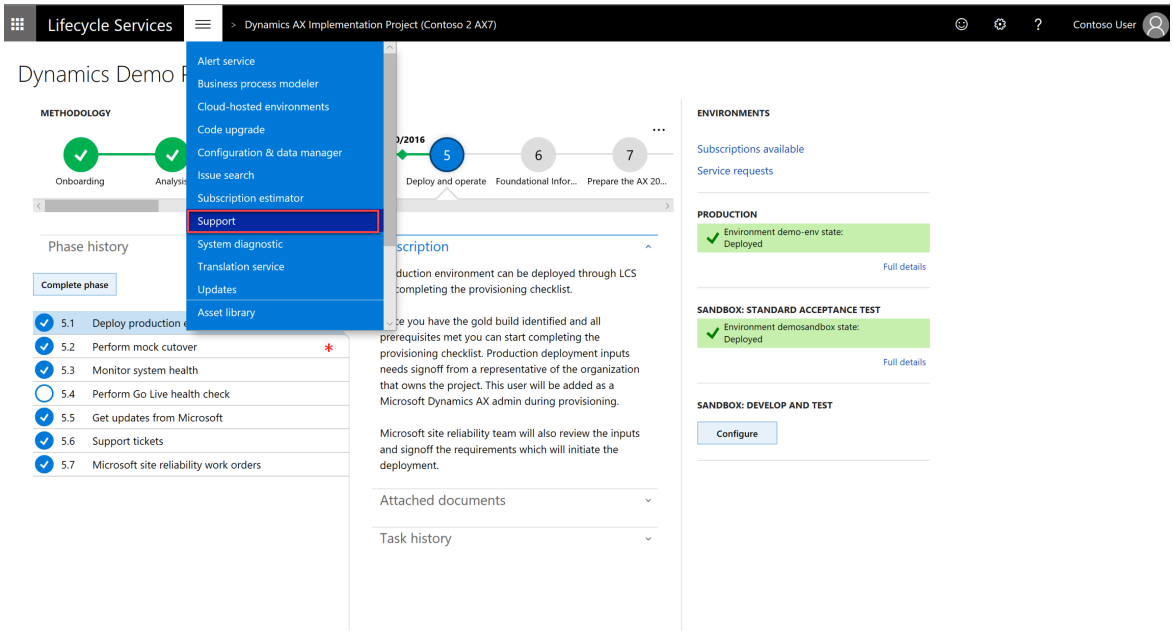
By default, the System user role has the privilege, *SysLCSCPSIssueEntry* assigned. This privilege controls access to the **Contact your support team** menu item on the Help menu. If you want to prevent users from being able to create and submit issues from the client, remove this privilege from the System user role.

Manage support experiences for Finance and Operations apps

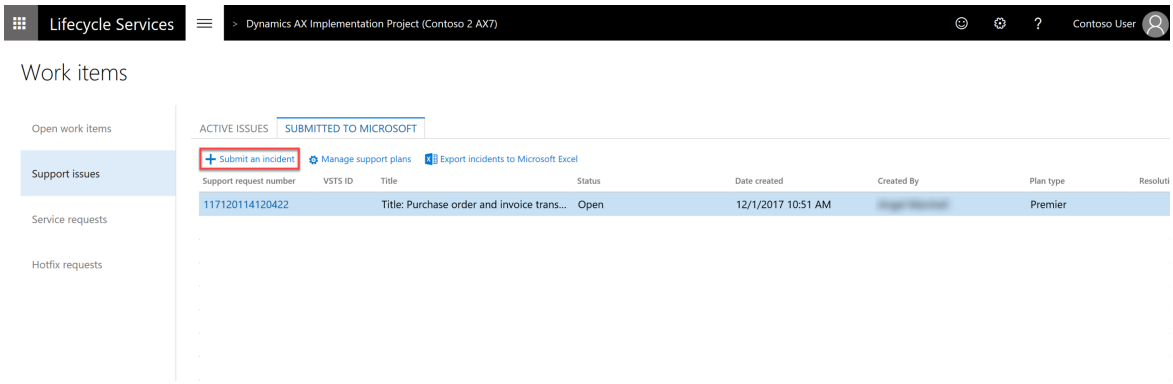
2/18/2021 • 3 minutes to read • Edit Online

Open a new incident

1. In LCS, go to the project for which you want to file a support incident.
2. Click the **Support** tile.



3. On the **Submitted to Microsoft** tab, click the **Submit an incident** button.



4. Select an issue category.
5. Select an issue area.
6. In the **Describe your issue** area, enter the following:
   - Select **Yes** if the issue occurred in an environment. Select the environment name.
   - Enter a short description of your issue in the **Title** field.
   - Provide details about the issue detail and the steps needed to reproduce the error.
   - If applicable, enter an error message.
   - If possible, attach screenshots that illustrate the problem. To do this, click **Attach file from**

computer.

> **NOTE**
>
> When you create an incident, Issue search will populate the top 10 "Possible issue solutions" search results based on the your selection and input, and dynamically refresh these results as more details are provided during support case creation.
>
> Standalone Issue search is still accessible using the drop-down menu if you need to search for more solutions.

7. Enter the primary contact information. These contact details will be used by the customer support team to contact you about the case.
8. Select the support contract and the severity level.
   - Support contracts for on-premises environments have a limited incident count.
   - Support contracts for cloud environments have an unlimited incident count.
   - For on-premises products or cloud environments, from the list of available support contracts, select the support option to use if you have multiple tier support contracts.
9. Click **Submit**.

After you click **Submit**, an incident is created and added to the **Incidents** list. You will receive an email message from the Microsoft Support Engineer assigned to your case.

Support plans in Lifecycle Services

Support plan entitlements are derived based on several different identifiers. Not all will apply to your situation. If you are missing a support plan or entitlement in LCS, determine which identifier is needed to tie it to your project in LCS. If there is more than one organization, note which one is current by clicking on your name in the upper-right corner of LCS. Select the organization that applies to your scenario and contains the benefits that you want to utilize.

Unique contract ID/access ID

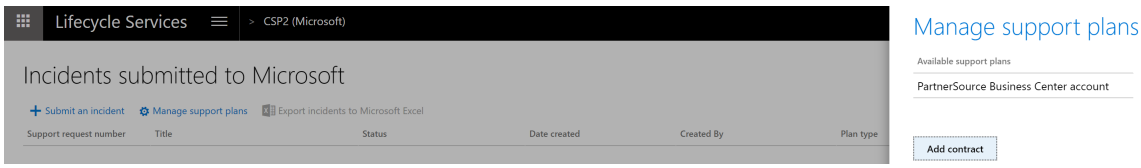
The following online support plans require a unique contract ID/access ID combination linked to your sign-in in LCS:

- Unified
- Premier
- Advanced support for partners

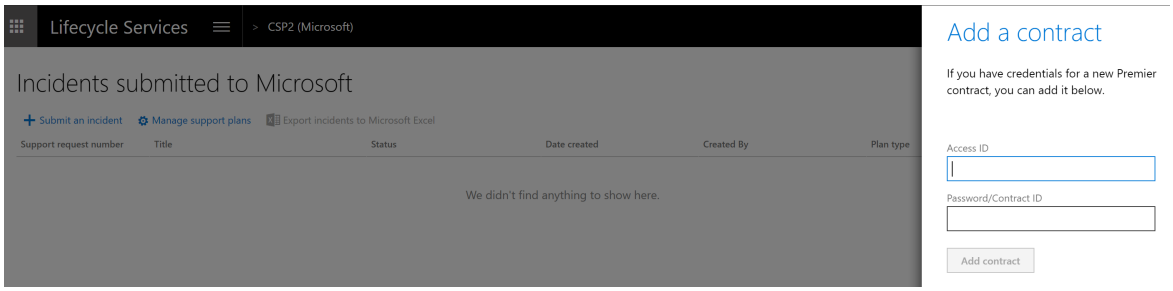
If you do not know your unique contract ID/access ID combination, contact your Microsoft account manager to have an ID created for you.

To link your contract ID/access ID to your account, complete the following steps:

1. From within a project, select **Support** from the main menu, and then select **Manage Support plans**.
2. Select **Add contract**.



3. Enter your access ID and your password or contract ID, and then select **Add contract**.



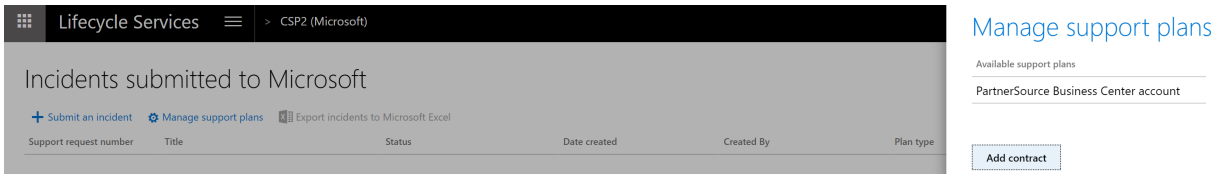
PartnerSource Business Center account

The following support plan incidents can be used as part of your PartnerSource Business Center (PSBC) account if they exist:

> **NOTE**
>
> No online support plans can be utilized through the PSBC account.

- Advanced support for partners on-premises incidents.
- Advantage or Advantage + on-premises incidents.
- Other pay per incident types of plans with an existing incident count in PSBC.

If you do not find the PartnerSource Business Center account, ensure that your sign in is added as a professional in your organization in PSBC. Make sure that you are signing in with the same Microsoft or work account login. This account is only applicable in an on-premises project.



Sign-in specific options

The following incidents and support benefits will appear based on your sign in, if applicable:

- MPN gold and silver incidents.
- Signature cloud support.
- Individual incidents and 5 packs purchased on [support.microsoft.com/supportforbusiness].

> **NOTE**
>
> Incidents must be purchased with a Microsoft account such as @hotmail.com or @outlook.com. Work or Azure Active Directory accounts cannot have incidents tied to them.

Tenant subscription

The following entitlements will appear based on your subscription and ProDirect purchases within your tenant organization:

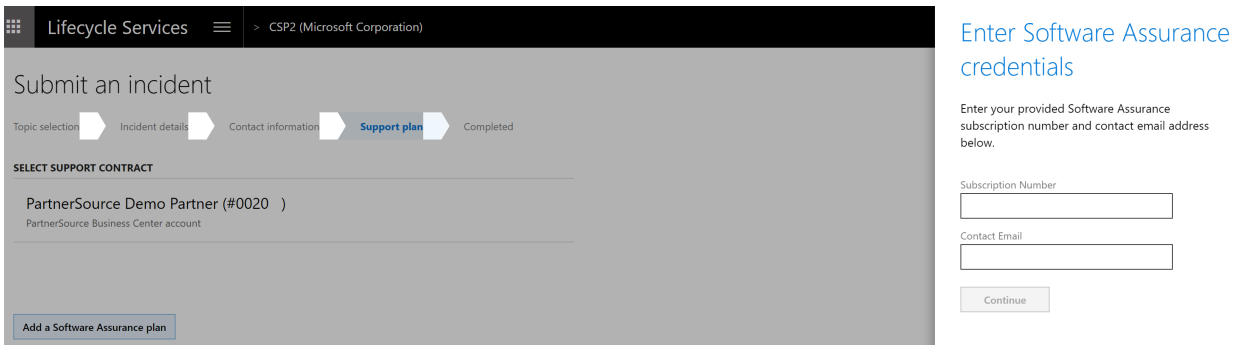
- Subscription
- ProDirect

Software assurance

The following entitlements can be added by linking a subscription number and contact email:

- Software assurance

To add, select **Add a Software Assurance plan** when you create the support incident. Enter the subscription number and the contact email, and then click **Continue**.



Report production outage

For a quick and effective way to escalate issues to Microsoft Support in the event that the services in a production environment are degraded or become unavailable, see Report a production outage.

Phone support

We prefer that you contact Support following the steps in Open a new incident. If you're unable to open a new incident in LCS, phone support is available using Premier phone support.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

System administration home page

2/18/2021 • 4 minutes to read • Edit Online

This topic points to content for system administrators of Finance and Operations. This content will help you configure the system so that it works smoothly and effectively for your organization.

One Version

In July 2018 we announced a change to the way we deliver Dynamics 365 updates that will help you stay current in a consistent, predictable, and seamless manner. The following topics are intended to provide clarity on the Finance and Operations service updates, processes, and tools you can use to stay current.

- One Version service updates overview
- One Version service updates FAQ
- Service update availability
- Apply updates to cloud environments
- Configure service updates through Lifecycle Services (LCS)
- Pause service updates through Lifecycle Services (LCS)
- Get notified about service updates through Lifecycle Services (LCS)

Implementation management with Lifecycle Services

Microsoft Dynamics Lifecycle Services (LCS) is a collaboration portal that provides an environment and a set of regularly updated services that can help you manage the lifecycle of your Finance and Operations implementations.

The lifecycle of an implementation spans many phases from pre-sales through Analysis, Design and Development, Test, and Deployment to Operation, possibly in multiple iterative roll-outs. It can last a few months to multiple years, based on the scope and complexity of the project and the chosen deployment model, for example, in the managed cloud or on-premises.

The management of the implementation involves many different stakeholders from the customer and partner organizations and, especially in the cloud-hosted deployment model, from Microsoft. The implementation is supported through tools provided on LCS and through processes defined within the Microsoft FastTrack and through the partner's implementation approach.

- Lifecycle Services resources
- Lifecycle Services (LCS) user guide

Deployment

You can deploy in the cloud or on-premises. Cloud deployments offer an enterprise resource planning (ERP) service that is fully managed by Microsoft. On-premises deployments are deployed locally in a customer's data center.

- Software lifecycle policy and cloud releases
- Cloud deployment overview
- System requirements for cloud deployments
- On-premises deployment home page
- System requirements for on-premises deployments

Upgrade

An upgrade can involve moving to a new product version, migrating and upgrading code, moving to an update, or deploying a hotfix.

Although the processes for each type of upgrade are similar, they differ enough that you should review the topics for a specific task before you begin.

- Upgrades, updates, and hotfixes resources

Database management

For information to help you move a database to new environment and restore a database to a specific point in time, see Database movement operations home page.

Security

Finance and Operation apps uses role-based security. Access is granted only to security roles, not to individual users. Users are assigned to roles. A user who is assigned to a security role has access to the set of privileges that is associated with that role. A user who isn't assigned to any role has no privileges.

Role-based security is aligned with the structure of the business. The security roles that a user is assigned to depend on the user's responsibilities in the organization, and their participation in business processes. The administrator grants access to the duties that users in a role perform, not to the program elements that users must use.

Because rules can be set up for automatic role assignment, the administrator doesn't have to be involved every time that a user's responsibilities change. After security roles and rules have been set up, business managers can control day-to-day user access, based on business data.

- Role-based security
- Security architecture
- Encryption in Finance and Operations apps

Batch processing

Many tasks can be run as part of batch jobs. For example, batch jobs can include tasks for printing reports, doing maintenance, or sending electronic documents. By using batch jobs, you can avoid slowing down your computer or the server during typical working hours.

- Batch processing overview
- Batch processing and batch servers

Optimization advisor

- Optimization advisor overview
- Optimization advisor (video)
- Create rules for Optimization advisor

Office integration

The integration with Microsoft Office provides a set of productive, collaborative, and integrated user experiences that take advantage of the Microsoft Office suite. This functionality can help your organization become more efficient and effective.

- Office integration overview

- Office integration tutorial
- Open entity data in Excel and update it by using the Excel add-in
- Create Open in Excel experiences
- Add templates to the Open lines in Excel menu
- Customize the Open in Microsoft Office menu
- Configure and send email
- Troubleshoot the Office integration

Mobile

The Finance and Operations mobile app enables your organization to make its business processes available on mobile devices. After you enable the mobile workspaces for your organization, users can sign in to the app and immediately begin to run business processes from their mobile devices.

- Mobile app home page
- Available mobile workspaces

Process Automation

The process automation framework allows administrators to view and create automated processes that will be scheduled with the batch server. The added layer of visibility of scheduled work is presented in a calendar view that can be extended for use in application areas to allow non-system administrator users to view work that impacts their area.

- Process Automation home page

General administration

- Demo data overview
- Cross-company data sharing
- Add links to your organization's legal terms and privacy statement
- License codes and configuration keys report
- Maintenance mode
- Preconfigured system accounts
- Export business-to-business (B2B) users to Azure Active Directory
- Set the session inactivity timeout
- Build OData metadata cache when AOS starts
- Configure and manage database logging

Add links to your organization's legal terms and privacy statement

2/18/2021 • 2 minutes to read • Edit Online

This topic explains how administrators can add links to their organization's legal terms and privacy statement in the **About** pane of Microsoft Dynamics 365 Finance, Supply Chain Management, and Commerce.

Organizations often need to ensure that the links to their legal terms and privacy statement are readily available and visible to users in order to meet legal and compliance requirements. Administrators of an organization can follow these steps to have the links to their legal terms and privacy statement be available in the **About** pane (**Settings** > **About**).

Add links

1. Go to the **System parameters** page and click **Legal and Privacy.** On this page:
    a. Enter the link to a page that outlines the legal terms for your organization.
    b. Enter the link to a page that outlines the privacy statement for your organization.

    > **NOTE**
    > Make sure that you enter the full URL, starting with either *https* or *http*.

2. Click **Save**.
3. If you are using Commerce, go to the **Distribution schedules** page. On this page:
    a. Select the **1110 – Global configuration** job.
    b. Click **Run now**.

    > **NOTE**
    > To verify that the job completed, go to the **Download sessions** page.

Validate links

Validate the links in Finance, Supply Chain Management, and Commerce

To validate that the links have been added, on the toolbar at the top of the page, click the **Settings** icon, and then click **About**. In the **Links** section of the pane, you should see two new links:

- **Your organization's Legal terms**
- **Your organization's Privacy and Cookies**

Click these links to validate that the appropriate pages open.

NOTE

The links open in a new window, so if you have a pop-up blocker enabled, you will need to add an exception to your pop-up blocker settings to launch a new window.

Validate the links in Modern Point of Sale (MPOS) and Cloud Point of Sale (CPOS)

To validate that the links have been added, go to the **Settings** page. In the **About** section, click the links to validate that the appropriate pages open.

License codes and configuration keys report

2/18/2021 • 2 minutes to read • Edit Online

This topic points you to a report that lists the license codes and configuration keys available in Finance and Operations.

When you purchase Finance and Operations, all functionality is included. By default, some features and functionality that you do not use may be enabled. The administrator should disable the features that are not needed by disabling license codes and configuration keys.

When a license code or configuration key is disabled, the associated module or feature is removed from the user interface. Large sets of functionality, such as modules, are controlled by license codes. Many license codes, in turn, enable configuration keys that allow you to enable and disable functionality at a more detailed level.

To view the report

The **License codes and configuration keys report**, included with the Technical reference reports, lists each configuration key that is available. The report also indicates the license code and menu items associated with each configuration key.

Cross-company data sharing

2/18/2021 • 5 minutes to read • Edit Online

This topic provides information about cross-company data sharing. Cross-company sharing is a mechanism for sharing reference and group data among companies in a Finance and Operations deployment. This feature resembles the virtual companies feature in Microsoft Dynamics AX 2012.

What is this feature and how does it work?

Cross-company data sharing lets you replicate (share) reference and group data among companies. Data integrity is verified before replication occurs.

Here are some examples of cross-company data sharing and the basic logic:

- The same payment terms and payment day definitions are used across 15 legal entities.
- The same terms of delivery are used across seven legal entities in three countries/regions.
- Records created, updated, and deleted in any of the companies within the policy will be replicated immediately, across all the companies.
- Fields that are not selected for sharing are maintained in each company and will not trigger any replication.
- As part of enabling a policy, it is optional to copy any existing records.

Cross-company data sharing has the following limitations:

- It can’t be used to share transactional data between companies.
- Only reference and group data can be shared, or tables that have specifically been enabled. For example, **Data Sharing Type** is set to **Duplicate**.
- It supports replication of fewer than one million total records per job. This total is calculated as the number of shared records × the number of shared companies. The limit is increased to two million records from the Platform Update for version 10.0.10.
- It supports replication for up to 100 companies per policy. The limit is increased to 300 companies from the Platform Update for version 10.0.10.
- Only one level of child relationships is exposed. To protect data consistency, replication doesn't occur if another level is required.
- Fields that reference Financial dimensions, for example **Ledger** or **Default** dimension, can't be shared across companies. o Dimensions hold a loose foreign key reference to the backing dimension data, which can reference both company-specific and non-company specific data. Determining the appropriate action to be taken for each dimension value has inherent complexity and would require a change from the current implementation, which could dramatically impact performance.
- It can’t be used with dual-write.

Policies

Data sharing is managed by defined policies that are saved in data packages. Templates that Microsoft has tested and supports are available as downloadable data packages on Microsoft Dynamics Lifecycle Services (LCS). Policies let you control the following aspects of data sharing:

- The fields that are replicated
- The entities that participate in the replication
- The companies that participates in the sharing

The same company and table can only be in one policy. It is possible to share the same table in more than policy.

This can happen when the limits of records or companies are reached, or to create policies for tables that need to be shared differently for different country/regions.

> **NOTE**
> Only required foreign key fields are selected by default. Optional foreign keys need to be selected manually to be included. The best practice is to add one or more tables when selecting a foreign key field, unless the table has already been added.

Policy templates that Microsoft has tested and supports are available as downloadable data packages on Lifecycle Services (LCS).

> **IMPORTANT**
> Although customers can modify the Microsoft data templates that are available from LCS, this scenario isn't supported.

Conflict resolution

Validation rules are run when a sharing policy is enabled. If inconsistencies are detected, the user who implements the system can choose which records from which company should win.

Considerations for successful data sharing

Several entities in the Microsoft data packages have references that you must consider when you enable the entities. Some data sharing policies can't be enabled if references don't match. Other policies can be enabled, but you should use the Find inconsistency checker tool to verify that your data is consistent. Here are some examples:

- The Production group sharing policy has a reference to a company's chart of accounts. Therefore, all companies that are added to this sharing policy must use the same chart of accounts.
- If you want to enable entities that use number sequences, the number sequence types must be the same across all companies in a sharing policy for those entities.
- Setup options must be the same across the companies that are involved in the sharing policy. Examples of setup options include the setting that specifies whether tax is included by default.

When should I use cross-company data sharing?

Use cross-company data sharing for the following business scenarios:

- Sharing of simple reference and group data in a single deployment
- Sharing among companies that have very similar configurations
- Sharing scenarios that have been explicitly tested by Microsoft

Cross-company data sharing isn't supported for the following scenarios:

- Franchising solutions, where thousands of records are shared across thousands of companies.
- Sharing of transactional records for reporting or management purposes, such as consolidations.
- Sharing across deployments.
- Complex scenarios, such as replication of subtype/supertype tables or tables that have date effectivity rules.
- Tables that do not have a unique index.

Customer and vendor master data sharing

Customer and vendor master data sharing allows you to share customer and vendor data across multiple companies. If you would like to be considered for this feature, complete the Data sharing application and contact

Support.

With the release of Platform update for version 10.0.12, customer and vendor master data sharing can be enabled using the **Customer and vendor master data sharing** feature in the **Feature management** module. There is no need to complete a survey first. It is important to consider limits in the number of records and companies stated above.

> **NOTE**
> Default dimensions set up against a customer or vendor cannot be shared across companies. When configuring the customer or vendor record for cross-company data sharing, the **DefaultDimension** field is disabled, and cannot be included in the data sharing policy.

> Default dimensions hold a loose foreign key reference to the backing dimension data, which can reference both company-specific and non-company specific data. Determining the appropriate action to be taken for each dimension value has inherent complexity and would require a change from the current implementation, which could dramatically impact performance.

Download a cross-company data sharing template from LCS

1. Sign in to LCS.
2. On the home page, click **Shared asset library**.
3. In the **Asset type** list, click **Data package**.
4. Click any of the available data package files to download them.

For details about how to use a template, see Configure financial cross-company data sharing.

Currently supported cross-company data sharing templates

PACKAGE NAME ON LCS	DATA SHARING POLICIES
Financial data sharing templates	• Bank parameters • Ledger journal names • Payment days • Payment schedules • Payment terms • Tax exempt codes

PACKAGE NAME ON LCS	DATA SHARING POLICIES
Supply chain data sharing templates	• Barcode parameters • Barcode setup • Buyer group • Charges group • Commission • Destination code • Non-conformance type • Order entry deadline group • Order origin code • Order pool • Production group • Production pool • Reason for delivery • Supplementary item group • Terms of delivery • Work time calendar

Additional resources

Configure financial cross-company data sharing (Task guide)

Maintenance mode

2/18/2021 • 3 minutes to read • Edit Online

This topic provides information about maintenance mode in Finance and Operations. When maintenance mode is turned on, it provides a safe way for system administrators to make system changes that might affect system functionality. For example, configuration keys can be enabled or disabled. While maintenance mode is on, only system administrators and users who have the **Maintenance mode user** role can sign in to the system. By default, maintenance mode is turned off. When maintenance mode is off, you can't edit the **License configuration** page.

Turn maintenance mode on and off on sandbox and production environments through Lifecycle Services

You can now turn maintenance mode on and off directly through Lifecycle Services (LCS) on your sandbox and production environments. Refer to the following steps to do this:

1. Go to the environment details page and on the **Maintain** menu, click **Enable Maintenance Mode**.
2. In the slider, set **Turn maintenance mode on** for the environment and select **Confirm**.
3. A servicing operation will begin and your system will go into maintenance mode.
4. On completion, the environment state will be **In Maintenance**. At this point, only the system administrator will have access to the environment.
5. After you are done making system-wide changes, you can turn off maintenance mode by clicking **Disable Maintenance Mode** under the **Maintain** menu.
6. This will start a servicing operation that takes your environment out of maintenance mode. You can see the progress of the operation in the environment details page.
7. After this is complete, your environment goes back to the **Deployed** state. Now all users can sign in to the environment.
8. You can check the environment history page to see when the maintenance mode was turned on or turned off. To get to the environment history page, select **History** and **Environment changes** on the environment details page.

Turning maintenance mode on and off for your sandbox and production environment is very similar to a servicing operation. If turning maintenance mode on or off fails, you will see options such as **Resume**, **Rollback**, and **Abort**. You also have the option to **download the logs** to troubleshoot why the operation failed.

Turn maintenance mode on and off in DevTest/Demo environments hosted in a Microsoft subscription

1. Establish an RDP connection to the developer machine.
2. On the developer machine, sign in to SQL Server by using the credentials for the axdbadmin user from LCS. Then switch to the AXDB database, and run the following command.

   ```
   update SQLSYSTEMVARIABLES SET VALUE = 1 where PARM = 'CONFIGURATIONMODE'
   ```

3. Restart the **World Wide Web Publishing Service** to reset IIS.
4. After the service is restarted, the system will be in maintenance mode.

5. When you've completed your maintenance mode activities, repeat steps 2 and 3, but set the value to 0 in step 2.

Turn maintenance mode on and off in DevTest/Demo environments and VHD-based environments hosted in customers' subscription

You can turn on maintenance mode locally by running the following command.

NOTE

On some virtual machines (VMs), the exact location of the Deployment.Setup.exe tool might differ. Check AosServiceWebRootbin.

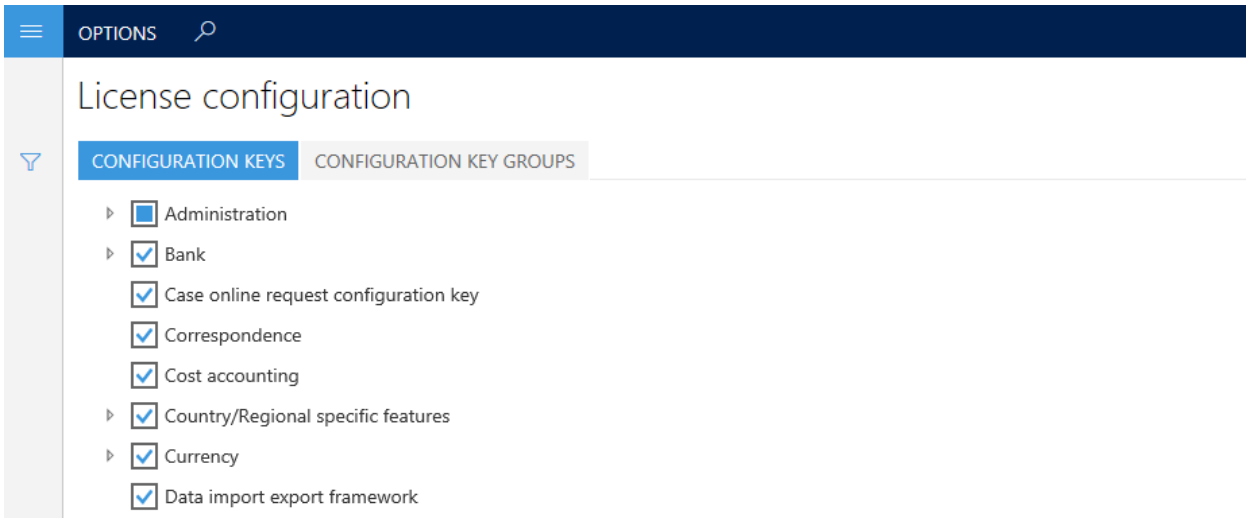
```
J:\AosService\PackagesLocalDirectory\Bin\Microsoft.Dynamics.AX.Deployment.Setup.exe --metadatadir
J:\AosService\PackagesLocalDirectory --bindir
J:\AosService\PackagesLocalDirectory\Bin --sqlserver . --sqldatabase axdb --sqluser axdbadmin --sqlpwd
********* --setupmode maintenancemode --isinmaintenancemode true
```

The following table describes the parameters that are used in this command.

PARAMETER NAME	DESCRIPTION
--setupmode maintenancemode	Use this parameter to inform the setup tool that the system will be put into or taken out of maintenance mode.
--metadatadir	Use this parameter to specify the metadata directory. You should use the default packages directory.
--bindir	Use this parameter to specify the binaries directory. You should use the default packages directory.
--sqlserver	Use this parameter to specify the Microsoft SQL Server. For one-box environments, use a period (.).
--sqluser	Use this parameter to specify the SQL Server user. You should use **AOSUser**.
--sqlpwd	Use this parameter to specify the SQL Server password.
--isinmaintenancemode	Use this parameter to turn configuration mode on or off. Use **true** to turn it on and **false** to turn it off.

Enable (or disable) configuration keys

After the instance of Application Object Server (AOS) is restarted, the system will be in maintenance mode. You can then enable configuration keys, as shown in the following screenshot.



If you try to access the system while in maintenance mode, but you aren't a system administrator or a user who has the **Maintenance mode user** role, you may receive an error message.

You can turn off maintenance mode by running the following command.

```
J:\AosService\PackagesLocalDirectory\Bin\Microsoft.Dynamics.AX.Deployment.Setup.exe --metadatadir
J:\AosService\PackagesLocalDirectory --bindir J:\AosService\PackagesLocalDirectory\Bin --sqlserver . --
sqldatabase axdb --sqluser axdbadmin --sqlpwd ********* --setupmode maintenancemode --isinmaintenancemode
false
```

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Preconfigured system accounts

2/18/2021 • 2 minutes to read • Edit Online

Pre-configured system accounts are included on deployed environments so that Microsoft can manage and operate the Finance and Operations service and provide specific features to customers. The following table provides information about each account, including the purpose and use case for the account.

IMPORTANT

Do not delete these system accounts. Deleting these accounts will cause a disruption in key functionality provided by Microsoft.

ACCOUNT DETAIL	PURPOSE/USE CASE OF THE ACCOUNT
`Axrunner`	This account is used to monitor the health of the environment and provide alerts when necessary.
`FRServiceUser`	This account is the Financial Reporting service user account, which is used by the Management Reporter application for integrations with Finance and Operations.
`RetailServiceAccount`	This account is used for Retail services to connect to the Finance and Operations environment.
`SysHealthServiceUser` or `Axping` (depending on the deployed product version)	This account is used to monitor the availability and health of the environment and provide alerts when necessary.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Export business-to-business (B2B) users to Azure Active Directory

2/18/2021 • 2 minutes to read • Edit Online

You can automatically export business-to-business (B2B) users to Azure Active Directory (Azure AD).

In the past, B2B users were exported manually to a .csv file. Then the Azure AD tenant administrator had to use this file to manually add the users to Azure AD using the Azure portal.

To enable the automatic export feature, a one-time setup and configuration process must be completed. When the process is completed, you can use the **Provision Azure AD B2B user** workflow task to automatically export B2B users to Azure AD.

The one-time set up and configuration means that you'll need to:

1. Set up a B2B invitation service application in Azure AD.
2. Configure the B2B invitation service settings in Finance and Operations.

Set up a B2B invitation service application in Azure AD

The tenant administrator of your Azure AD tenant will need to complete the following steps.

1. Log on to the Azure portal as the tenant administrator.
2. Click **Azure Active Directory** > **Properties**.
3. Copy the **Directory ID** (this is the tenant ID) and save it. You will need this later.
4. Click **App registrations** > **New application registration**.
5. Enter the following information, and then click **Create**.
   a. In the **Name** field, enter the name of the application. For example: **B2B admin application**.
   b. In the **Application type** field, select **Web app /API**.
   c. In the **Sign-on URL** field, enter the URL for Finance and Operations.
6. Click the **App registrations** tab, click the newly created application, copy the **Application ID**, and save it. You will need this later.
7. Click **All settings** > **Required permissions** > **Add**.
8. In the **Add API access** pane, do the following:
   a. Click the **Select an API** tab. Click **Microsoft Graph**, and then click **Select**.
   b. In the **Select permissions** tab, select the following **application permissions** and set them to **Yes**:
      - **Invite guest users to the organization**
      - **Read and write directory data**
      - **Read and write all users' full profiles**
   c. Select the following **delegated permissions** and set them to **Yes**:
      - **Invite guest users to the organization**
      - **Read and write directory data**
      - **Read and write all users' full profiles**

- **Sign in and read user profile**

d. Click **Select** and **Done**.

9. In the **Required permissions** blade, click **Grant Permissions**, and then click **Yes** to assign the permissions.
10. Click **All settings** > **Keys**, and then do the following:
    a. Enter a name of the key in the **Description** field.
    b. Set the expiration duration in the **Expires** field.
11. Click **Save**. Saving the key will display the **Value**.

> **WARNING**
> Be sure to copy the key **Value** after saving the key. This value will not be available when you leave the blade.

Configure the B2B invitation service settings

1. Sign in to Finance and Operations as administrator.
2. Navigate to the **B2B Invitation Configuration** page, and click **Edit**.
3. Select **Enabled**.
4. Verify that the **Tenant ID** is the same as the **Directory ID** (which you noted in step 3 of the previous procedure).
5. In the **Client ID** field, enter the **Application ID** (which you noted in step 6 of the previous procedure).
6. Enter the key **Value**, copied from the above procedure, into the **Application Key** field.
7. **Save** the settings.

Now you can start using the **Provision Azure AD B2B user** workflow task in your workflows to automatically export B2B users to Azure AD.

> **NOTE**
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Role-based security

2/18/2021 • 5 minutes to read • Edit Online

This topic provides an overview of the elements of role-based security in Finance and Operations.

In role-based security, access is not granted to individual users, only to security roles. Users are assigned to roles. A user who is assigned to a security role has access to the set of privileges that is associated with that role. A user who is not assigned to any role has no privileges.

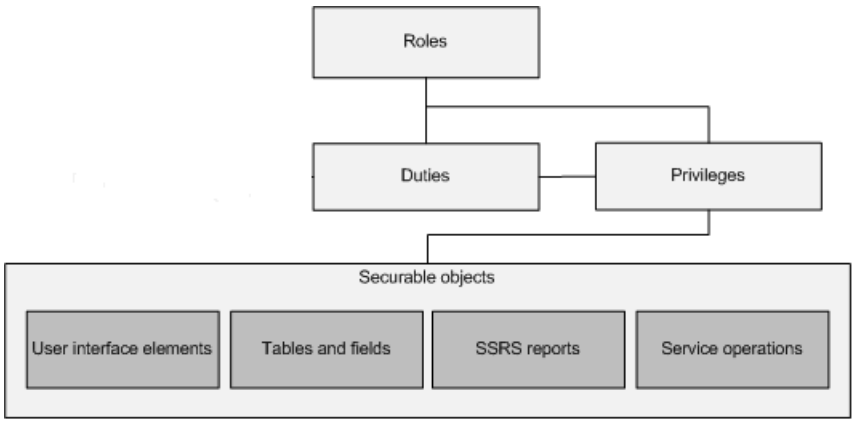
In Finance and Operations apps, role-based security is aligned with the structure of the business. Users are assigned to security roles based on their responsibilities in the organization and their participation in business processes. The administrator grants access to the duties that users in a role perform, not to the program elements that users must use.

Because rules can be set up for automatic role assignment, the administrator does not have to be involved every time that a user's responsibilities change. After security roles and rules have been set up, business managers can control day-to-day user access based on business data.

Overview of role-based security

This section provides an overview of the elements of role-based security. The security model is hierarchical, and each element in the hierarchy represents a different level of detail. Permissions represent access to individual securable objects, such as menu items and tables. Privileges are composed of permissions and represent access to tasks, such as canceling payments and processing deposits. Duties are composed of privileges and represent parts of a business process, such as maintaining bank transactions. Both duties and privileges can be assigned to roles to grant access to Finance and Operations.

The following illustration shows the elements of role-based security and their relationships.



Security roles

All users must be assigned to at least one security role in order to have access to Finance and Operations. The security roles that are assigned to a user determine the duties that the user can perform and the parts of the user interface that the user can view.

Administrators can apply data security policies to limit the data that the users in a role have access to. For example, a user in a role may have access to data only from a single organization. The administrator can also specify the level of access that the users in a role have to current, past, and future records. For example, users in a role can be assigned privileges that allow them to view records for all periods, but that allow them to modify records only for the current period.

By managing access through security roles, administrators save time because they do not have to manage access separately for each user. Security roles are defined one time for all organizations. In addition, users can be automatically assigned to roles based on business data. For example, the administrator can set up a rule that associates a Human resources position with a security role. Any time that users are assigned to that position, those users are automatically added to the appropriate security roles.

Security roles can be organized into a hierarchy. The role hierarchy allows the administrator to define a role based on another role. For example, the sales manager role could be defined as a parent role of the manager role and the salesperson role. A parent role automatically inherits the duties, privileges, and conditions that are assigned to its child roles. Therefore, a user who is assigned to the parent role can perform all of the tasks that users in the child roles can perform. A role can have one or more child roles or one or more parent roles.

By default, sample security roles are provided. All functionality is associated with at least one of the sample security roles. The administrator can assign users to the sample security roles, modify the sample security roles to fit the needs of the business, or create new security roles. By default, the sample roles are not arranged in a hierarchy.

Duties

Duties correspond to parts of a business process. The administrator assigns duties to security roles. A duty can be assigned to more than one role.

In the security model, duties contain privileges. For example, the **Maintain bank transactions** duty contains the **Generate deposit slips** and **Cancel payments** privileges. Although both duties and privileges can be assigned to security roles, we recommend that you use duties to grant access to Finance and Operations.

You can assign related duties to separate roles. These duties are said to be segregated. By segregating duties, you can better comply with regulatory requirements, such as those from Sarbanes-Oxley (SOX), International Financial Reporting Standards (IFRS), and the United States Food and Drug Administration (FDA). In addition, segregation of duties helps reduce the risk of fraud, and helps you detect errors or irregularities.

Default duties are provided. The administrator can modify the privileges that are associated with a duty, or create new duties.

Privileges

In the security model, a privilege specifies the level of access that is required to perform a job, solve a problem, or complete an assignment. Privileges can be assigned directly to roles, however we recommend that you only assign duties to roles. This is so that the privileges are first grouped together into a duty, which makes it easier to maintain.

A privilege contains permissions to individual application objects, such as user interface elements and tables. For example, the **Cancel payments** privilege contains permissions to the menu items, fields, and tables that are required to cancel payments.

By default, privileges are provided for all features in Finance and Operations. The administrator can modify the permissions that are associated with a privilege, or create new privileges.

Permissions

Each function, such as a form or a service, is accessed through an entry point. Menu items, web content items, and service operations are referred to collectively as entry points.

In the security model, permissions group the securable objects and access levels that are required to run a function. This includes any tables, fields, forms, or server side methods that are accessed through the entry point.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

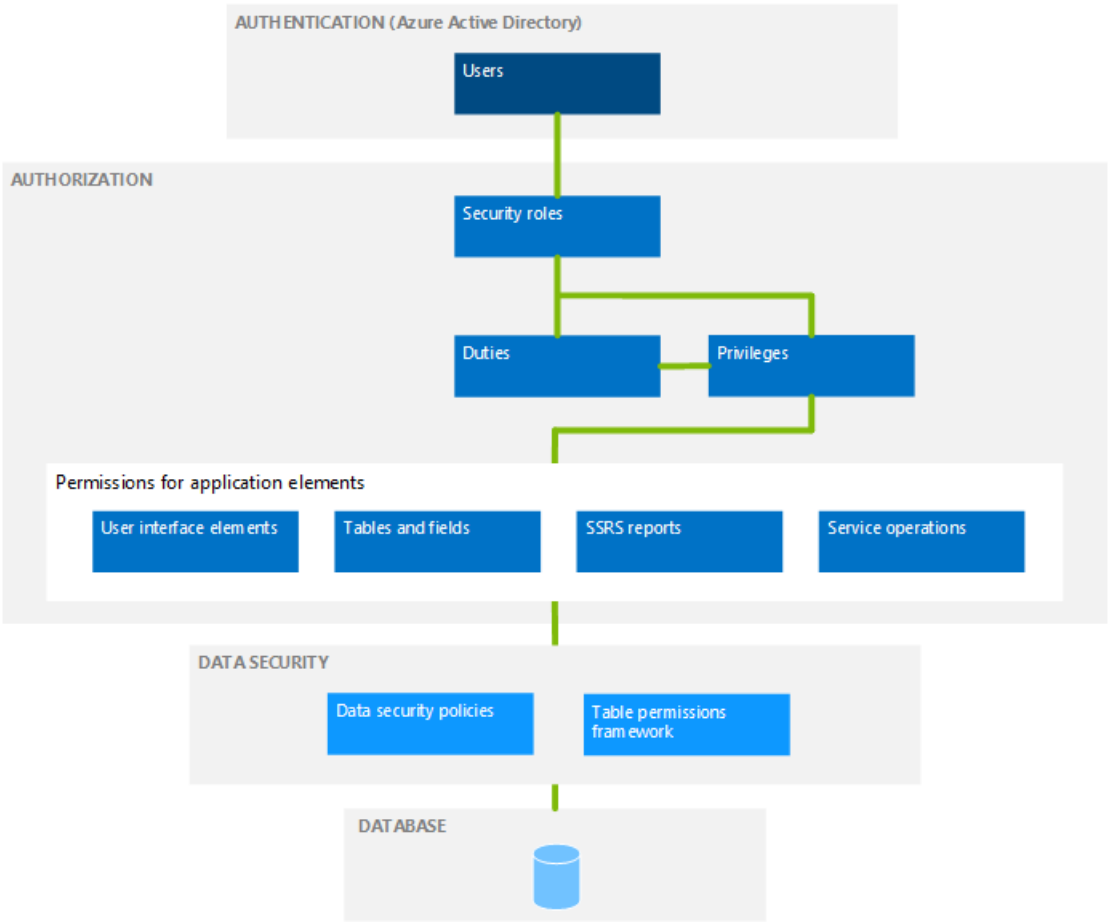
The survey will take about seven minutes. No personal data is collected (privacy statement).

Security architecture

2/18/2021 • 2 minutes to read • Edit Online

This topic provides an overview of the security architecture of Finance and Operations.

When you understand the security architecture, you can more easily customize security to fit the requirements of your business. The following diagram provides a high-level overview of the security architecture.



Authentication

By default, only authenticated users who have user rights can establish a connection.

Microsoft Azure Active Directory (AAD) is a primary identity provider. To access the system, users must be provisioned into a Finance and Operations instance and should have a valid AAD account in an authorized tenant.

Authorization

Authorization is the control of access to Finance and Operations applications. Security permissions are used to control access to individual elements of the program: menus, menu items, action and command buttons, reports, service operations, web URL menu items, web controls, and fields in the Finance and Operations client.

Individual security permissions are combined into privileges, and privileges are combined into duties. The administrator grants security roles access to the program by assigning duties and privileges to those roles.

Context-based security controls access to securable objects. When a privilege is associated with an entry point (such as a menu item or a service operation), a level of access, such as **Read** or **Delete**, is specified. The

authorization subsystem detects the access at run time, when that entry point is accessed, and applies the specified level of access to the securable object that the entry point leads to. This functionality helps to ensure that there is no over-permissioning, and the developer gets the access that was intended.

For more information, see Role-based security.

Data security

Authorization is used to grant access to elements of the program. By contrast, data security is used to deny access to tables, fields, and rows in the database.

Use the extensible data security framework to supplement role-based security by restricting access to table records based on security policies. A security permission, as part of a user role, increases the access a user has to data, while a security policy decreases access to data.

For more information, see Extensible data security policies.

Additionally, the Table Permissions Framework helps protect some data. Data security for specific tables is enforced by Application Object Server (AOS).

Auditing

Auditing of user sign in and sign out is enabled, which means that the system logs when a user signs in or out of the application. A sign out is logged even if the user's session expires or ends.

A system administrator or security administrator can access the audit logs by going to the **User log** page (**System administration** > **Inquiries** > **User log**).

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Encryption in Finance and Operations apps

2/18/2021 • 2 minutes to read • Edit Online

Encryption at rest

Microsoft uses encryption technology to protect customer data while at rest in an environment's SQL Server database and Azure Storage.

All instances utilize Microsoft SQL Server Transparent Data Encryption (TDE) and Azure Storage encryption to perform real-time encryption of data when written to the disk at rest.

Finance and Operations apps use server-side encryption using service-managed keys. All key management aspects such as key issuance, rotation, and backup are handled by Microsoft.

In addition to the default encryption at rest provided above, you can use the encryption API available in the **Global** X++ class. The methods **Global::editEncryptedField()** and **Global::editEncryptedStringField()** use the environment-specific data encryption certificate to perform data encryption and decryption. You can use these methods as an additional layer of protection beyond the default encryption at rest technology used for data storage.

Encryption in transit

Connections established between customers and Microsoft datacenters are encrypted, and all public endpoints are secured using industry-standard Transport Layer Security (TLS) 1.2. TLS effectively establishes a security-enhanced browser-to-server connection to help ensure data confidentiality and integrity between desktops and datacenters.

Supported TLS versions

Finance and Operations apps support TLS 1.2 only. Earlier TLS versions, 1.0 and 1.1, are not supported.

Supported cipher suites

Finance and Operations apps only support the following cipher suites:

- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256

Additional resources

- Azure Data Encryption-at-Rest
- Microsoft SQL Server Transparent Data Encryption (TDE)
- Azure Storage encryption
- Insider tips on development

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Set the session inactivity timeout

2/18/2021 • 2 minutes to read • Edit Online

> **IMPORTANT**
>
> Some or all of the functionality noted in this topic is available as part of a preview release. The content and the functionality are subject to change. For more information about preview releases, see Service update availability.

The session inactivity timeout setting represents the amount of time a user can be inactive before the user's session times out and closes. It only affects user browser sessions.

You can set the values from 5 minutes to 60 minutes.

This function has a default value of 30 minutes. You can set the value up to 60 minutes, however doing so might cause extra load on the system.

> **NOTE**
>
> This feature is available as of Platform update 29.
>
> If you previously set a session inactivity timeout in the web.config (**WebClientStatefulSessionTimeoutInSeconds** key) through a support request, then that old value will still be honored. The change in default will only affect those who had not explicitly set a new session inactivity timeout in the web config.

To change the value, follow these steps:

1. Select **System administration > Setup > System parameters** to open the **System parameters** page.
2. On the **General** tab, in the **Session management** section, enter a value in the **Session inactivity timeout in minutes** field.
3. Select **Save**.

   If you set the value to greater than 30, you will be prompted to confirm your selection. The confirmation prompt says "Increasing the inactivity session timeout can cause extra load on your system, which can lead to a decrease in performance. Are you sure you want to continue?" The higher the value, the higher the load will be, which can affect negatively system performance. Select **Yes** to save the changes, or **No** to revert to the existing value.

Alerting users before sessions end due to inactivity

To give users awareness of an impending session suspension due to inactivity and to help prevent users from losing any unsaved changes when this occurs, users will be notified before their sessions are set to be terminated due to inactivity and given an opportunity to reconnect. The notice given to the user is dependent on the **Session inactivity timeout** setting.

- If the **Session inactivity timeout** is more than 30 minutes, the user will see a countdown notification starting **5 minutes** before the session is set to close.
- If the **Session inactivity timeout** is between 10 and 30 minutes, the user will see a countdown notification starting **2 minutes** before the session is set to close.
- If the **Session inactivity timeout** is less than 10 minutes, the user will see a countdown notification starting **30 seconds** before the session is set to close.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Out-of-box security reports

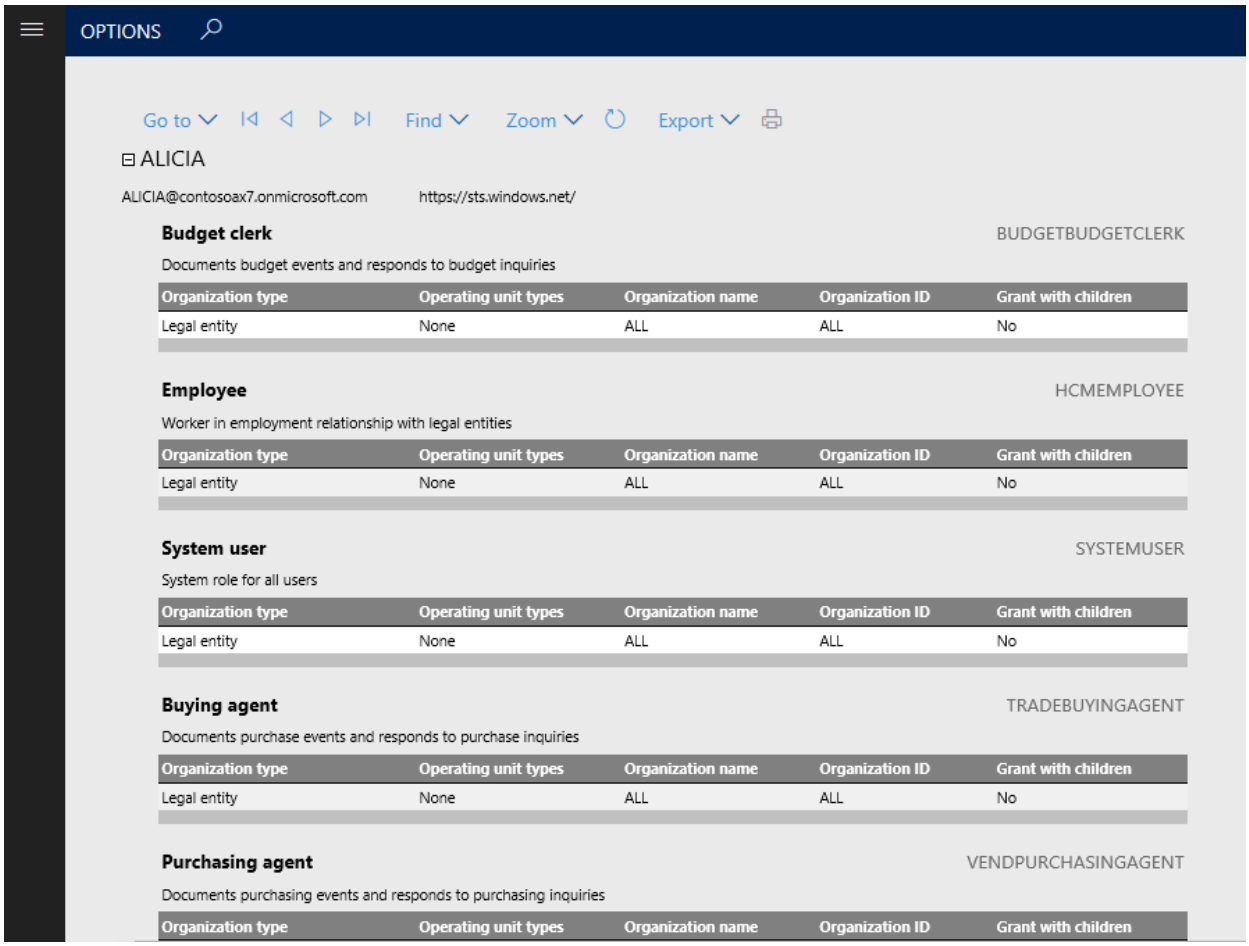
2/18/2021 • 3 minutes to read • Edit Online

Finance and Operations provides a set of rich security reports to help you understand the set of security roles running in your environment and the set of users assigned to each role. In addition to the reports noted in this topic, developers can generate a workbook containing all user security privileges for all roles using **Visual Studio > Dynamics 365 > Addins > View related objects and licenses for all roles**.

Each of the security reports can be found under **System administration > Inquiries > Security.** A description of each report is provided below.

User role assignments

The **User role assignments** report generates a view of the current user role assignments in your system. By default, the report includes all users with roles assigned. You can optionally limit the report to a specific set of users by entering a list of users when generating the report. On the **User role assignments** parameters pane, go to **Records to include** > **Filter.** From here you can add or remove filters to the list of users the report will be generated for.

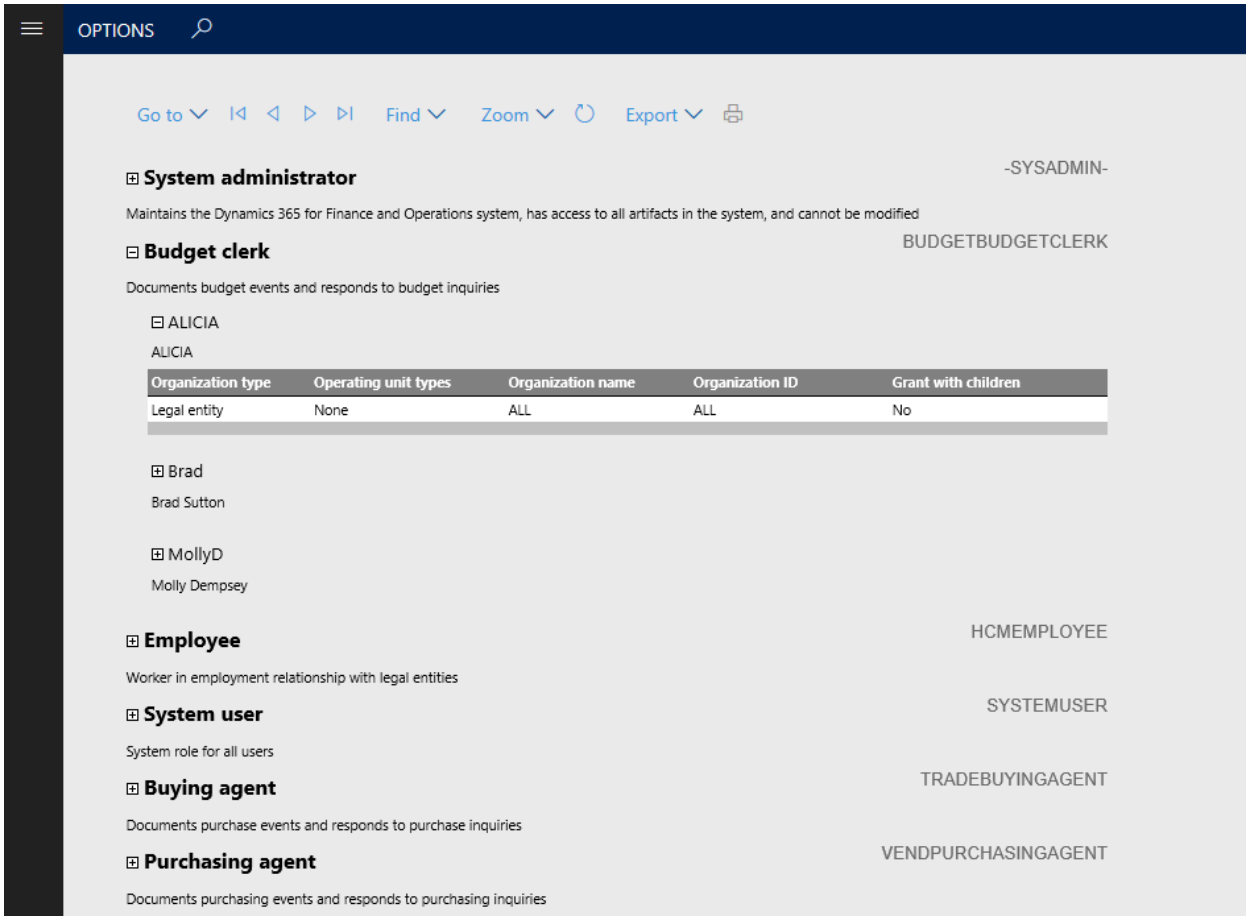


For each user in the report a list of roles is provided, along with any restrictions at the legal entity or organization level.

Role to user assignments

The **Role to user assignment** report provides an aggregation of role assignments. Expanding a role in the

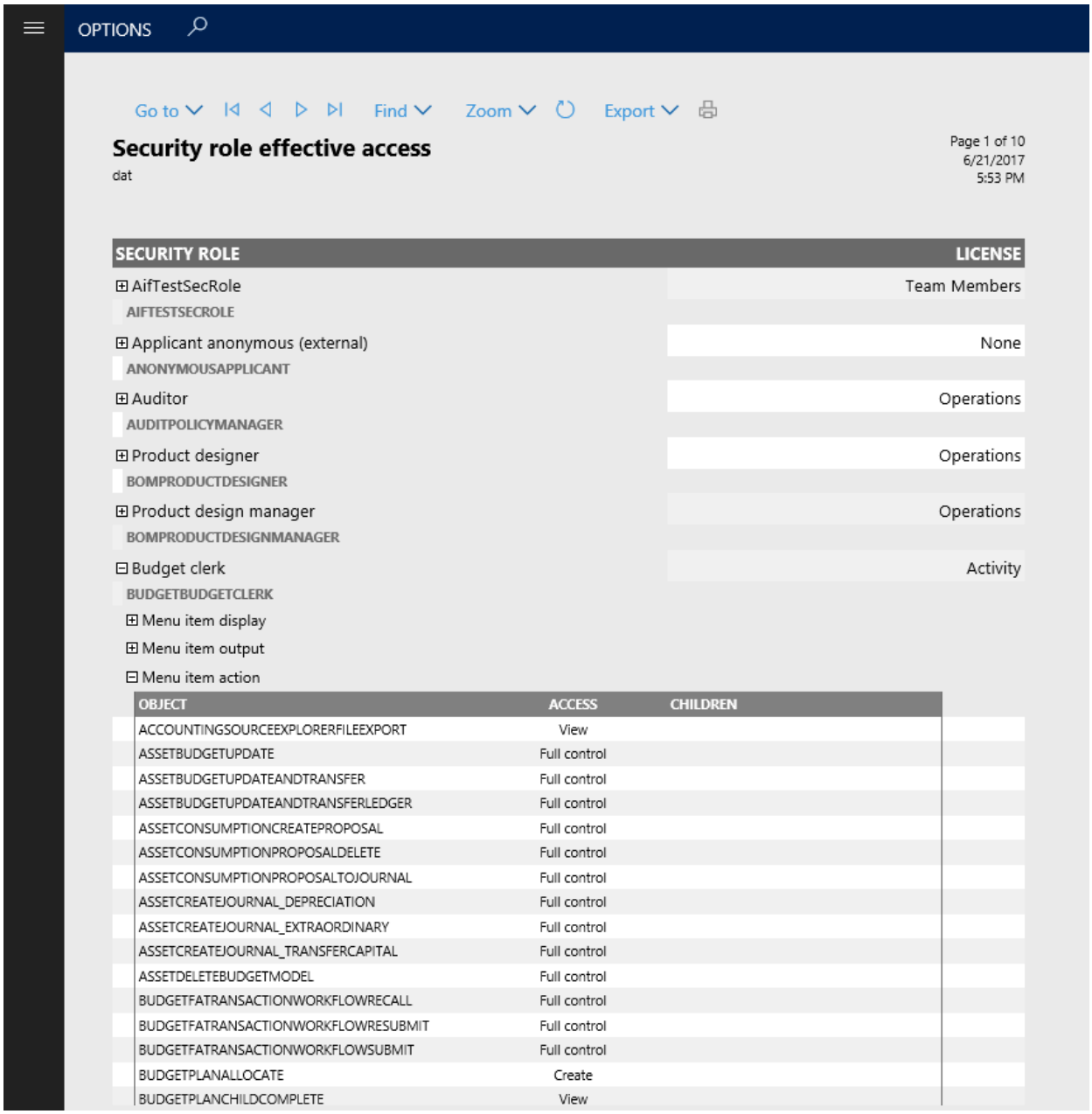
report shows the list of users assigned to the role, and expanding the user name shows any restrictions the role has applied. The same method for filtering the set of users can be applied to this report as described for the **User role assignments** report.



Security role access

The **Security role access** report provides a view of the effective permissions for each security role. This report provides a flattened list of permissions grouped by type across all sub-roles, duties, and privileges contained in the role.

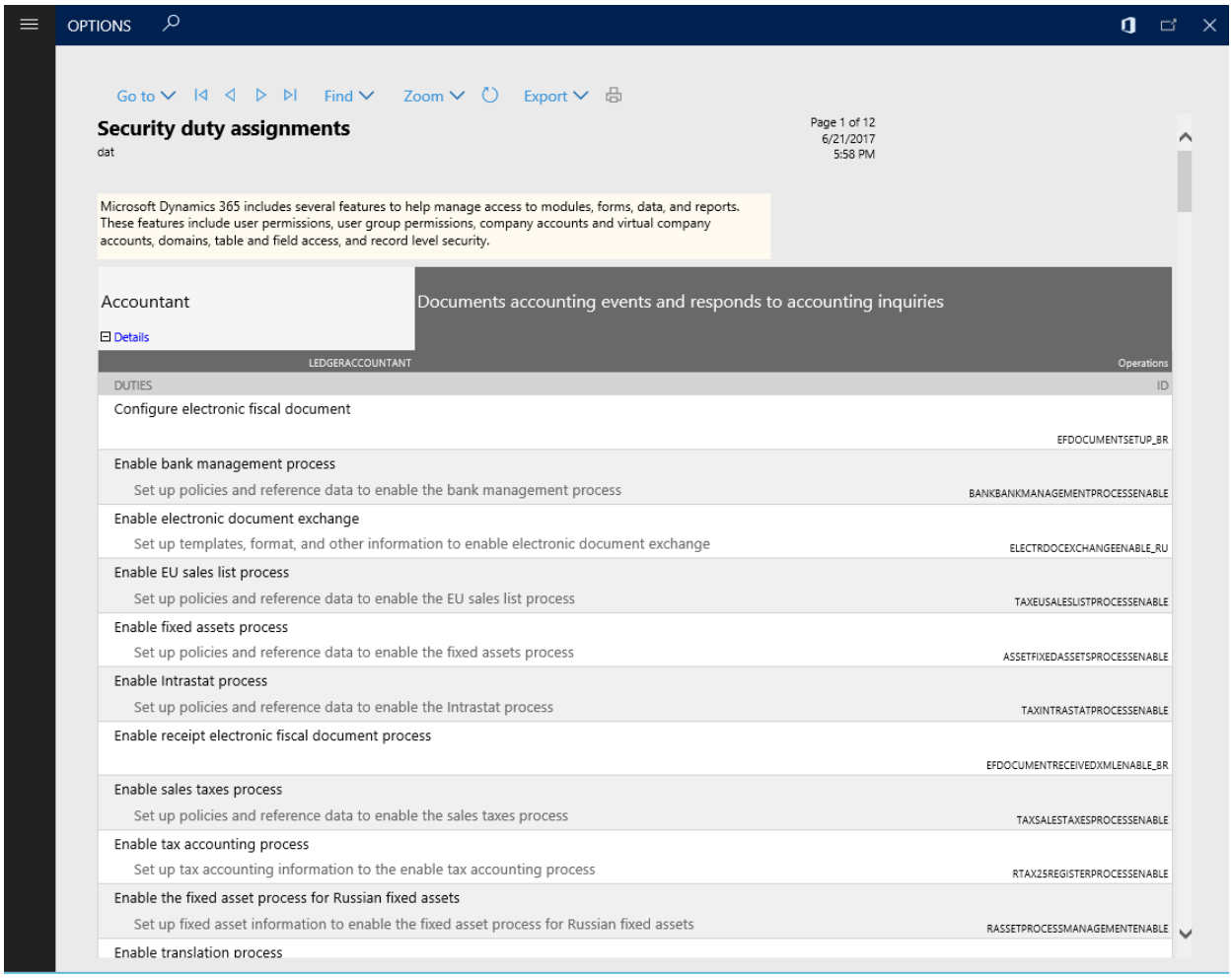
The data set backing the **Security role access** report can be very large, causing the report to take some time to run. If there have been no changes to security roles since the last time the report was run, you can skip building the report by setting the **Rebuild collection** option to **No** on the report parameters pane. This will render the report from the existing data set. If it is the first time the report has run, or there could be changes to the role definitions, the **Rebuild collection** option should be set to **Yes**. You can optionally limit the roles to be included in the report by adding a filter under **Records to include**.



Expanding a role shows the category of objects the role has access to. Expanding one of the object types will show a detailed list of each object of that type included in the role.

Security duty assignments

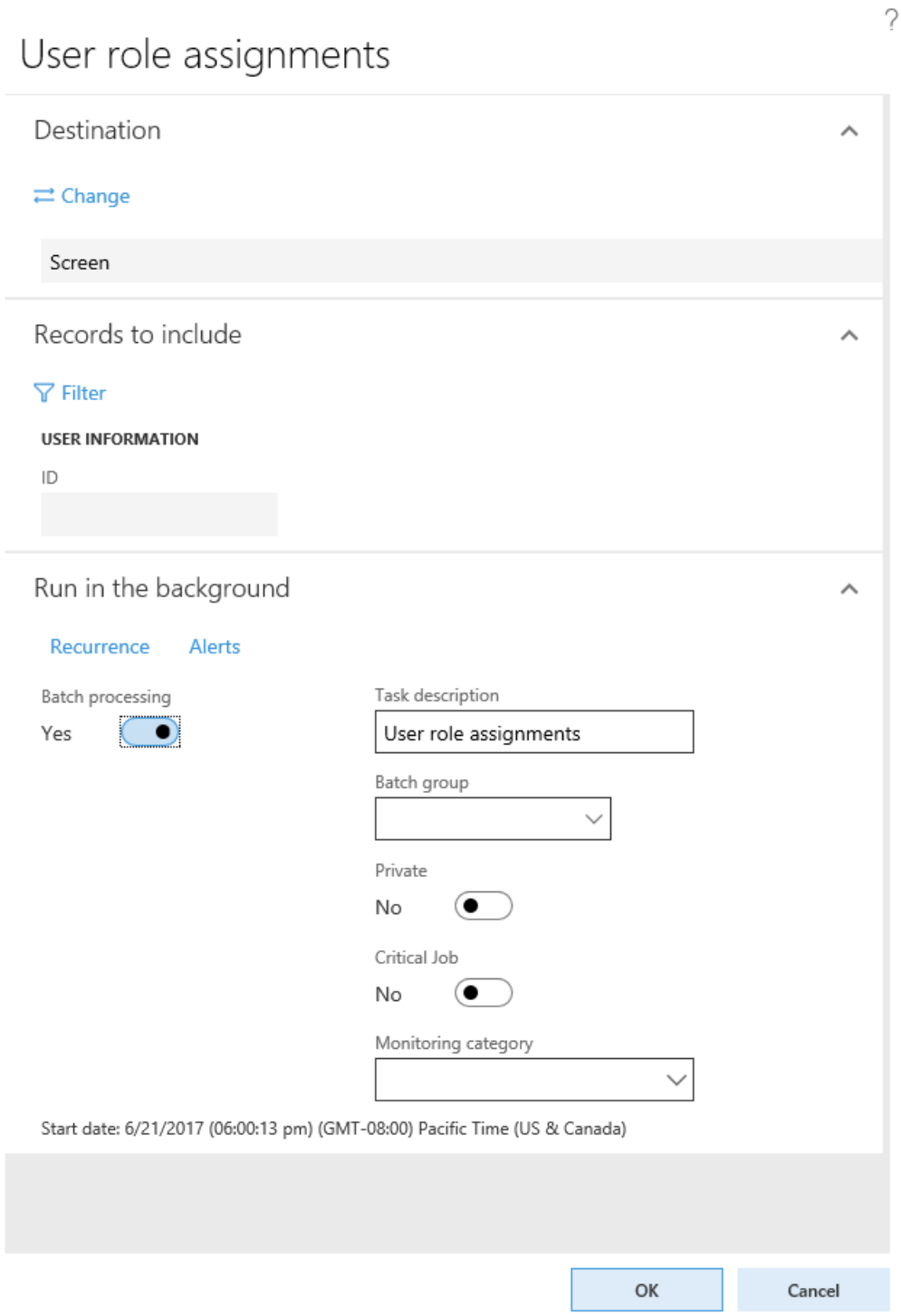
The **Security duty assignments** report provides a view of all the duties contained within a role. This report can be configured to run on any collection of roles to ensure that segregation of duties is maintained between roles. By default, the report will include all roles. To limit the roles included, leverage the filtering provided in the **Records to include** section.



Expanding a role in the **Security duty assignments** report will show each duty assigned to the role, along with details of the duty.

Batch processing of reports

Any of the above reports can be set to run as a batch job by going to the **Run in the background** section of the report's parameter pane. Set **Batch processing** to **Yes**, then provide a batch task job name, batch group, and whether the job should run as Private or Critical. The report will then be created when the batch task runs.



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Create new users

2/18/2021 • 3 minutes to read • Edit Online

Before you can access Finance and Operations apps, you must first be added to the **Users** page (**System administration** > **Users** > **Users**). Users include internal employees of your organization, or external customers and vendors. Users can be imported or added manually. All users must be correctly licensed for compliant use.

For information about how to buy and license for Finance and Operations apps, see Microsoft Dynamics 365 Licensing Guide.

Assign a license to a user

System admins can assign licenses to users in the Microsoft 365 admin center.

Add an external user in Azure AD and assign a license

External users must be represented in your tenant directory (Azure Active Directory (Azure AD)) so that they can be assigned licenses. Those external users should be added to the tenant in Azure AD as guest users and then assigned the appropriate licenses. A requirement for Finance and Operations apps is that the guest user's company must use Azure AD. For more information, see Add Azure Active Directory B2B collaboration users in the Azure portal.

Import new users from Azure AD

1. Go to **System administration** > **User** > **Users**.
2. On the Action Pane, select **Import users**.
3. Select the users to be imported. The list includes Azure AD users that are currently not users in this environment.
4. Select **Import users**.
5. Select **Close**.

> **NOTE**
>
> The value for the **Company** field will be set based on the current session company for the admin. After import, you must assign roles and organizations as applicable. For more information, see Assign users to security roles. Conditionally, it might also be required to associate the user with a **Person** and to update user options such as language.

Manually add a new user

1. Go to **System administration** > **Users** > **Users**.
2. On the Action Pane, select **New**.
3. In the **User ID** field, enter a unique identifier for the user.
4. In the **User name** field, enter the user's name.
5. In the **Provider** field:

- For internal users, use the defaulted value. For example, your Azure AD tenant prefixed with https://sts.windows.net/.
- For non-Azure AD users, such as Service-2-Service accounts, enter a basic text value. For example, NA. This

value will help avoid incorrect authentication calls that might result in errors if a valid identity provider value is used.

- For external or guest users, add their Azure AD tenant name after https://sts.windows.net/.

6. In the **Email** field, enter the user's full Email/User Principle Name.
7. In the **Company** field, select the default startup company for the user.
8. Select **Save**.

The values for Identity provider and Telemetry ID will be updated based on a Microsoft graph call, when the user record is saved. The Telemetry ID is based on the user's Object ID/Security Identifier (SID) in Azure AD.

> **NOTE**
>
> After you add a user, you must assign roles and organizations as applicable. For more information, see Assign users to security roles. Conditionally, it might also be required to associate the user with a **Person** and to update **User options** such as language.

Change a user ID

To change a user ID, you must rename the key in the database. When you change a user ID by using this procedure, all related user settings are modified to use the new user ID. For example, the usage information in the **SysLastValue** table is updated to reference the new user ID.

> **NOTE**
>
> The user ID is the primary key of the user information table. Renaming the primary key can take some time for existing users because all references to the key are also updated in the database.

1. Go to **System administration > Users > Users**.
2. Select a user in the list and select **Options> Record info**.
3. Select **Rename**.
4. Enter a new and unique value for the User ID, and then select **OK**.
5. Select **Yes** to confirm.

Additional resources

For more options to implement B2B users, see Export B2B users to Azure AD.

For information about preconfigured system accounts, see Preconfigured system accounts

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

User session management

2/18/2021 • 2 minutes to read • Edit Online

The user session setting represents the amount of time a user can be signed in before the user's session expires. After the user's session expires, the user is required to sign in with their credentials.

The **Maximum session length** can be up to 2,160 hours (90 days), with a minimum of 1 hour.

> **NOTE**
> This feature is available in public preview as of version 10.0.16. To enable this feature, go to the Feature management page and enable the **(Preview) Enable session management for users** feature.

To change the maximum session length, follow these steps:

1. Go to**System administration > Users > User session management**.
2. Select **New**.
3. In the new row, select the drop-down menu in the **User ID** field.
4. In the user list, select a user.
5. In **Maximum session length (hours)**, enter a value.
6. Select **Save**.

To update the user's maximum session length:

1. Select the user that you want to update by selecting the row.
2. In **Maximum session length (hours)**, enter a value.
3. Select **Save**.

To delete a user's maximum session length and replace it with another user's session:

1. Select the user that you want to delete by selecting the row.
2. In the **User ID** field, select the drop-down menu and select another user.
3. In the **Maximum session length (hours)** column, enter an hour.
4. Select **Save**.

To delete the user's maximum session length:

1. Select the user that you want to delete by selecting the row.
2. Select **Delete**.

To delete the multiple users' maximum session length:

1. Select the rows that you want to delete.
2. Select **Delete**.

Import users from Azure Active Directory

2/18/2021 • 2 minutes to read • Edit Online

Import select users

This procedure can be used by system administrators to import select users from Azure Active Directory (Azure AD).

1. User will be imported with the current session company as their default company. Change current company if applicable before importing users.
2. Go to **System administration** > **Users** > **Users**.
3. Click **Import users**.
4. Select the users that should be imported and select **Import users**.

After import is completed it will be required to assign roles to users.

Import users in bulk

This procedure can be used by system administrators to import a large number of users from Azure Active Directory. Note that it is not possible to select users when using the Batch import option.

Run the import as a batch job

1. User will be imported with the current session company as their default company. Change current company if applicable before importing users.
2. Go to **System administration** > **Users** > **Users**.
3. Click **Batch import**.
4. Expand the **Run in the background** section.
5. Select **Yes** in the **Batch processing** field.
6. In the **Batch group** field, enter or select a value. This is an optional step.
7. Select **Yes** in the **Private** field. This is an optional step.
8. Select **Yes** in the **Critical job** field. This is an optional step.
9. In the **Monitoring category** field, select an option.
10. Click **OK**.

After import is completed, it will be required to assign roles to users.

Run in a sandbox environment

1. Select **Batch import**.
2. Select **OK**.

Set up segregation of duties

2/18/2021 • 2 minutes to read • Edit Online

You can set up rules to separate tasks that must be performed by different users. This concept is named segregation of duties. For example, you might not want the same person to acknowledge the receipt of goods and to process payment to the vendor. Segregation of duties helps you reduce the risk of fraud, and it also helps you detect errors or irregularities. You can also use segregation of duties to enforce internal control policies. Complete the following procedure to create a rule. You must be a system administrator to complete the procedure.

1. Go to **System administration** > **Security** > **Segregation of duties** > **Segregation of duties rules**.
2. Click **New**.
3. In the **Name** field, type a value for the rule.
4. In the **First duty** field, click the drop-down button to open the lookup.
5. In the list, find and select the desired record. Select the first duty that is controlled by the rule.
6. In the **Second duty** field, click the drop-down button to open the lookup.
7. In the list, find and select the desired record. Select the second duty that is controlled by the rule.
8. In the **Severity** field, select an option. Select the severity of the risk that occurs when the same user or role performs both duties.
9. In the **Security risk** field, type a value. Enter a description of the security risk.
10. In the **Security mitigation** field, type a value. Enter a description of the actions that you take to mitigate the security risk. For example, you can mitigate the risk by conducting more detailed reviews of the process, by conducting a monthly managerial review, or by sharing resources with other departments.
11. Click **Save**.

IMPORTANT

Compliance with the rules for segregation of duties is not verified when you create a rule. You can create a rule that creates a conflict for existing roles. Existing user role assignments can also be in conflict with the new rule. You must validate compliance after you create or modify a rule. For more information, see Identify and resolve conflicts in segregation of duties

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Identify and resolve conflicts in segregation of duties

2/18/2021 • 2 minutes to read • Edit Online

This topic explains how to identify and resolve conflicts in segregation of duties. You can set up rules to separate duties that must be performed by different users. This concept is named segregation of duties. When the definition of a security role or the role assignments of a user violate the rules, the conflict is logged. All conflicts must be resolved by the administrator. Complete the following procedure to identify and resolve conflicts.

After a rule has been added, verify that all existing roles are compliant.

Verify that existing roles and duties comply with new rules for segregation of duties

1. Go to **System administration** > **Security** > **Segregation of duties** > **Segregation of duties rules**.
2. Select **Validate duties and roles**. If any roles violate the rules, a message is displayed that contains the name of the rule, the role, and the names of the conflicting duties. Conflicting roles must be modified using **Security configuration** and can't include conflicting duties. If no roles violate the selected rule, a message indicates that all roles comply.

> **NOTE**
> The validation is only performed for the selected rule. It is important to validate compliance for each rule.

When you create or modify a role, the rules for segregation of duties are automatically enforced. You cannot assign conflicting duties to a role.

Next, verify that all existing role assignments are compliant.

Verify that user role assignments comply with new rules for segregation of duties

1. Go to **System administration** > **Security** > **Segregation of duties** > **Verify compliance of user-role assignments**.
2. Select **OK**. A notification displays the results of the validation. Conflicts are logged on the **Segregation of duties unresolved conflicts** page.

When you assign users to roles, the rules for segregation of duties are automatically enforced. If you try to assign a user to roles that contain conflicting duties, you receive an error message. You must then resolve the conflict by denying or allowing the additional role assignment. The additional role will be assigned after the assignment is allowed.

> **NOTE**
> Conflicts are currently not verified for users that are assigned roles based on the Active Directory Domain groups.

View and resolve conflicting user role assignments

1. Go to **System administration** > **Security** > **Segregation of duties** > **Segregation of duties unresolved conflicts**.
2. Select a conflict, and then select one of the following actions:

- **Deny assignment**: This will deny the assignment of the user to the additional security role. If you deny an automatic role assignment, the user is marked as excluded from the role. The excluded user isn't granted the access associated with the role and can't be assigned to the role until the administrator removes the exclusion.
- **Allow assignment**: This will override the conflict and allow the user to be assigned to the additional security role. If you override a conflict, you must enter a reason in the **Reason for override** field. All overridden role assignments can be viewed on the **Segregation of duties conflicts** page.

> **NOTE**
>
> If several conflicts are listed for the same user, select the user record and evaluate assigned roles on the **Users** page. To avoid this conflict, validate each rule after it's added or modified.

Assign users to security roles

2/18/2021 • 2 minutes to read • Edit Online

To use anything other than common capabilities in Finance and Operations apps, users must be assigned to security roles. You can assign users to roles automatically, based on rules and business data, exclude users from automatic role assignment, or add users to roles manually.

Automatically assign users to roles

This procedure explains how system administrators can automatically assign users to roles, based on business data.

1. Go to **Navigation pane > Modules > System administration > Security > Assign users to roles**.
2. In the tree, select 'Accounting supervisor'. Select the role that you want to configure the rule for. In this example, select Accounting supervisor.
3. Select **Add rule** to open the dialog menu.
4. In the **Select a query** list, find and select the desired record. Select the query to use for this rule.
5. In the **Membership rule name** list, click the link in the selected row.
6. Select **Edit query**. Edit the query, as needed.
7. Select **OK**.
8. Select **Run automatic role assignment**.
9. Go to **Navigation pane > Modules > System administration > Users > Users** (ideally in a separate browser tab).
10. Review the roles assigned to various users to confirm that the role assignment query was correct. Adjust and re-run if needed.

Exclude users from automatic role assignment

1. Close the page.
2. Go to **Navigation pane > Modules > System administration > Security > Assign users to roles**.
3. In the tree, select 'Accounting supervisor'. Select a role. For this example, select Accounting supervisor.
4. In the **Users assigned to role** menu, select **Manually assign / exclude users**.
5. In the **Assign users to or exclude users from role** list, mark the selected row. Select a user.
6. On the **Action pane**, select **Exclude from role**.
7. Select **Exclude from role** to exclude the selected users from the role. To remove exclusions, select the users that you want to remove exclusions for, and then click **Reset status**. When you remove an exclusion by resetting the user's status, the user's role is assigned automatically. However, the user is not immediately assigned to the role or excluded from the role when you reset the status. Instead, the user is either assigned to the role or removed from the role the next time that the rules for automatic role assignment are run.

Manually assign users to roles

Users who are manually assigned to security roles must also be manually removed by the administrator. These users are not removed from roles by rules for automatic role assignment.

1. Go to **Navigation pane > Modules > System administration > Security > Assign users to roles**.
2. In the tree, select a role, and in the **Users assigned to role** menu, select **Manually assign / exclude users**.

3. In the **Assign users to or exclude users from role**, users that have not been assigned the role are listed with the **Assignment mode** set to **None**. Select one or more users that should be assigned the role.
4. On the **Action pane**, select **Assign to role**. The **Assignment mode** is updated to **Manual** and the users now have a new role assigned.

Import or export a customized security configuration by using Data management

2/18/2021 • 2 minutes to read • Edit Online

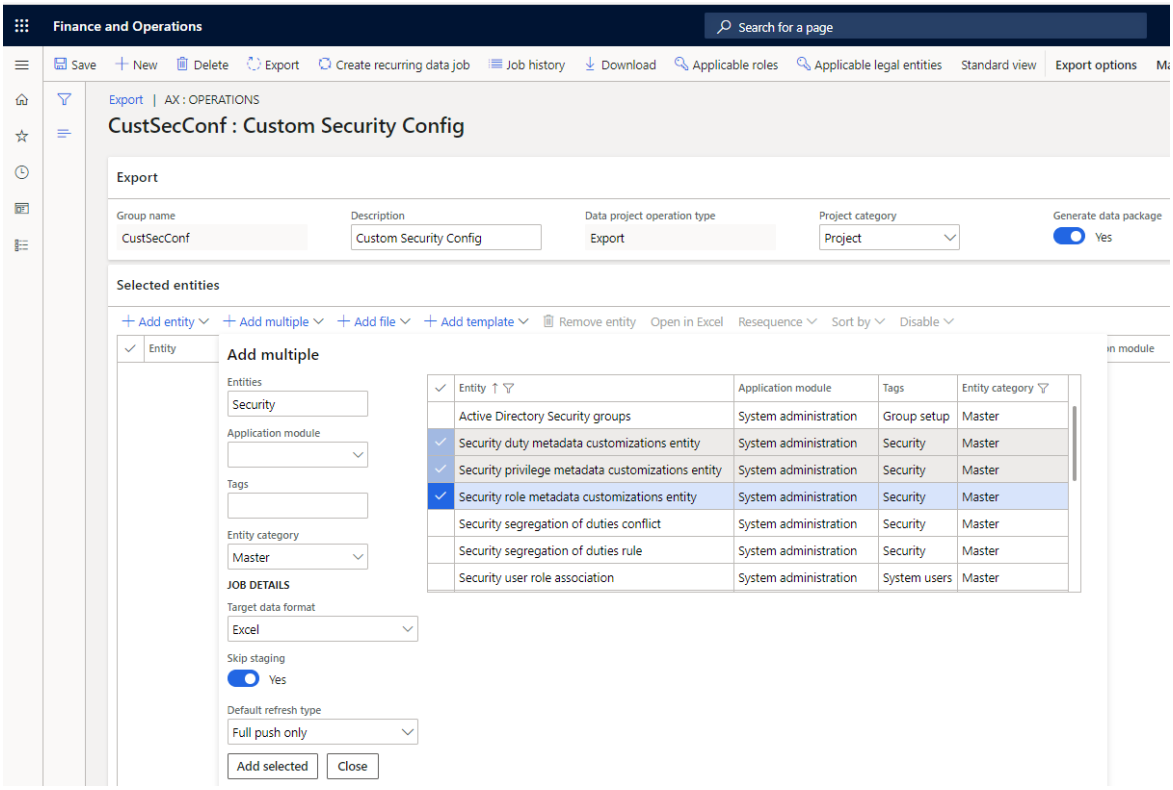
The topic explains how a customized security configuration can be exported and imported across environments by using the Data management framework. This functionality can be used when, for example, a customized security configuration must be moved from a test environment to a production environment.

The following entities hold the customized, role-based security (that is, privileges, duties, and roles) that has been added or modified by using security configuration:

- Security privilege metadata customization entity
- Security duty metadata customization entity
- Security role metadata customization entity

Export customized security configuration

1. Go to **System administration** > **Workspaces** > **Data management**.
2. Select the **Export** tile.
3. In the **Group name** field, enter a name for the group.
4. Set the **Generate data package** option to **Yes**.



5. Select **Add multiple** to open the drop-down dialog box.
6. Filter the entities by setting the following fields:
   - In the **Entities** field, enter **Security**.
   - In the **Entity category** field, select **Master**.

7. In the **Target data format** field, select **Excel**.
8. Select the applicable security customization entities.
9. Select **Add selected**.

   In version 10.0.12 and later, ignore any warning messages about data length. Those messages aren't applicable, because the entities that are included use containers in data package mode.
10. Select **Close**.
11. Make sure that the **Sequence** field is set in the order of the entity dependencies. Privileges should be first, then duties, and finally roles.
12. Select **Export**.
13. Select **Close**.
14. Wait for the job to be completed. Select **Refresh** to view the status.
15. Select **Download package**.
16. Save the package.

Import customized security configuration

1. Go to **System administration** > **Workspaces** > **Data management**.
2. Select the **Import** tile.
3. In the **Group name** field, enter a name for the group.
4. Select **Add file**.
5. Select **Upload and add**.
6. Find the exported package, and then select **Open**.

   In version 10.0.12 and later, ignore any warning messages about data length. Those messages aren't applicable, because the entities that are included use containers in data package mode.
7. Select **Close**.
8. Select **Import**.
9. Select **Close**.
10. Wait for the job to be completed. Select **Refresh** to view the status.

Related security configuration entities

- **SystemSecurityUserRoleOrganizationEntity** – Assignment of organizations to security roles.
- **Security segregation of duties rule** – Segregation of duties rules.
- **Security segregation of duties conflict** – Segregation of duties conflicts. This entity has unresolved conflicts but also reviewed conflicts.

Additional resources

- Data import and export jobs overview
- Move all user and security settings with data entities (blog post), by André Arnaud de Calavon

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Security diagnostics for task recordings

2/18/2021 • 2 minutes to read • Edit Online

Before you begin

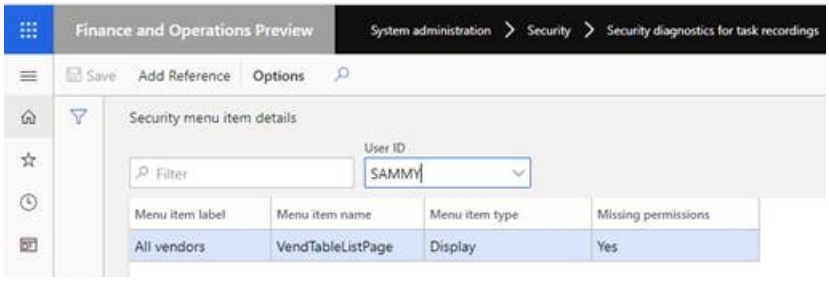
This topic provides information about how to analyze and manage security permission requirements based on a task recording. Before you complete the steps in this topic, you must have a task recording of the business process that you want to analyze. To record a business process, see Task recorder resources.

Manage security for a task recording

1. Go to **System administration** > **Security** > **Security diagnostic for task recording**.
2. Open the task recording from its location. Select **Open from this PC** or **Open from Lifecycle Services**, and then select **Close**.
3. This will open the **Security menu item details** page that lists the security objects required for the process.

> **NOTE**
>
> The **Action** and **Output** menu items are not included in the list.

4. In the **User ID** field, select a user. If the user does not have permissions for some menu items, the **Missing permissions** field will update to **Yes**.



5. Select **Add Reference** to see a list of the security objects, including roles, duties, and privileges that grant the missing permission.
6. Select a security object from the list:

   - If **Role** is selected, select **Add role to user**. This will open the **Assign users to roles** page. For more information, see Assign users to security roles page.
   - If **Duty** is selected, select **Add duty to role**, select the roles that the duty should be added to, and then select **OK**.
   - If **Privilege** is selected, select **Add privilege to duties**, select the roles that the duty should be added to, and then select **OK**.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

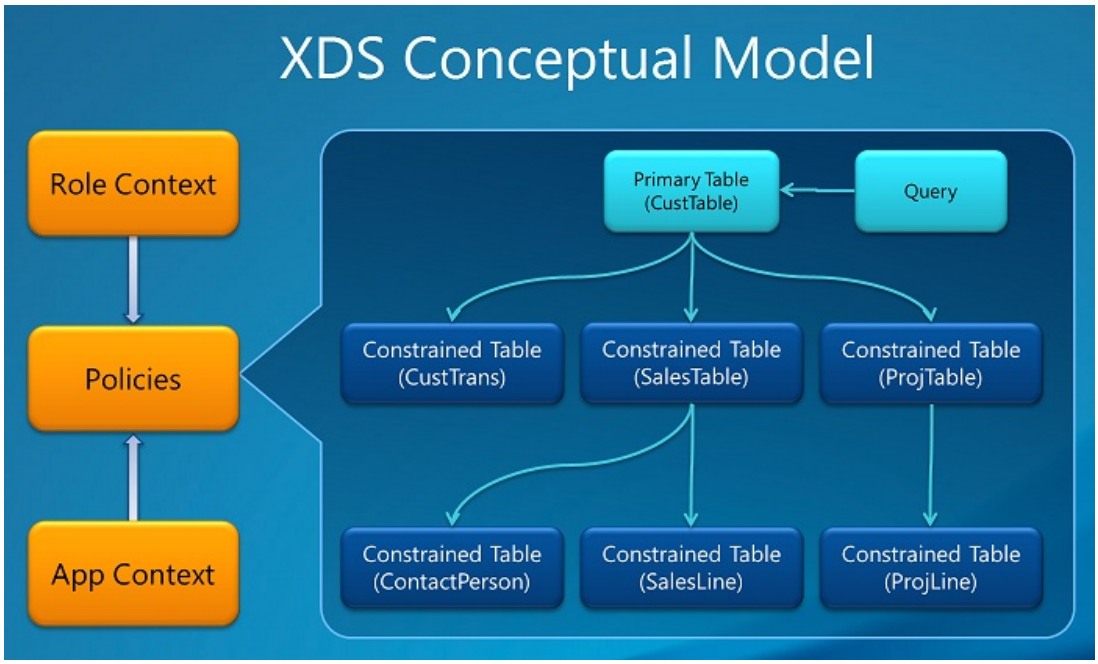
Extensible data security policies

2/18/2021 • 2 minutes to read • Edit Online

This topic provides an overview of Extensible Data Security (XDS) policies in Finance and Operations apps. XDS allows developers to supplement role-based security by restricting access to table records based on security policies. The query in the policy applies a filter and only records that satisfy the conditions of the filter will be accessible from the restricted tables.

Data security policy components

- **Constrained tables**: The table or tables from which data is filtered or secured. For example, in a policy that secures access to customers based on customer group, the **CustTable** would be the constrained table.
- **Primary table**: Used to secure the content of the related constrained table. In the above example, the **CustGroup**table would be the primary table. The primary table must have an explicit relationship to the constrained table.
- **Policy query**: Used to secure the constrained tables content using a range condition on the primary table contents. Only records that are included in the range will be accessible. The range can, for example, be based on a specific value for Customer Group.
- **Context** – Controls the conditions under which a policy is applicable. Two main types of contexts are available:
  - **Role context**: Based on the roles that the user is assigned. There are two sub-options for role context:
    - **RoleName** – Indicates that the security policy is only applied to the application user assigned to the role equal to the value of RoleName.
    - **RoleProperty** – This value is used in combination with the **ContextString** property to specify multiple user roles context. It is applied when the Context String value defined in the **Role Property** field for the policy is the same as the **ContextString** field value for the assigned user roles.
  - **Application context**: Applied if the context string set by the application using the XDS::SetContext API is the same as the value defined in the **Context String** field for the policy.



In the Application Object Tree (AOT), policies and their components are displayed under **Security > Policies**.

Important considerations

The policy query is added to the WHERE clause, or ON clause, on SELECT, UPDATE, DELETE and INSERT operations involving the specified constrained tables. Unless carefully designed and tested, policy queries can have a significant performance impact. Therefore, make sure to follow simple but important guidelines when developing an extensible data security policy. For more information, see the "Developing efficient extensible data security policies" section in Developing Extensible Data Security Policies (White paper) [AX 2012].

When two or more security policies apply, the intersection (not the union) of the records that are included by each policy are the only records that can be accessed. This means that a record must satisfy all the applicable security policies before access to the record is allowed.

Additional resources

For information about how to debug policies, create more advanced policies, including chaining of restricted tables, table relations based on expressions and much more please refer to these resources:

- Create a simple security policy
- Developing Extensible Data Security Policies (white paper) [AX 2012]
- Securing Data by Dimension Value by using Extensible Data Security (white paper) [AX 2012]
- Extensible Data Security examples – by Andre Arnaud De Calavon [blog]
- Extensible Data Security (XDS) Framework in D365FO - by Alex Meyer [blog]

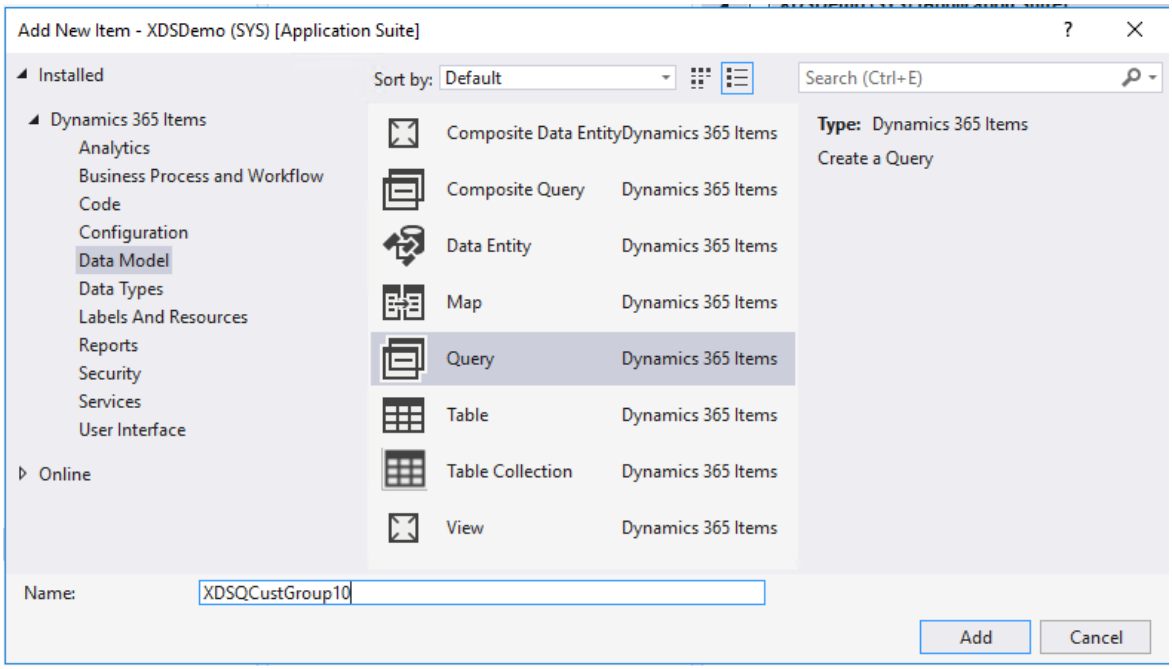
Create a security policy

2/18/2021 • 2 minutes to read • Edit Online

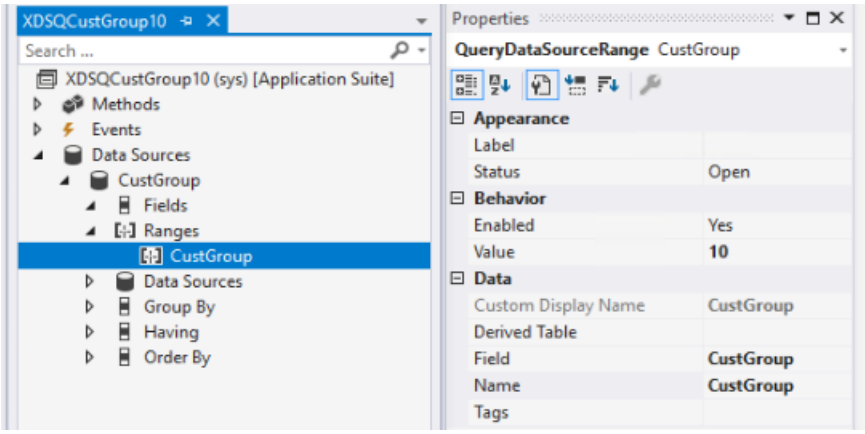
This topic explains how to create a simple security policy that secures access to customers and customer groups, based on a range for a customer group.

Add a new query

1. In Visual Studio, add a new query, such as XDSQCustGroup10, to your project/solution. The query will be used to restrict data access from the **Constraint** table.

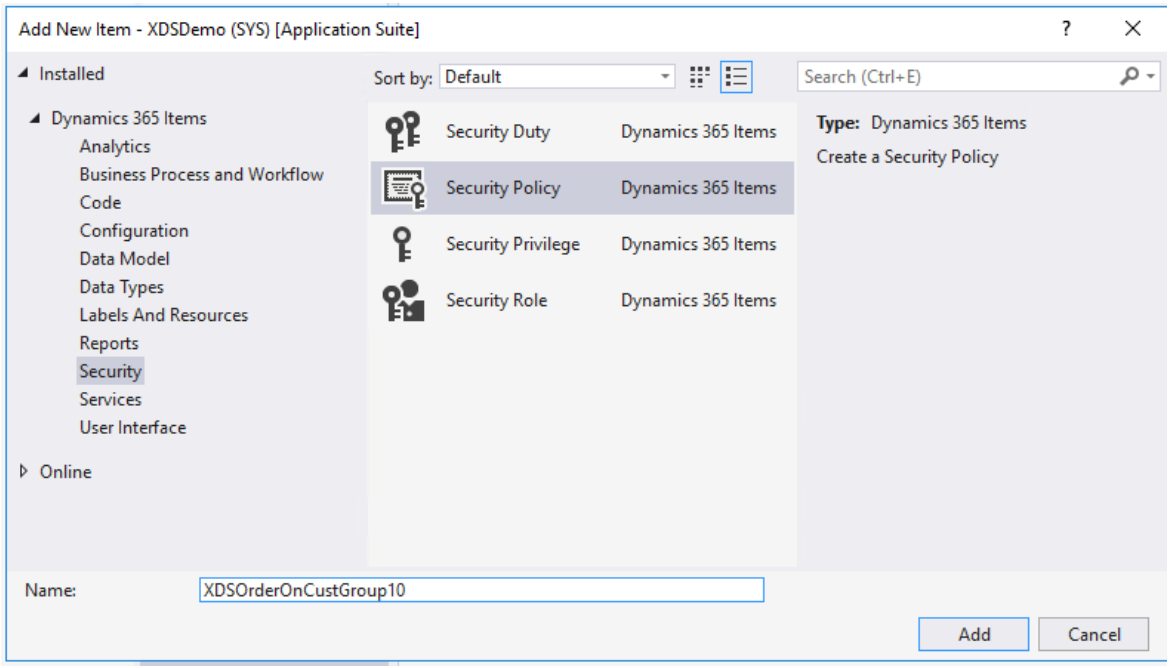


2. Right-click **Data Sources**, and the select **New Data Source**.
3. In the **Table** field, enter the primary table name **CustGroup**.
4. Right-click **Ranges**, and then select **New Range**.
5. Set the **Enabled** field to **Yes**.
6. In the **Data Source** field, enter the primary table name, in this case, 'CustGroup'.
7. In the **Value field**, enter **10** to restrict access to data where CustGroup has value of 10, by defining the Range for the CustGroup field.

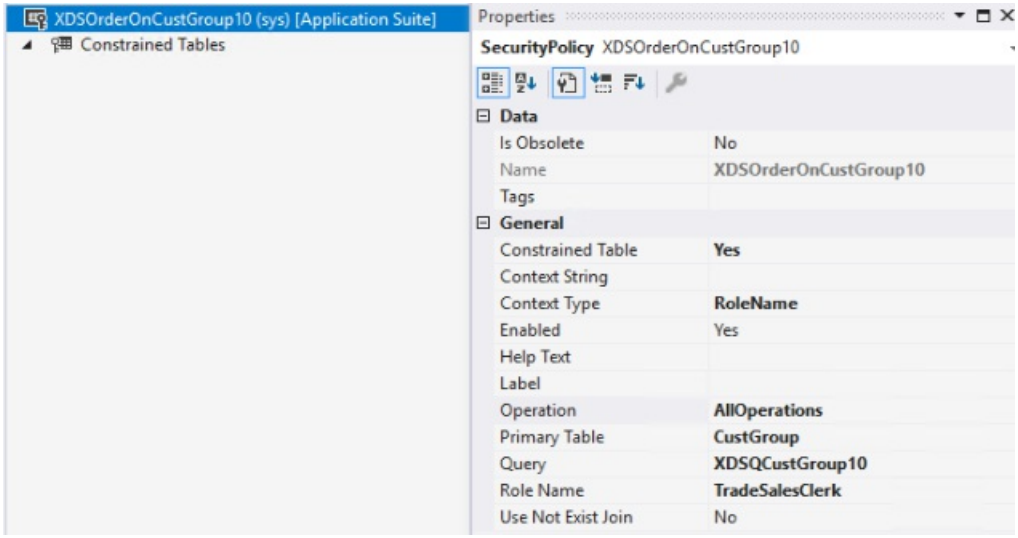


Add a new security policy

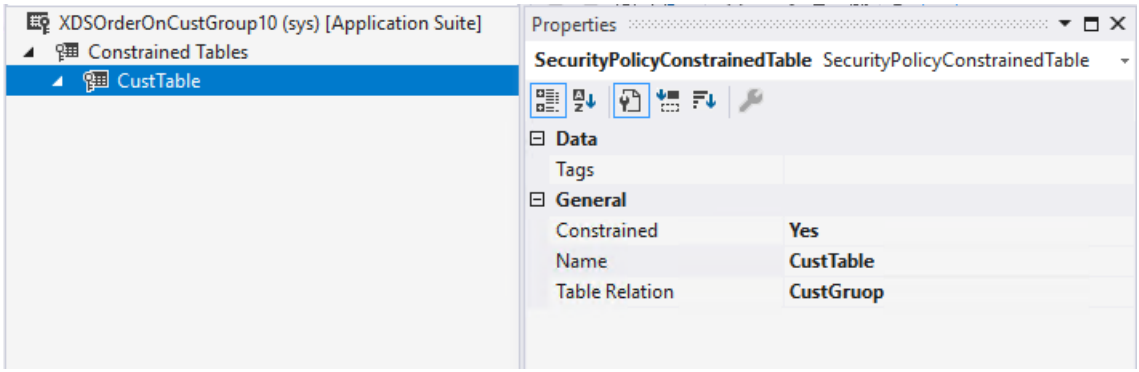
1. Add a new security policy, such as XDSCustTableOnCustGroup10.



2. Set **Constrained Table** to **Yes**. This will also secure access to the primary table. In this example this is the **CustGroup** table.
3. Set the **Context Type** field to **RoleName**.
4. Set the **Enabled** field to **Yes**.
5. Set the **Operation** field to **AllOperations**. Other available values for **Operation** include **Select**, **Insert**, **Update**, **Delete**, and **InsertUpdateDelete**.
6. Set **Primary Table** field to **CustGroup**.
7. Set the **Query** field to the name of the query created above, for example 'XDSQCustGroup10'.
8. Set the **Role Name** field to 'TradeSalesClerk'. Because **Context Type** is set to RoleName for this policy, it is required to enter the AOT name for a user role.



9. Next, add constrained tables. In this simple example add one table.



   a. Right-click **Constrained tables**, and then select **New > Constrained Table**.

   b. Set **Constrained** to **Yes**.

   c. In the **Name** field, enter the Constrained table, for example 'CustTable'.

   d. In the **Table Relation** field, enter the relationship to the primary table, in this case 'CustGroup'.

10. As a final step, it is required that you build and synchronize the solution to activate the policy.

Stay compliant with user licensing requirements

2/18/2021 • 5 minutes to read • Edit Online

This topic provides an overview of how customers can stay compliant with the user licensing requirements for Finance and Operations apps, such as Microsoft Dynamics 365 Finance, Dynamics 365 Supply Chain Management, and Dynamics 365 Commerce.

The licensing requirements for users are determined by the security roles that are assigned to those users. Security roles are built based on a hierarchy of:

- Sub-roles
- Duties
- Privileges
- Directly referenced securable objects

For more information, see Role-based security.

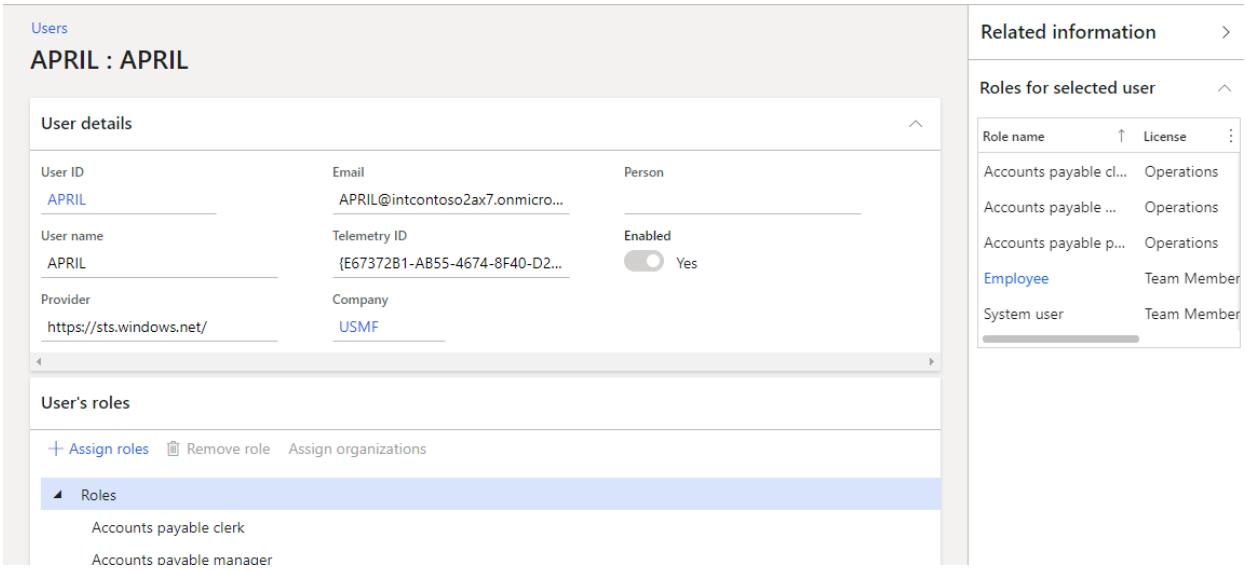
The licensing requirements for users are determined at the organization or tenant level. This topic is focused on the requirements for a single environment. If you have multiple environments, the requirements must be analyzed across all of them.

A licensing requirement is assigned to every securable object. Examples of licensing requirements include **None**, **Team members**, **Activity**, and **Operations**. The **Operations** licensing requirement indicates that a full user license is required. Some privileges are unique to a specific full user license and require a *base* or *attach* license for the full user license be assigned to the user. For more information, see the User license estimator report section later in this topic.

The rest of this topic describes the various tools that you can use to ensure that the actual licensing complies with the expected licensing requirements.

Roles for selected user FactBox on the Users page

You assign roles to users on the **Users** page (**System administration > Users**). You can view license requirements for each role in the **Roles for selected user** FactBox.



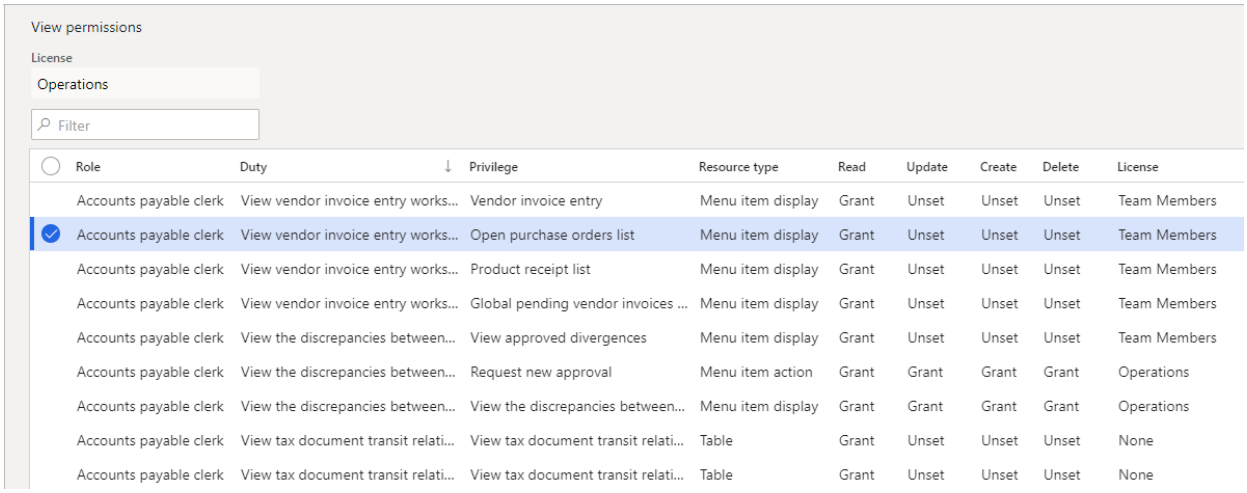
The maximum license requirement determines the actual licensing requirement for a user. If any license

requirement is identified as **Operations**, you must use the User license estimator report to determine the full user licensing requirements.

> **NOTE**
>
> Starting with platform update for version 10.0.15, the required license types are extended to include application specific license types such as Commerce, Finance, and Supply Chain Management. This extension makes it possible to view the actual license requirements when you assign roles to users and when using the Roles for the selected user FactBox. More than one value can be displayed for the required license type.

View permissions page

During security configuration on the **Configure security** page (**System administration > Security > Configure security**), you can select any security object, a role, duty, or permissions, and then select **View permissions** to view all permissions that are currently included and their licensing requirements. The header of the **View permissions** page shows the required license level.

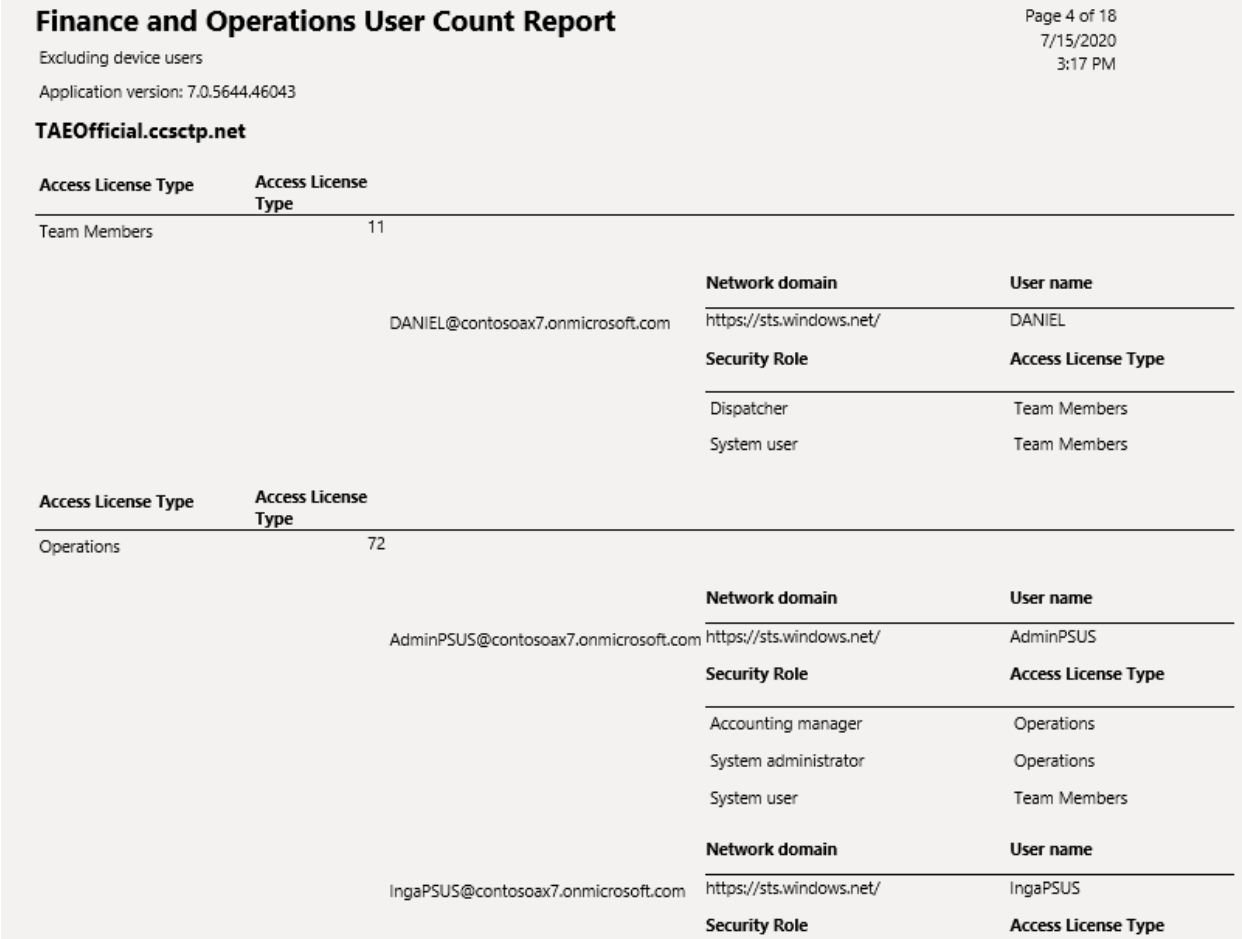


Role	Duty	Privilege	Resource type	Read	Update	Create	Delete	License
Accounts payable clerk	View vendor invoice entry works...	Vendor invoice entry	Menu item display	Grant	Unset	Unset	Unset	Team Members
Accounts payable clerk	View vendor invoice entry works...	Open purchase orders list	Menu item display	Grant	Unset	Unset	Unset	Team Members
Accounts payable clerk	View vendor invoice entry works...	Product receipt list	Menu item display	Grant	Unset	Unset	Unset	Team Members
Accounts payable clerk	View vendor invoice entry works...	Global pending vendor invoices ...	Menu item display	Grant	Unset	Unset	Unset	Team Members
Accounts payable clerk	View the discrepancies between...	View approved divergences	Menu item display	Grant	Unset	Unset	Unset	Team Members
Accounts payable clerk	View the discrepancies between...	Request new approval	Menu item action	Grant	Grant	Grant	Grant	Operations
Accounts payable clerk	View the discrepancies between...	View the discrepancies between...	Menu item display	Grant	Grant	Grant	Grant	Operations
Accounts payable clerk	View tax document transit relati...	View tax document transit relati...	Table	Grant	Unset	Unset	Unset	None
Accounts payable clerk	View tax document transit relati...	View tax document transit relati...	Table	Grant	Unset	Unset	Unset	None

> **NOTE**
>
> Starting with platform update for version 10.0.15, the required license types are extended to include application specific license types such as Commerce, Finance, and Supply Chain Management. This extension makes it possible to identify the specific security objects that determine the actual license requirements while also configuring security. More than one value can be displayed for the required license type.

User license counts report

The **User license counts** report (**System administration > Inquiries > License**) is used to get a count of required licenses per license type (for example, **Team members**, **Activity**, and **Operations**).



Finance and Operations User Count Report

Page 4 of 18
7/15/2020
3:17 PM

Excluding device users

Application version: 7.0.5644.46043

TAEOfficial.ccsctp.net

Access License Type	Access License Type
Team Members	11

DANIEL@contosoax7.onmicrosoft.com

Network domain	User name
https://sts.windows.net/	DANIEL

Security Role	Access License Type
Dispatcher	Team Members
System user	Team Members

Access License Type	Access License Type
Operations	72

AdminPSUS@contosoax7.onmicrosoft.com

Network domain	User name
https://sts.windows.net/	AdminPSUS

Security Role	Access License Type
Accounting manager	Operations
System administrator	Operations
System user	Team Members

IngaPSUS@contosoax7.onmicrosoft.com

Network domain	User name
https://sts.windows.net/	IngaPSUS

Security Role	Access License Type

The report also provides details about each user and the licensing requirements for each assigned role. Users are listed under the highest license type. If the license requirement is identified as **Operations**, you must use the User license estimator report to determine the specific full user licensing requirements.

The **User counts history** report shows total counts per date, but without any details.

> **NOTE**
>
> This report depends on the **Named user license count reports processing** batch job. To determine when the batch was last run, use the **Batch job history** page.

User license estimator report

If the previously mentioned tools have identified that any users require a license of the **Operations** type, you can use the **User license estimator** report (**System administration > Inquiries > License reports**) to get a count of specific licensing requirements for those users. The report includes only those users.

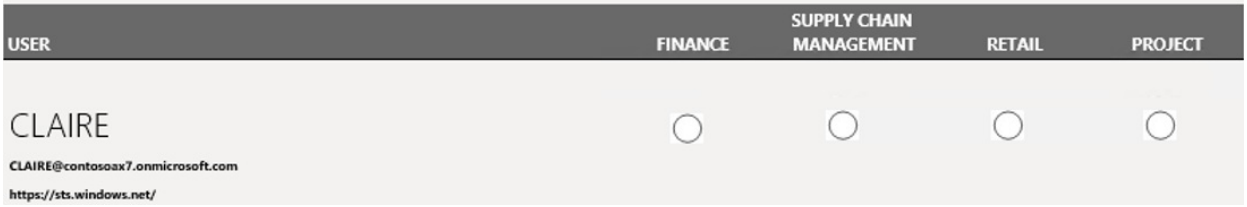
If no privileges that require a specific full user license have been assigned to a user, that user is shown, but no specific full user licenses are marked. That user will be compliant with any full user license that is assigned. In the example in the following illustration, the **Database management administrator** role is assigned to a user, and no other roles that require a license of the **Operations** type are assigned.

Role	Security Role Description	Team Members	Activity	Finance, SCM, Commerce
Administrator				
Data management administrator	Super user for the data management activities in the system. In addition to the capabilities of the DataManagementMigrationUser and DataManagementOperationsUser, this role provides access to the DataManagementITWorkspace - an operational workspace to monitor all data management activities			•

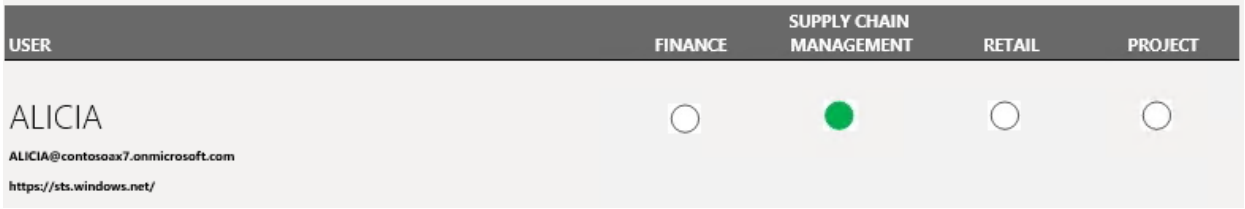
Based on the licensing guide, this user is compliant with any full user license.

Admin rights apply across the Finance, Supply Chain Management, and Commerce apps. For example, if you have a Finance license, you have the admin rights for Finance, Supply Chain Management, and Commerce.

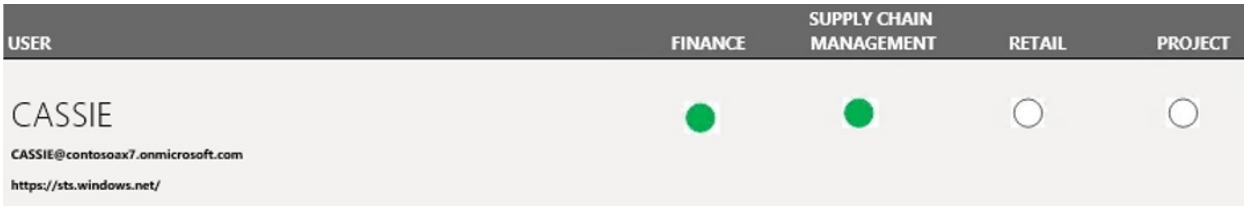
This principle might apply to several roles that are provided by Microsoft and customizations, based on the included privileges. No specific license is indicated for these users on the **User license estimator** report.



Alternatively, if privileges that require a specific full user license have been assigned to a user, that license is indicated when you look at the licenses that are assigned to the user. In the example in the following illustration, the user must have a *base* license for Supply Chain Management.



If privileges that require more than one specific full user license have been assigned to a user, those licenses are indicated when you look at the licenses that are assigned to the user. In this case, the user must have a *base* license for one of the licenses and an *attach* license for all other full user licenses that are required. In the example in the following illustration, the user must have either a *base* license for Finance and an *attach* license for Supply Chain Management, or a *base* license for Supply Chain Management and an *attach* license for Finance.



The totals per specific full user license counts aren't divided into *base* licenses and *attach* licenses.

> **NOTE**
>
> The total count per specific full user license and evaluation of custom privileges are included starting in the release of the platform update for version 10.0.13. The report can't separate the *base* and *attach* license requirements. Therefore, it lists only the total full user licenses requirements.

Additional resources

For information about how to buy and license Finance and Operations apps, see Microsoft Dynamics 365 Licensing Guide.

For information about how to assign licenses to users in the Microsoft 365 admin center, see Assign licenses to users.

Additional user licenses are required when multiple implementation projects exist for the same tenant. For more information, see Multiple LCS projects and production environments on one Azure AD tenant.

View independent software vendor license status

2/18/2021 • 2 minutes to read • Edit Online

This topic explains how you can view the status of independent software vendor (ISV) licenses for Finance and Operations apps, such as Dynamics 365 Finance, Dynamics 365 Supply Chain Management, and Dynamics 365 Commerce.

License codes and configuration keys are part of the ISV licensing model for Finance and Operations apps.

> **NOTE**
> Configuration keys that are provided by Microsoft aren't part of the licensing model for Finance and Operations apps. The keys are only used to enable and disable functionality.

When an ISV license key and code are installed, the corresponding configuration key will be available and enabled on the **License configuration** page.

View ISV license status

Each ISV solution that is tied to a license runs only when a valid license code exists in the customer's environment. Therefore, if an ISV ties their solution to a license, but the customer doesn't have a valid license code, the solution doesn't run. To prevent the loss of functionality, it's important to track the expiration dates, if applicable, for license codes. To view the expiration dates, go to **System administration** > **Setup** > **License configuration**, and select the **License codes** tab.

To avoid downtime, review the expiration dates of the license keys, and if applicable, obtain and import new license keys before moving to a new version. The **License codes** tab shows the expired license codes. The corresponding configuration key won't appear in the **Configuration keys** tab.

Additional resources

For more information about the ISV licensing feature, see Independent software vendor (ISV) licensing.

For more information about how to import ISV licenses into an on-premises deployment, see Independent software vendor (ISV) licensing (on-premises).

For more information about the configuration keys report, see License codes and configuration keys report.

> **NOTE**
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Process automation

2/18/2021 • 2 minutes to read • Edit Online

Process automation allows simple scheduling of processes that will be run by the batch server. The updated calendar view of the scheduled work allows end users to view and take action on scheduled and completed work.

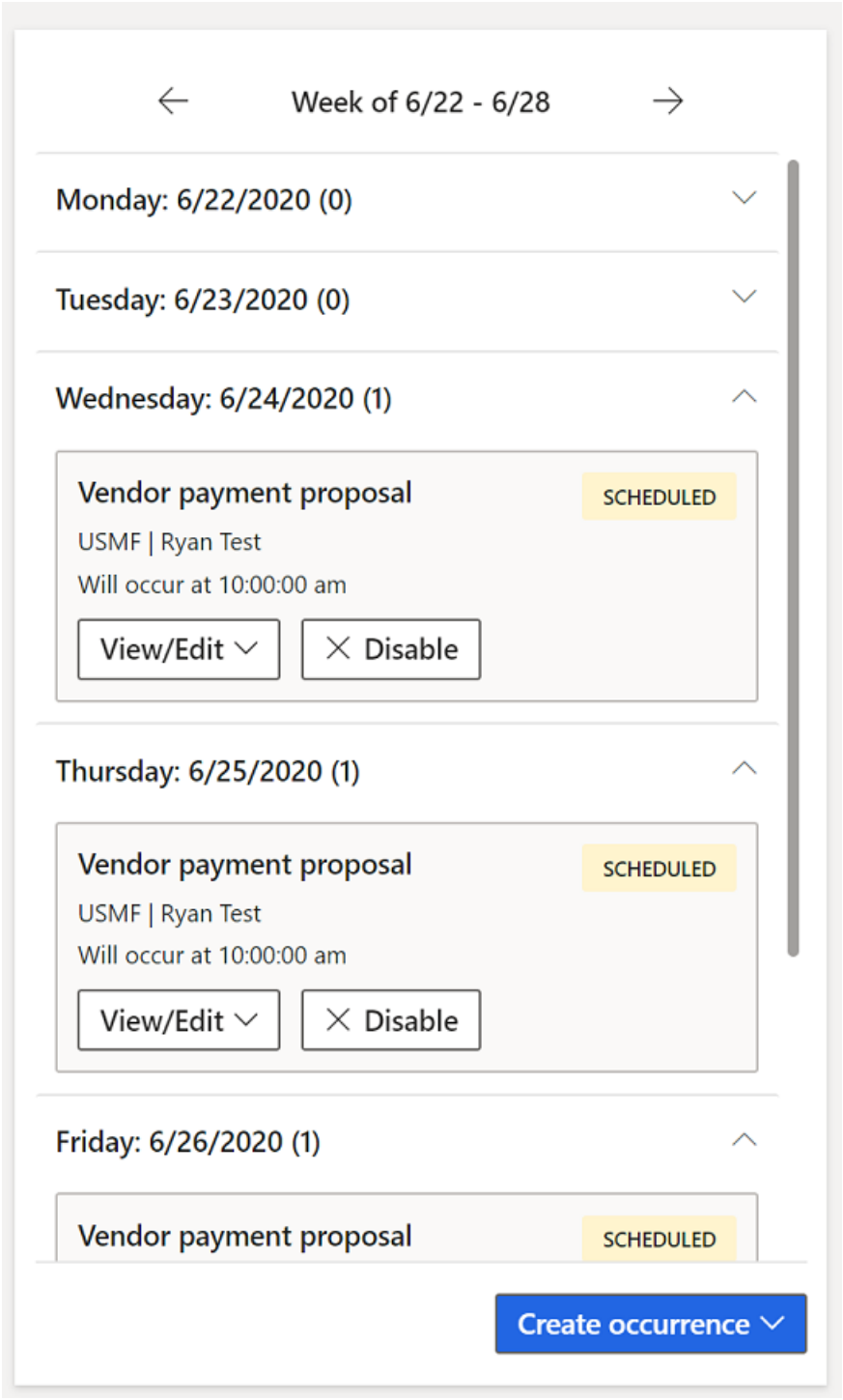
Administration

The central administration page for all process automations is found in the System Administration module under the **Setup** menu. This page will list all automated processes (series) that are set up in the system. It will also allow you to add new process automations directly from this page. After a series is set up, you can manage each series from this list. You can choose to edit the entire series, delete it, view all occurrences in a list view, or disable the series if you would like to pause the scheduled work for a while.

Any processes that are disabled in feature management won't show when the feature is disabled. Additionally, the process automation scheduling engine won't schedule any occurrences or background processes for a disabled feature. Re-enabling the feature will cause any scheduled occurrences or background processes in the past to run immediately.

Calendar view

One of the key benefits of process automation is the ability to see the scheduled work in a simple calendar view. This view allows you to see work for a week at a time. You'll see this view on the right side of the **Process automation** page. It will be populated with the scheduled work for the selected series.



Occurrence changes

Each occurrence can be modified without impacting other occurrences defined by the series that originated them. Occurrences of scheduled work can be edited from the calendar view by selecting the **View/Edit** button and selecting **Occurrence**. This page allows you access to all the settings originally shown in the series setup wizard and provides the ability to make a one-off change for the selected occurrence. An occurrence of scheduled work can also be turned off by selecting the **Disable** button from the calendar view.

Developer documentation

The process automation framework allows developers to extend the process automation framework. The Process automation framework documentation provides information about how you can create custom processes that you require to be run by the batch server scheduled with the process automation wizard and appear in the calendar view automatically.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Batch processing overview

2/18/2021 • 2 minutes to read • Edit Online

This topic provides an overview of batch processing.

Many tasks in Finance and Operations can be run as part of batch jobs. For example, batch jobs can include tasks for printing reports, performing maintenance, or sending electronic documents. By using batch jobs, you can avoid slowing down your computer or the server during typical working hours.

The tasks in a batch job can run either sequentially or at the same time. Additionally, you can create dependencies between tasks. In other words, the sequence of tasks can differ, depending on whether an earlier task succeeds or fails.

You can set up recurrence patterns for batch jobs. For example, you can set up a job to process invoices automatically at the end of every month.

To monitor batch jobs, you can set up alerts. Alerts can be sent when the batch job succeeds, fails, or has finished running.

After a batch job has been processed, you can view the history. The history includes any messages that were encountered while the job was running.

Use batch groups to categorize batch tasks and run them on specific servers. The servers in your environment might have different software installed, or they might be available at different times of the day. Batch groups are used to direct batch tasks to the most appropriate server. Tasks in the same batch job can belong to different batch groups.

For example, server A is set up to print reports, and server B is set up to send electronic documents. You can use batch groups to make sure that reporting tasks are run on server A and electronic documents are processed by server B.

For more information, see Batch processing and batch servers.

Batch functions

Administrators and Batch managers can perform common tasks including creating and copying batch jobs, changing a batch job user, and specifying a time period in which a job should not execute. For more information about these tasks, see the following topics:

- Create a batch job
- Batch manager security role
- Active batch periods
- Copy a batch job
- Set up alerts
- Enhanced batch forms
- Clean up the batch job history
- Abort an executing batch job

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Batch processing and batch servers

2/18/2021 • 5 minutes to read • Edit Online

This topic describes batch processing and batch servers, and how to plan for their use.

The batch framework provides an asynchronous, server-based batch processing environment that can process tasks across multiple instances of Application Object Server (AOS).

You should become familiar with the following aspects of the batch framework:

- A **batch job** is a process that is used to achieve a specific goal. A batch job consists of one or more batch tasks.
- A **batch task** is an activity that is run by a batch job. You can add batch tasks that have multiple types of dependencies to a batch job. You can also configure AOS instances to run multiple threads, each of which runs a task. All batch tasks that are waiting to be run can be run by any available AOS instance that is configured as a batch server. To improve throughput and reduce overall execution time, you can define a batch job as many tasks and then use a batch server to run the tasks against all available AOS instances.
- A **batch group** is an attribute of a batch task. A batch group lets the administrator determine or specify which AOS instance runs the task. When you create a new task, it's put in the default batch group. All batch servers are configured to process the default batch group and the waiting tasks from any job. Additionally, you can create a named batch group, and then set an affinity between that batch group and specific AOS instances. After you create this affinity, only the specified AOS instances will process tasks from the named batch group, and those AOS instances will process tasks from the named batch group only. You can also add the default batch group to the configured servers, if that batch group is required.

Batch server topology planning

The capacity of a batch server is based on the maximum number of threads that can run concurrently on the AOS instance. Each thread runs one batch task. You can add complex dependencies between or among tasks. You can run these tasks in serial steps or parallel steps, depending on the business logic and requirements. All tasks that don't have any dependencies are considered parallel tasks. AOS instances that are configured as batch servers periodically check for tasks that are waiting to be processed. The batch server assigns each parallel task to a thread and starts to process the thread.

You can run multiple threads across multiple AOS instances. Each AOS instance automatically runs multiple threads, depending on that capacity that is defined in the configuration settings. Therefore, parallel tasks from a job can be run on multiple threads across multiple AOS instances.

A batch server checks for available threads one time per minute. Therefore, you might have to wait for a minute before you see that a waiting task is picked up for processing by an available thread.

Batch server management planning

All batch servers can be managed from a single location.

One typical use of batch servers is to load balance jobs across multiple servers. You can set the number of threads that the batch server will process.

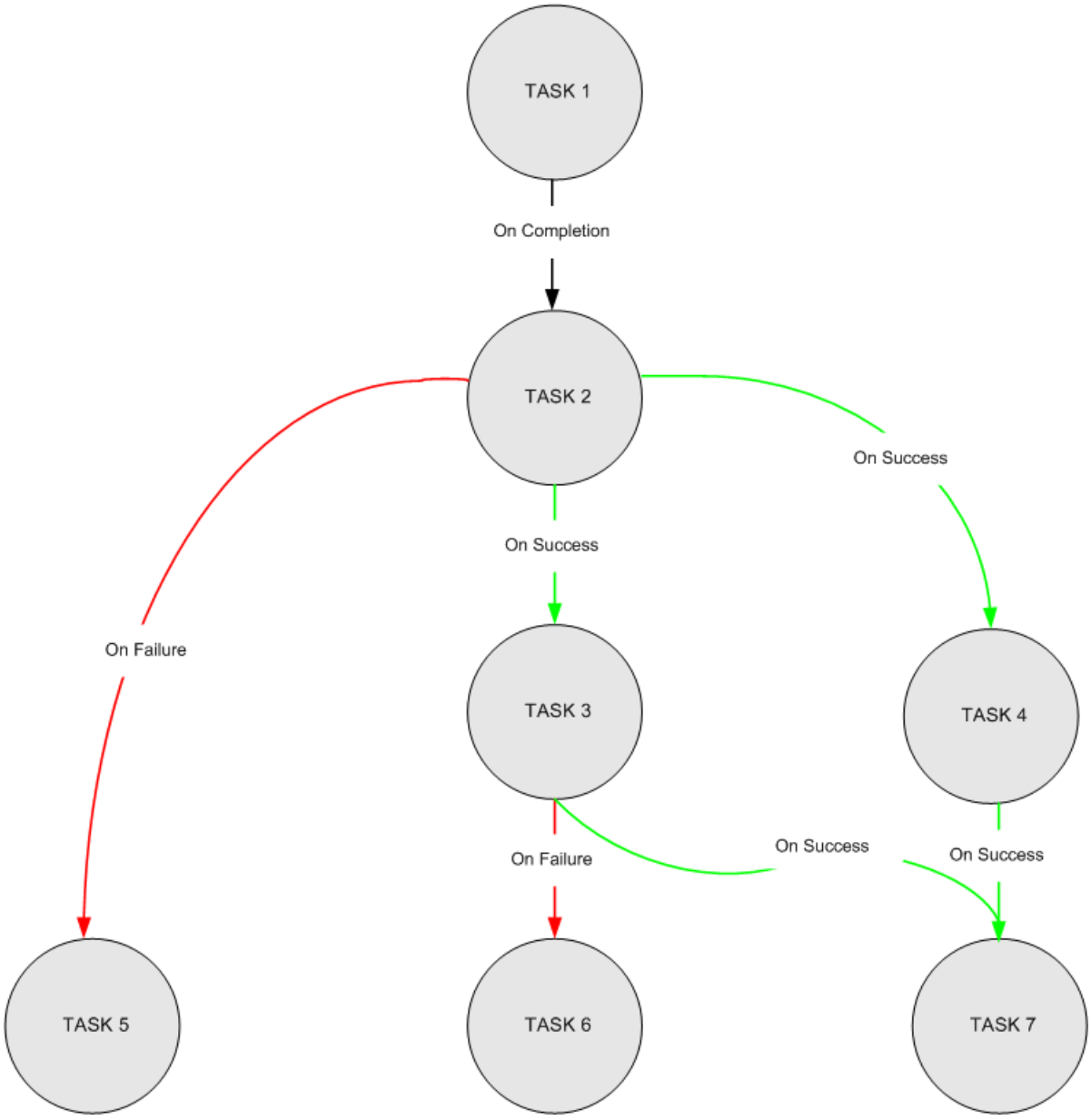
Because batch servers are also active AOS instances that service requests from the client and other associated components, you must carefully determine when an AOS instance should be available to process batches.

Walkthroughs

The following walkthroughs describe how tasks are processed, and how batch groups can be used to associate batch jobs with batch servers.

Batch processing of dependent tasks

For this example, you've created a job that is called JOB 1. As the following diagram shows, the job has seven tasks: TASK 1, TASK 2, TASK 3, TASK 4, TASK 5, TASK 6, and TASK 7.



The tasks have the following dependencies:

- TASK 1 is the first task.
- TASK 2 runs when TASK 1 is completed (regardless of the success or failure of TASK 1).
- TASK 3 runs when TASK 2 is successful.
- TASK 4 runs when TASK 2 is successful.
- TASK 5 runs when TASK 2 fails.
- TASK 6 runs when TASK 3 fails.
- TASK 7 runs when both TASK 3 and TASK 4 are successful.

Two batch servers, Batch1 and Batch2, are configured. Each server has a capacity of one thread.

Imagine that Batch1 checks for waiting tasks, assigns TASK 1 to its thread, and starts to run TASK 1. Although

Batch2 and its single thread are also available, TASK 2 continues to wait until TASK 1 is completed.

As soon as TASK 1 is completed, TASK 2 is ready to be run. This time, imagine that Batch2 checks for waiting tasks, assigns TASK 2 to its thread, and starts to run TASK 2.

If TASK 2 is successful, TASK 3 and TASK 4 are ready to be run. This time, imagine that Batch2 checks for waiting tasks, assigns TASK 3 to its thread, and starts to run TASK 3. Batch1 also checks for waiting tasks, assigns TASK 4 to its thread, and starts to run TASK 4.

If TASK 3 and TASK 4 are successful, one of the batch servers runs TASK 7.

If TASK 2 fails, one of the batch servers runs TASK 5.

If TASK 3 fails, one of the available batch servers runs TASK 6.

Note: For this walkthrough, we are using Batch1 and Batch2 to explain the concept. Any batch server that has available threads will start to run a waiting task. You must create a batch group to determine or specify which batch job runs on which server.

Batch processing that uses batch groups

This example shows how batch jobs can be processed on specific batch servers.

You have three batch servers: AOS1, AOS2, and AOS3. By default, all the batch servers process tasks from all batch jobs, depending on the number of available threads.

You create a named batch group, BG1, and configure it to run on AOS2 and AOS3. Therefore, tasks from jobs in BG1 will run only on AOS2 or AOS3, depending on the number available threads. AOS1 won't process tasks from jobs in BG1. Likewise, AOS2 and AOS3 will process tasks from BG1 only.

You can configure AOS2 and AOS3 to process tasks from other batch groups. These batch groups include the default batch group.

Batch excessive tasks configuration (Batch throttling)

Batch throttling can prevent excessive tasks by limiting the average number of executions of a certain batch class per minute. The default upper-bound is 60 tasks per minute. After that, batch framework will suspend the execution of classes for the offending class for another minute, to prevent that specific class from monopolizing the system resources.

> **NOTE**
>
> The batch framework is able to detect instances when there are no non-throttled tasks to be scheduled and executed at any given time. When this occurs, the batch will try to fetch batch tasks from the throttled classes queue to prevent resources from being idle.

Create a batch job

2/18/2021 • 2 minutes to read • Edit Online

A batch job is a group of tasks that are submitted to an Application Object Server (AOS) instance for automatic processing. Batch jobs are run by using the security credentials of the user who created the job. Use the following procedure to create a batch job. The demo data company used to create this procedure is USMF.

Create the batch job

1. Go to **Navigation pane > Modules > System administration > Inquiries > Batch jobs**.
2. Click **New**.
3. In the **Job description** field, type a value.
4. In the **Scheduled start date/time** field, enter a date and time.
5. Click **Save**.

Create a recurrence

1. On the Action Pane, click **Batch job**.
2. Click **Recurrence**. Use these options to enter a range and pattern for the recurrence.
3. Click **OK**.

Add alerts

1. On the Action Pane, click **Batch job**.
2. Click **Alerts**. Indicate if you want alert messages sent when the batch job ends, has an error, or is canceled. Then specify if you want the alerts to be displayed as pop-up messages.
3. Click **OK**.

Adjust batch job status

1. Go to **System administration > Inquiries > Batch jobs**.
2. Select the appropriate batch job.
3. On the Action Pane, click **Batch job > Functions > Change status**.
4. Select the appropriate status:
   - **Withhold**: Set the batch job as **withhold** so it is withheld from the batch job scheduler. Equivalent to *stop*.
   - **Waiting**: Set the batch job as **waiting** so it is waiting to be picked up by the batch job scheduler. Equivalent to *go*.
5. Click **OK**.

Copy a batch job

2/18/2021 • 2 minutes to read • Edit Online

When you want to create the same jobs for different legal entities, you can use the copy batch job functionality to copy an existing batch job and the batch tasks, including recurrences.

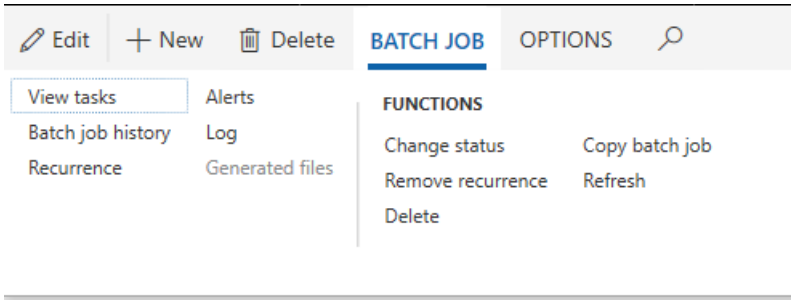
You can set the description, company, schedule start date and time, the recurrence, and the run by account at the same time. When you copy the batch job, any alerts and dependencies from the source job will also be copied.

> **NOTE**
> This feature is available as of Platform update 20.

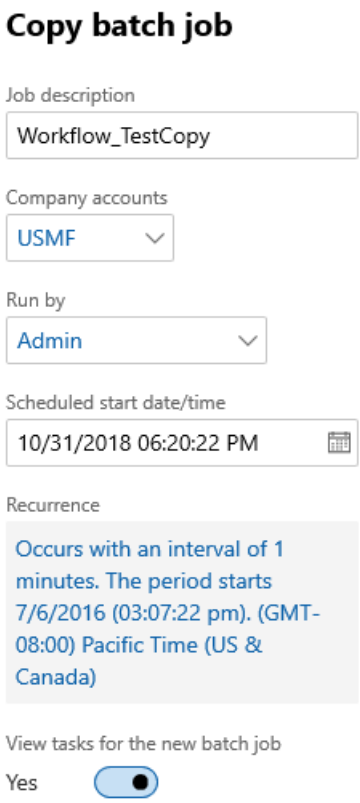
Copy a batch job

Complete the following steps to copy a batch job.

1. Click **System administration** > **Inquiries** > **Batch jobs**.
2. Select the job that you want to copy, and on the Action Pane, click **Batch Job** > **Copy batch job**.



4. Enter or add any changes. If you set **View tasks** to **Yes**, when you click **OK** you will go directly to the **Batch tasks** page for the copied job.



IMPORTANT

The copied batch job will be created with a **Withhold** status, so you will need to enable it. The **Run by** user can also be set to give this user the privilege to run the job without being a Sys Admin.

Enable the batch job

Complete the following steps to enable a batch job.

1. On the **Batch job** page, on the Action Pane, click **Batch job** > **Change status**.
2. Select the **Waiting** status, and then click **OK**.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Active batch periods

2/18/2021 • 2 minutes to read • Edit Online

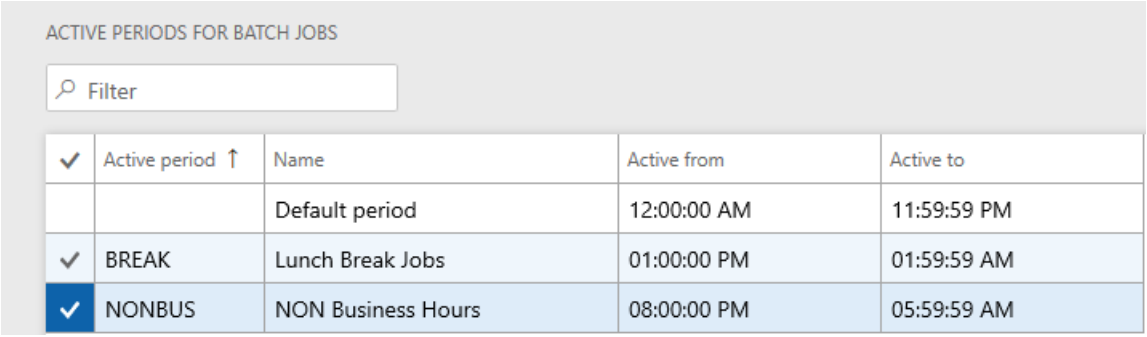
With the release of Platform update 21, an additional level of control over when batch jobs execute is now available. Previously, it was only possible to schedule a batch job to execute every hour for a specified number of hours or until a given date. Administrators can now provide information for an additional active period, such as in the following scenarios:

- Specifying time ranges during which jobs within a batch group can start execution.
- Selecting to run batch jobs outside of office hours only.
- Setting the recurrence for anytime within the active period. For example, you administrator might select to run the batch jobs every hour, but only between the hours of 6:00 PM and 8:00 AM.

> **NOTE**
> This feature is available as of Platform update 21.

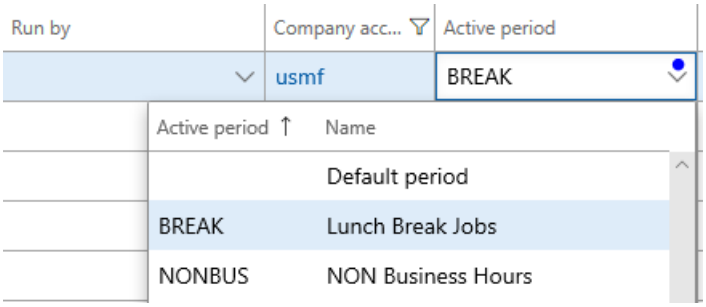
Set up active periods for batch jobs

1. Go to **System administration** > **Setup** > **Active periods for batch jobs**.
2. Enter the name of the batch job, and specify start and end dates that the batch job is active.
3. Click **Save**.



Assign active periods to batch jobs

1. Go to **System administration** > **Inquiries** > **Batch jobs**.
2. Select the batch job that you want to assign a period to, and click **Edit**.
3. In the **Active period** field, select the active period that you want to assign, and then click **Save**.



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Daylight saving time support for active batch periods

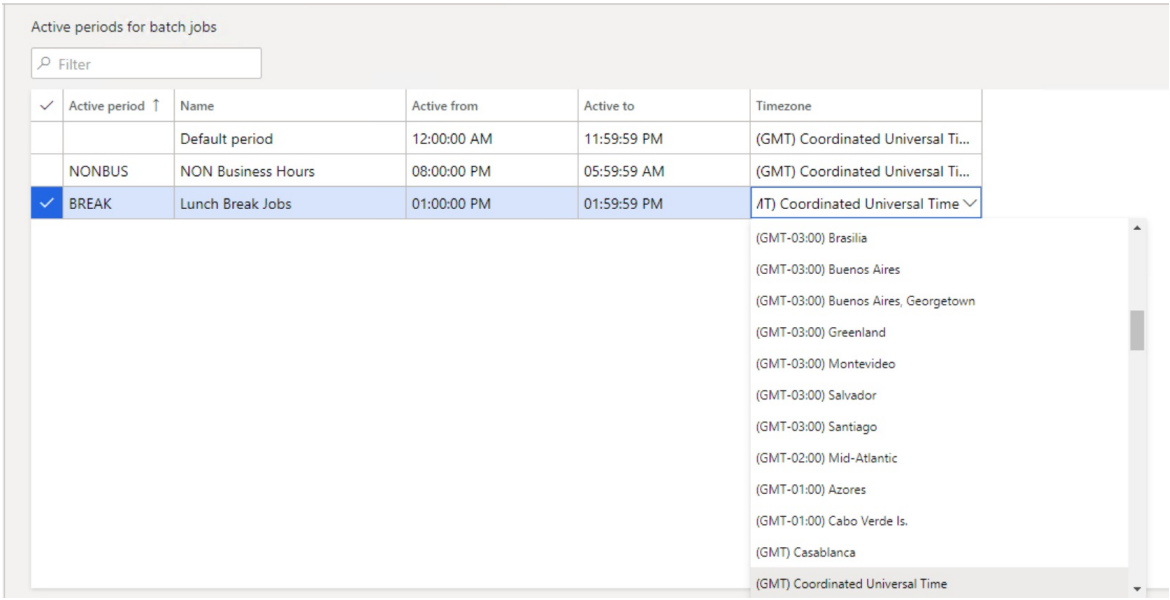
2/18/2021 • 2 minutes to read • Edit Online

Microsoft Dynamics 365 Finance version 10.0.12 includes a **Daylight Saving Time support for batch job active periods** feature that can be turned on in Feature management. This feature introduces daylight saving time (DST) support for the active periods for batch jobs and lets users associate their active periods with different time zones.

> **NOTE**
>
> This feature is a one-way feature. In other words, it can't be turned off after it's turned on.

When this feature is turned on, the following changes occur:

- On the **Active periods for batch jobs** page, a **Timezone** field is added for each active period. This field specifies the time zone that the active period uses. By default, every active period initially uses the Coordinated Universal Time (UTC) time zone.



- The start and end times of existing active periods are adjusted according to the UTC time zone. 'Although the active periods will continue to start and end at the same times that they previously started and ended, the times that are shown might change if the user's preferred time zone isn't UTC.
- Active periods will follow the DST adjustments of the time zones that they are associated with.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Batch manager security role

2/18/2021 • 2 minutes to read • Edit Online

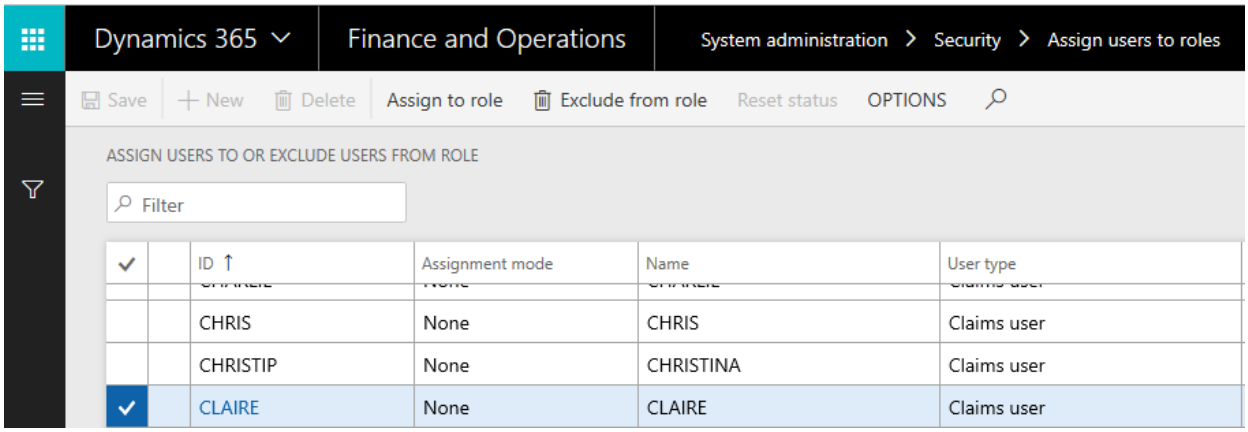
Before Platform update 20, users needed to be assigned to the system admin or IT admin security role to manage batch jobs. With the release of Platform update 20, there is a more targeted role, Batch manager. With this security role, a user now has permissions to copy batch jobs, change who will execute jobs, and specify the time ranges during which jobs can execute. The Batch maintain security privilege is part of the Batch manager security role and it allows a user to create an ad hoc batch job and grant privileges to other users.

> **NOTE**
>
> This feature is available as of Platform update 20.

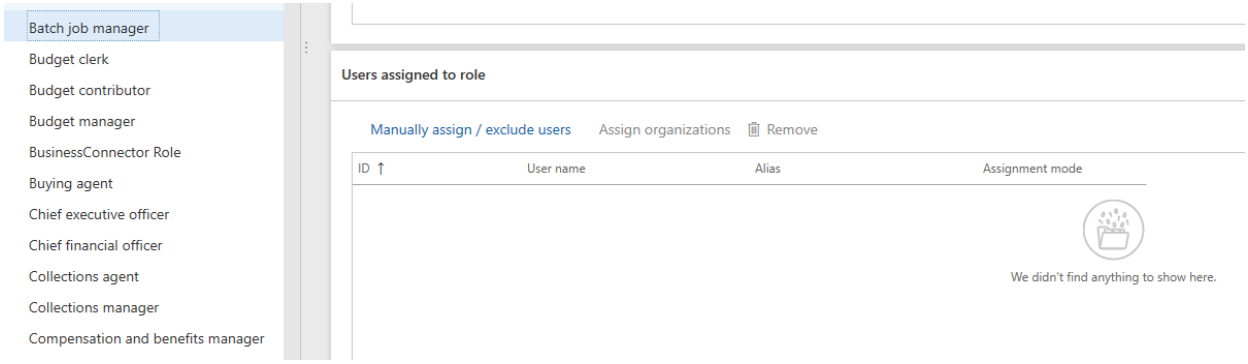
Assign the Batch manager role to a user

Complete the following steps to assign the Batch manager security role to a specific user.

1. Select **System administration** > **Security** > **Assign users to roles**.



2. Select **Batch Job Manager**, and on the left pane, select **Manually assign/exclude user**.

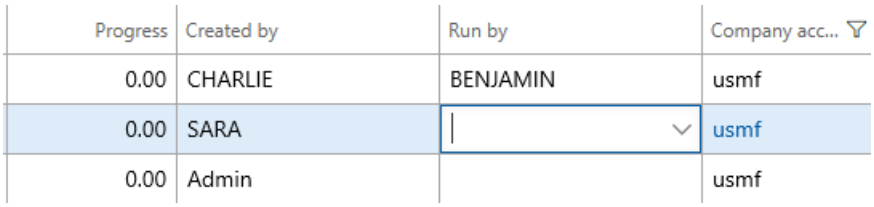


3. Select a user from the list, and then select **Assign to role**.
4. Close the page.

Run by user

The **run by user** functionality allows Batch managers to specify a user to run the batch job. This functionality is useful when you want to change the user who is currently assigned to run the job or if you want to quickly set a

user while copying the batch jobs from one company to another. You can also use this functionality to copy batch jobs.



Progress	Created by	Run by	Company acc...
0.00	CHARLIE	BENJAMIN	usmf
0.00	SARA		usmf
0.00	Admin		usmf

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Set up alerts

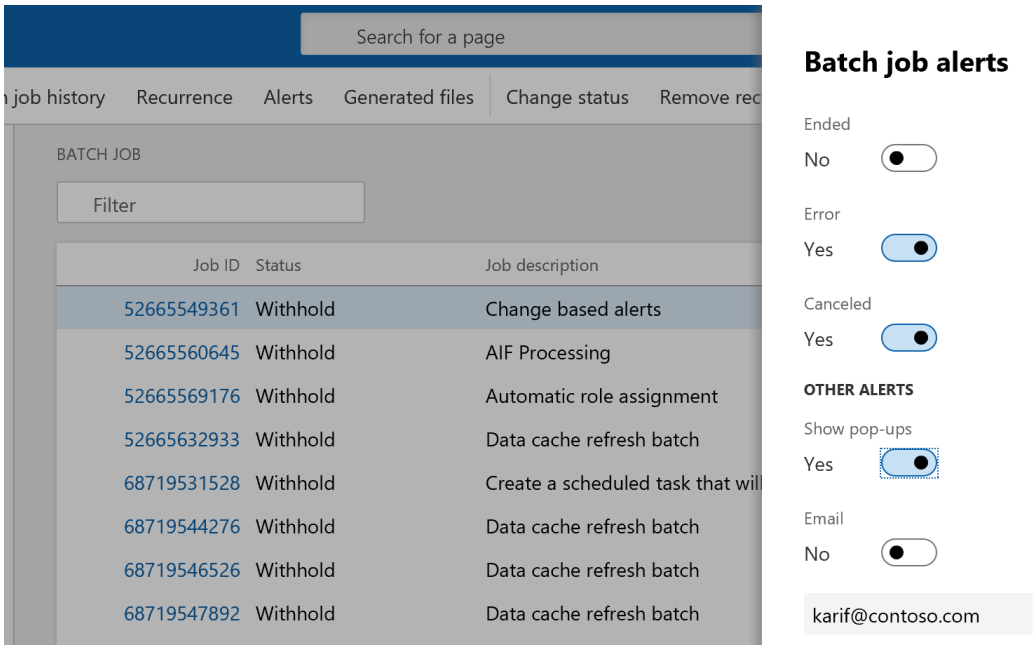
2/18/2021 • 2 minutes to read • Edit Online

Alerts form a notification system for critical events in Finance and Operations. You can use alerts to stay informed about events that you want to track during the workday. You can set up a set of alert rules so that you're alerted when a batch job ends, ends in error, or is canceled. You can select whether the alerts are emailed to you or appear as notifications in the Action center. Alerts can be set up per batch job and per user.

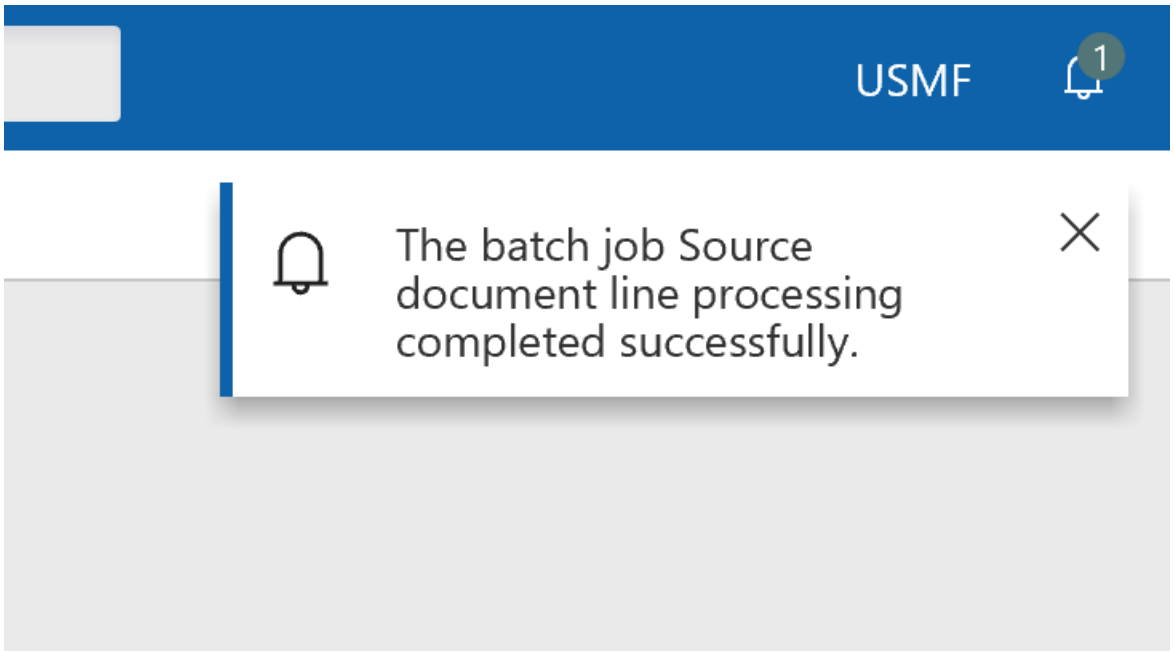
Set up alerts for batch enhanced forms

Follow these steps to set up alerts for batch enhanced forms.

1. Go to **System administration** > **Inquiries** > **Batch jobs**.
2. Select a batch job in the list, and then, on the Action Pane, select **Alerts**.
3. In the **Batch job alerts** dialog box, configure the alerts, and then select **OK**.



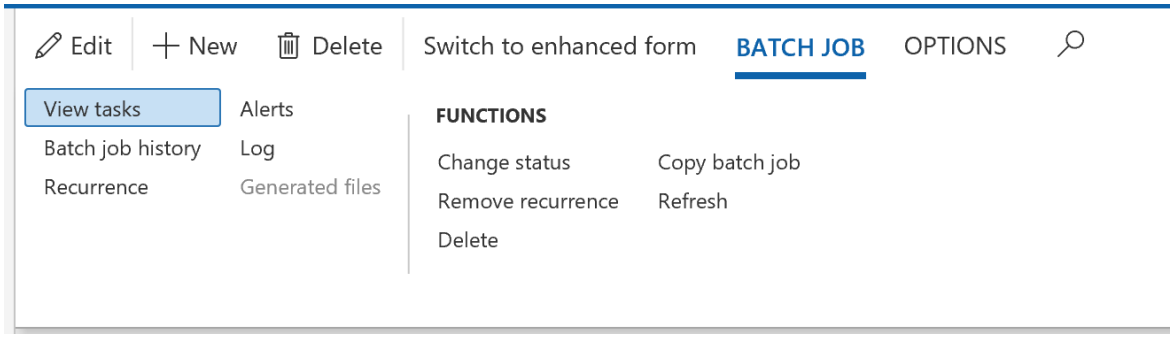
4. Check the Action center for alert notifications.



Set up alerts for batch legacy forms

Follow these steps to set up alerts for batch legacy forms.

1. Go to **System administration** > **Inquiries** > **Batch jobs**.
2. Select a batch job in the list, and then, on the Action Pane, on the **Batch job** tab, select **Alerts**.



3. In the **Batch job alerts** dialog box, configure the alerts, and then select **OK**.

> **NOTE**
>
> To receive email notifications, in the **Batch job alerts** dialog box, set the **Email** option to **Yes**.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Enhanced batch forms

2/18/2021 • 2 minutes to read • Edit Online

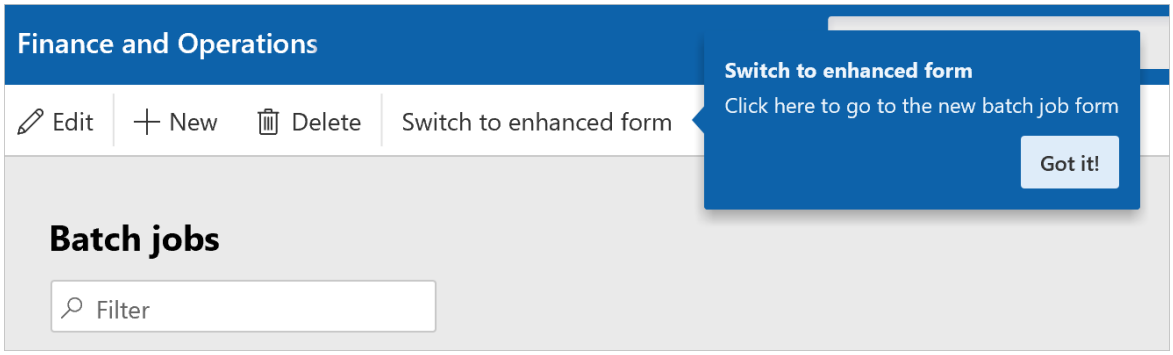
You can open an enhanced detail transaction form by selecting the job ID for a batch job. The enhanced form provides a header and lines that summarize the batch tasks and constraints that are related to the selected batch job.

Switch to the enhanced form

Follow these steps to switch to the enhanced form.

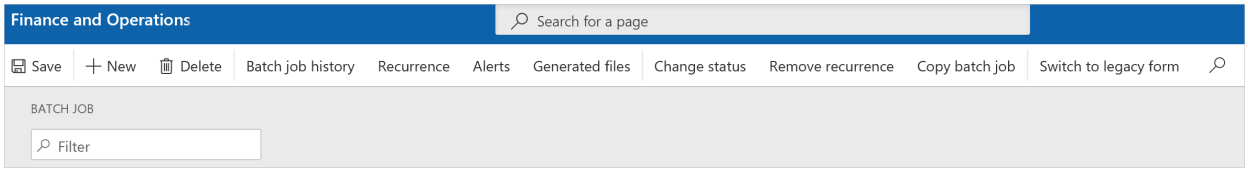
1. Go to **System administration** > **Inquiries** > **Batch jobs**.

   You're notified about the enhanced form. The notification shows the location of the **Switch to enhanced form** button on the Action Pane.



2. Select **Switch to enhanced form**.

To switch back to the unenhanced form, select **Switch to legacy form** on the Action Pane of the enhanced form.



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

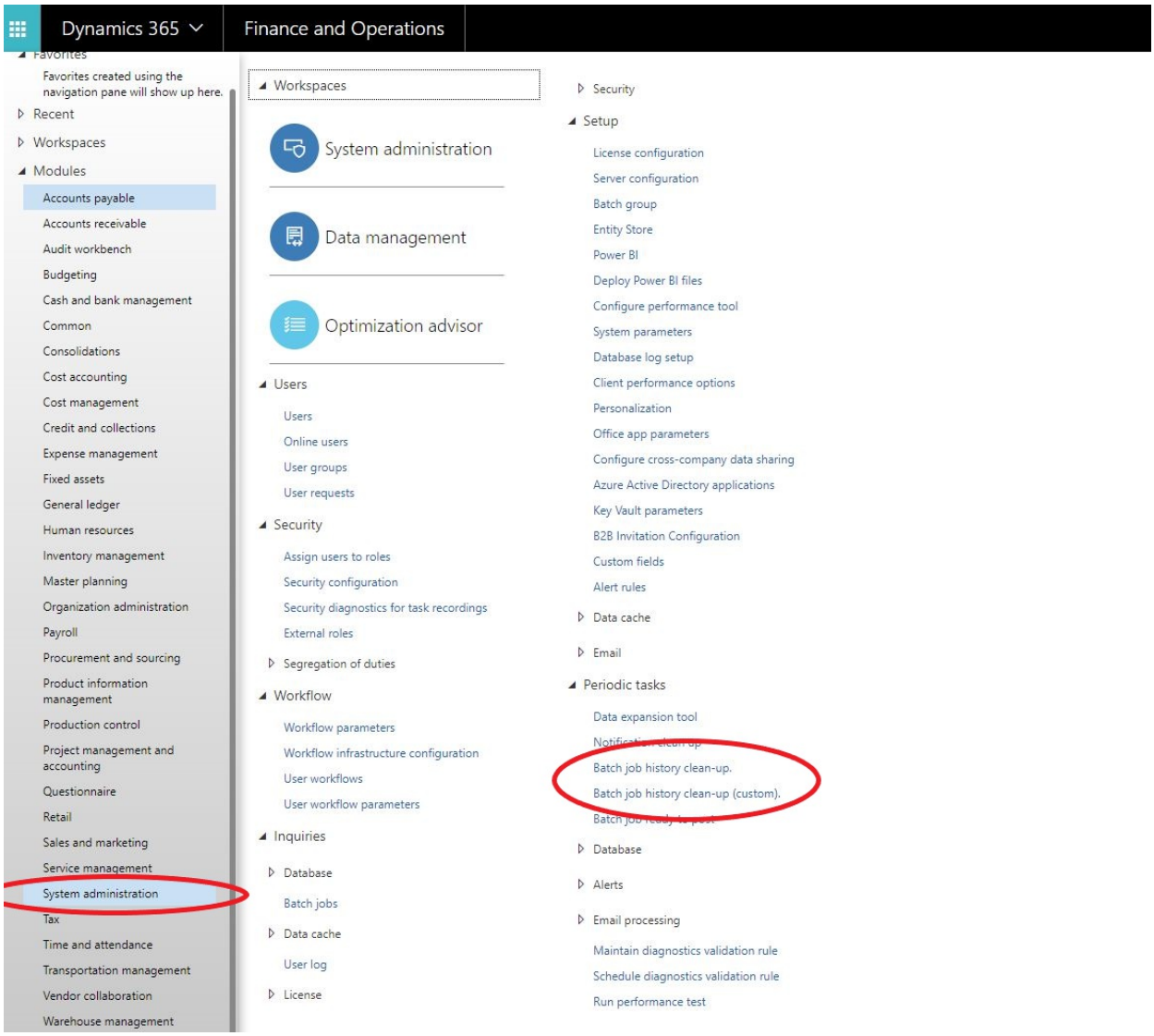
Clean up the batch job history

2/18/2021 • 2 minutes to read • Edit Online

When you run a batch job, a history is recorded. This history can be used to monitor the correct execution of jobs. However, when several batch jobs have been created, especially batch jobs that have a high recurrence, lots of batch job history entries are generated. Too many entries in the history table can negatively affect the performance of future jobs.

Two pages that have been added to the **System administration** module make it easy to clean up the batch job history:

- Batch job history clean-up
- Custom batch job history clean-up



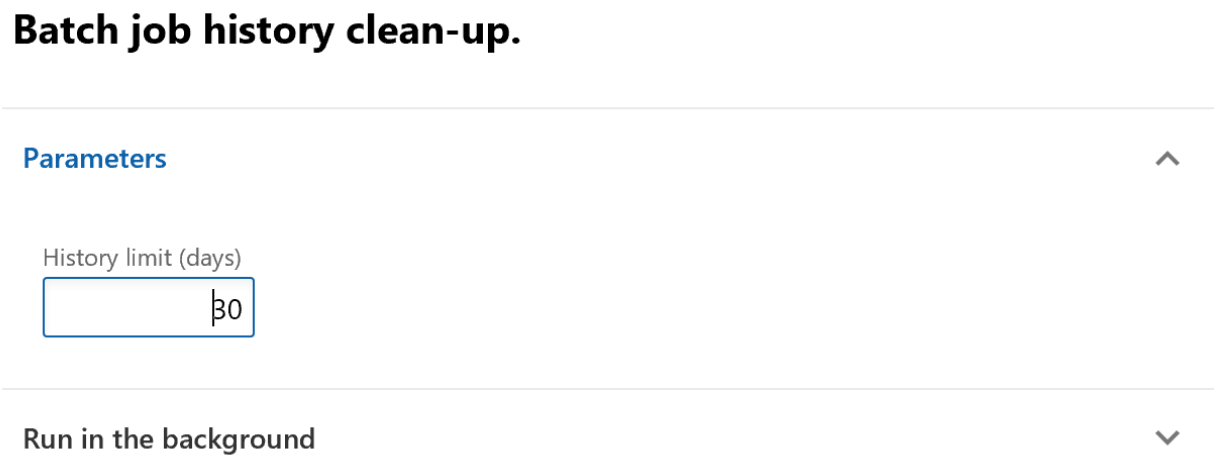
NOTE

We recommend that you regularly clean up the batch job history, and that you do this cleanup outside of business hours.

Batch job history clean-up

Follow these steps to quickly clean up all history entries that are older than a specified number of days.

1. On the **Periodic tasks in System administration** module, select **Batch job history clean-up**.
2. In the **History limit (days)** field, specify the number of days to keep a history of batch jobs.
3. Select **OK**.



Batch job history clean-up (custom)

The custom batch job lets you to apply additional filtering, based on criteria such as status, job description, company, or user. You can also add other filter criteria by selecting the **Filter** button.

1. On the **Periodic tasks in System administration** module, select **Batch job history clean-up (custom)**.
2. In the **History limit (days)** field, specify the number of days to keep a history of batch jobs.
3. On the **Records to include** FastTab, specify any filter criteria that you require, and then select **OK**.
4. Select **OK**.

Batch job history clean-up (custom).

Parameters

History limit (days)

30

Records to include

Filter

BATCH JOBS HISTORY

Status

Job description

End date/time

Company

Created by

Run in the background

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Cancel an executing batch job

2/18/2021 • 2 minutes to read • Edit Online

NOTE

This feature is available as of Platform update 27.

Sometimes canceling a batch job can take a long time if already executing tasks will take a long time to finish. Canceling the batch job provides a system administrator or batch job manager with the ability to cancel already executing tasks for jobs that are in the process of being canceled. This provides a much faster mechanism to cancel a long running job that is impacting system usage elsewhere.

NOTE

It is important to note that this feature should be used with caution. When you cancel a running process, it is an inherently unsafe action that can lead to data corruption, resulting in either orphaned or incomplete data. This action should only be used to mitigate other issues caused by the running tasks.

Complete the following steps to immediately cancel the running task.

1. Go to **System administration** > **Inquiries** > **Batch jobs**.
2. Select a batch job that has a **Status** of **Canceling**.
3. On the **Batch tasks** tab, select **Cancel** on the task, and then select **OK**.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Optimization advisor overview

2/18/2021 • 4 minutes to read • Edit Online

This topic describes how you can use Optimization advisor to help ensure optimal configuration of Finance and Operations.

Overview

Incorrect configuration and setup of a module can adversely affect the availability of application features, system performance, and the smooth operation of business processes. The quality of business data (for example, the correctness, completeness, and cleanliness of the data) also affects system performance, and an organization's decision-making capabilities, productivity, and so on.

The **Optimization advisor** workspace is a tool that lets power users, business analysts, functional consultants, and IT support functions identify issues in module configuration and business data. Optimization advisor suggests best practices for module configuration and identifies business data that is obsolete or incorrect.

Optimization advisor periodically runs a set of best practice rules. A default set of rules is available, however users can also create rules that are specific to their customizations, solutions from independent software vendors (ISVs), and business data. For more information about how to create rules, see Create rules for Optimization advisor.

When a violation of a rule is detected, an optimization opportunity is generated and appears in the **Optimization advisor** workspace. A user can take appropriate corrective action directly from the **Optimization advisor** workspace.

Opportunities can be company-specific or cross-company, depending on the type of setup and data that is being validated. Cross-company opportunities can be viewed from all companies. To view the opportunities for a specific company, you must first select the company.

Standard security policies apply to optimization opportunities. For example, the optimization opportunities that are related to configuration of the **Warehouse management** module are visible only to users who have access to Warehouse management and can change its setup.

When you take action on some optimization opportunities, the system calculates the impact of the opportunity in terms of the reduction in the runtime of business processes. Unfortunately, this feature isn't available for all optimization opportunities.

To learn more about Optimization advisor, watch the short Optimization advisor in Dynamics 365 for Finance and Operations video.

Optimization rules

To view the complete list of Optimization advisor rules and to see how often the rules are evaluated, go to **System administration** > **Periodic tasks** > **Maintain diagnostics validation rule**. Only rules that have a status of **Active** are evaluated. The evaluation frequency can be set to **Daily**, **Weekly**, **Monthly**, or **Unscheduled**.

To trigger the evaluation of unscheduled rules, or to reevaluate periodic rules outside their predefined schedule, go to **System administration** > **Periodic tasks** > **Schedule diagnostics validation rule**. Then, in the **Diagnostic rule validation** dialog box, select an evaluation frequency. All rules that have the specified frequency will be reevaluated.

The current set of optimization rules can be divided into the following categories.

Module configuration and setup

The setup of Warehouse management is a complicated process. To make the process easier, some rules have been introduced to help validate the correctness of the setup. For example, one rule validates the setup of warehouse location directives for fixed product variant locations for sales orders and transfer orders.

Additionally, some rules check whether features that have been enabled are actually used. For example, one rule determines whether you're using the **Master planning** module. If the rule determines that you aren't using the module, an optimization opportunity is generated to suggest that you turn off the planning processes.

System configuration

If specific functionality that is controlled by a configuration key isn't used, an optimization opportunity is generated to suggest that you disable the configuration key. Examples of configuration keys include **Catch weight**, **Budget planning**, **Project**, and **Approved vendor list**.

Business data consistency and cleanup

If master data isn't correct (for example, if you have unit of measure conversions for units that haven't been defined, or if you have unit of measure conversions that have a division by 0 [zero]), an optimization opportunity is generated to suggest that you correct the data.

If you have too many batch job history entries, obsolete items, closed on-hand entries for warehouse enabled items, and so on, or if those entries and items are too old, optimization opportunities are generated to suggest that you clean up the data. By keeping your data clean, you can help improve overall system performance.

Best practices

If you aren't running some business processes according to best practices (for example, if you run inventory pre-closing before the inventory is closed, or if you use the scheduled batch for subledger journal batch transfer), optimization opportunities inform you about the best practice and ask that you follow it.

Optimization opportunities

To view the optimization opportunities that are generated during the evaluation of optimization rules, open the **Optimization advisor** workspace.

In this workspace, you can view more information about an opportunity by selecting **More information**. If you want the system to take action and correct the setup, clean the data, and so on, so that you don't have to open the corresponding pages yourself, select **Take action**.

There is no workflow for optimization opportunities. After you select **Take action** or use a navigation path that is provided in the **More information** dialog box, the optimization opportunity disappears from the list. If the corrective action doesn't completely resolve an issue, the opportunity will be generated again the next time that the rule is evaluated.

If an opportunity doesn't apply to your role, you can select **Hide from my list**. Even if the rule behind this opportunity is triggered again later, you won't see the opportunity in your list.

To deactivate the evaluation of specific rules, select the opportunity that was generated by the rule, and then select **Deactivate analysis**.

Additional resources

Create rules for Optimization advisor

Optimization advisor in Dynamics 365 for Finance and Operations (Video)

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Create rules for Optimization advisor

2/18/2021 • 6 minutes to read • Edit Online

This topic explains how to create new rules for **Optimization advisor**. For example, you can create a new rule that identifies which Request for Quotations (RFQ) cases have an empty title. Using titles on cases makes them easily identifiable and searchable. While quite simple, this example shows what can be achieved with optimization rules.

A *rule* is a check on application data. If the condition that the rule evaluates is met, opportunities to optimize processes or improve data are created. The opportunities can be acted upon and, optionally, the impact of the actions can be measured.

To create a new rule for the **Optimization advisor**, add a new class that extends the **SelfHealingRule** abstract class, implements the **IDiagnosticsRule** interface, and is decorated by the **DiagnosticRule** attribute. The class must also have a method decorated with the **DiagnosticsRuleSubscription** attribute. By convention, that is done on the **opportunityTitle** method, which will be discussed later. This new class can be added to a custom model with a dependency on the **SelfHealingRules** model. In the following example, the rule being implemented is called **RFQTitleSelfHealingRule**.

```
[DiagnosticsRule]
public final class RFQTitleSelfHealingRule extends SelfHealingRule implements IDiagnosticsRule
{
…
}
```

The **SelfHealingRule** abstract class has abstract methods that must be implemented in inheriting classes. The core is the **evaluate** method, which returns a list of the opportunities identified by the rule. Opportunities can be per legal entity or can apply to the whole system.

```
protected List evaluate()
{
    List results = new List(Types::Record);

    DataArea dataArea;

    while select id from dataArea
        where !dataArea.isVirtual
    {
        changecompany(dataArea.id)
        {
            container result = this.findRFQCasesWithEmptyTitle();

            if (conLen(result) > 0)
            {
                SelfHealingOpportunity opportunity = this.getOpportunityForCompany(dataArea.Id);
                opportunity.EvaluationState = SelfHealingEvaluationState::Evaluated;
                opportunity.Data = result;
                opportunity.OpportunityDate = DateTimeUtil::utcNow();

                results.addEnd(opportunity);
            }
        }
    }

    return results;
}
```

The method shown above loops over companies and selects RFQ cases with empty titles in the **findRFQCasesWithEmptyTitle** method. If at least one such case is found, then a company-specific opportunity is created with the **getOpportunityForCompany** method. Notice that the field **Data** in the **SelfHealingOpportunity** table is of type **Container**, and can therefore contain any data relevant to the logic specific to this rule. Setting **OpportunityDate** with the current timestamp registers the time of the latest evaluation of the opportunity.

Opportunities can also be cross-company. In this case, the loop over companies is not necessary and the opportunity must be created with the **getOpportunityAcrossCompanies** method.

The following code shows the **findRFQCasesWithEmptyTitle** method, which returns the IDs of the RFQ cases that have empty titles.

```
private container findRFQCasesWithEmptyTitle()
{
    container result;

    PurchRFQCaseTable rfqCase;
    while select RFQCaseId from rfqCase
        where rfqCase.Name == ''
    {
        result += rfqCase.RFQCaseId;
    }

    return result;
}
```

Two more methods that must be implemented are **opportunityTitle** and **opportunityDetails**. The former returns a short title for the opportunity, the latter returns a detailed description of the opportunity, which can also include data.

The title returned by **opportunityTitle** appears under the **Optimization opportunity** column in the **Optimization advisor** workspace. It also appears as the header of the side pane showing more information

about the opportunity. By convention, this method is decorated with the **DiagnosticRuleSubscription** attribute, which takes the following arguments:

- **Diagnostic area** – An enum of type **DiagnosticArea** that describes what area of the application the rule belongs to, such as **DiagnosticArea::SCM**.
- **Rule name** – A string with the rule name. This will appear under the **Rule name** column in the **Dianostics validation rule** form (**DiagnosticsValidationRuleMaintain**).
- **Run frequency** – An enum of type **DiagnosticRunFrequency** that describes how often the rule should be run, such as **DiagnosticRunFrequency::Daily**.
- **Rule description** – A string with a more detailed description of the rule. This will appear under the **Rule description** column in the **Dianostics validation rule** form (**DiagnosticsValidationRuleMaintain**).

> **NOTE**
>
> The **DiagnosticRuleSubscription** attribute is required for the rule to work. Typically, it is used on **opportunityTitle**, but it can decorate any method of the class.

The following is an example implementation. Raw strings are used for simplicity, but a correct implementation requires labels.

```
[DiagnosticsRuleSubscription(DiagnosticsArea::SCM,
                             'Assign titles to Request for Quotation cases',
                             DiagnosticsRunFrequency::Daily,
                             'This rule detects Requests for Quotation with empty titles.')]
public str opportunityTitle()
{
    return 'Assign titles to Request for Quotation cases';
}
```

The description returned by **opportunityDetails** appears on the side pane showing more information about the opportunity. This takes the **SelfHealingOpportunity** argument, which is **Data** field that can be used to provide more details about the opportunity. In the example, the method returns the IDs of the RFQ cases with an empty title.

```
public str opportunityDetails(SelfHealingOpportunity _opportunity)
{
    str details = '';
    container opportunityData = _opportunity.Data;
    int affectedRFQCasesCount = conLen(opportunityData);

    if (affectedRFQCasesCount != 0)
    {
        details = 'The following Request for Quotation cases have an empty title:\n';
        for (int i = 1; i <= affectedRFQCasesCount ; i++)
        {
            PurchRFQCaseId rfqCaseId = conPeek(opportunityData, i);
            details += rfqCaseId + '\n';
        }
    }

    return details;
}
```

The two remaining abstract methods to implement are **provideHealingAction** and **securityMenuItem**.

provideHealingAction returns true if a healing action is provided, otherwise, it returns false. If true is returned,

the method **performAction** must be implemented, or an error will be thrown. The **performAction** method takes a **SelfHealingOpportunity** argument, in which the data can be used for the action. In the example, the action opens the **PurchRFQCaseTableListPage**, for manual correction.

```
public boolean providesHealingAction()
{
    return true;
}

protected void performAction(SelfHealingOpportunity _opportunity)
{
    new MenuFunction(menuItemDisplayStr(PurchRFQCaseTableListPage), MenuItemType::Display).run();
}
```

Depending on the specifics of the rule, it might be possible to take an automatic action using the opportunity data. In this example, the system could generate titles for RFQ cases automatically.

securityMenuItem returns the name of an action menu item such that the rule is only visible to users who can access the action menu item. Security might require that specific rules and opportunities are accessible only to authorized users. In the example, only users with access to **PurchRFQCaseTitleAction** can view the opportunity. Notice that this action menu item was created for this example, and was added as an entry point for the **PurchRFQCaseTableMaintain** security privilege.

> **NOTE**
> The menu item must be an action menu item for security to work correctly. Other menu item types, such as **Display menu items** will not work correctly.

```
public MenuName securityMenuItem()
{
    return menuItemActionStr(PurchRFQCaseTitleAction);
}
```

After the rule has compiled, execute the following job to have it display in the user interface (UI).

```
class ScanNewRulesJob
{
    public static void main(Args _args)
    {
        SysExtensionCache::clearAllScopes();
        var controller = new DiagnosticsRuleController();
        controller.runOperation();
    }
}
```

The rule will display in the **Diagnostics validation rule** form, available from **System administration** > **Periodic tasks** > **Maintain diagnostics validation rule**. To have it evaluated, go to **System administration** > **Periodic tasks** > **Schedule diagnostics validation rule**, select the frequency of the rule, such as **Daily**. Click **OK**. Go to **System administration** > **Optimization advisor** to view the new opportunity.

The following example is a code snippet with the skeleton of a rule including all the required methods and attributes. It helps you get started with writing new rules. The labels and action menu items that are used in the example are only used for demonstration purpose.

```
[DiagnosticsRuleAttribute]
public final class SkeletonSelfHealingRule extends SelfHealingRule implements IDiagnosticsRule
{
    [DiagnosticsRuleSubscription(DiagnosticsArea::SCM,
                                 "@SkeletonRuleLabels:SkeletonRuleTitle", // Label with the title of the
rule
                                 DiagnosticsRunFrequency::Monthly,
                                 "@SkeletonRuleLabels:SkeletonRuleDescription")] // Label with a description
of the rule
    public str opportunityTitle()
    {
        // Return a label with the title of the opportunity
        return "@SkeletonRuleLabels:SkeletonOpportunityTitle";
    }

    public str opportunityDetails(SelfHealingOpportunity _opportunity)
    {
        str details = "";

        // Use _opportunity.data to provide details on the opportunity

        return details;
    }

    protected List evaluate()
    {
        List results = new List(Types::Record);

        // Write here the core logic of the rule

        // When creating an opportunity, use:
        //     * this.getOpportunityForCompany() for company specific opportunities
        //     * this.getOpportunityAcrossCompanies() for cross-company opportunities

        return results;
    }

    public boolean providesHealingAction()
    {
        return true;
    }

    protected void performAction(SelfHealingOpportunity _opportunity)
    {
        // Place here the code that performs the healing action

        // To open a form, use the following:
        // new MenuFunction(menuItemDisplayStr(SkeletonRuleDisplayMenuItem), MenuItemType::Display).run();
    }

    public MenuName securityMenuItem()
    {
        return menuItemActionStr(SkeletonRuleActionMenuItem);
    }

}
```

For more information, watch the short YouTube video: Optimization advisor in Dynamics 365 for Finance and Operations

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Report a production outage

2/18/2021 • 2 minutes to read • Edit Online

Lifecycle Services (LCS) has a feature called **Report production outage**. This feature is available to all customers who have purchased one or more Dynamics 365 Finance and Operations apps and have implementation projects with a production environment deployed in LCS. This feature provides a quick and effective channel to escalate issues to Microsoft Support in the event that the services in a production environment are degraded or become unavailable.

Following mutually inclusive conditions, a production outage can be defined as one or more system-wide issues on a live production environment that impact multiple users and prevent your business from performing daily operations.

Reporting flow

The following list shows the order in which an issue should be handled:

1. In a live production environment, a customer experiences an outage or other situation with prevents business from continuing.
2. The customer reports a production outage issue by using the LCS Support portal.
3. The customer selects a production outage issue and provides additional information.
4. A Microsoft support engineer acknowledges the production outage ticket within 30 minutes of submission and begins to immediately collaborate with stakeholders to investigate and resolve the issue.
5. A support engineer contacts the customer to provide a status update.

Access and availability

All users who have been added to a customer's implementation project have access to this feature. This includes project owners, organization admins, team members, and environment managers.

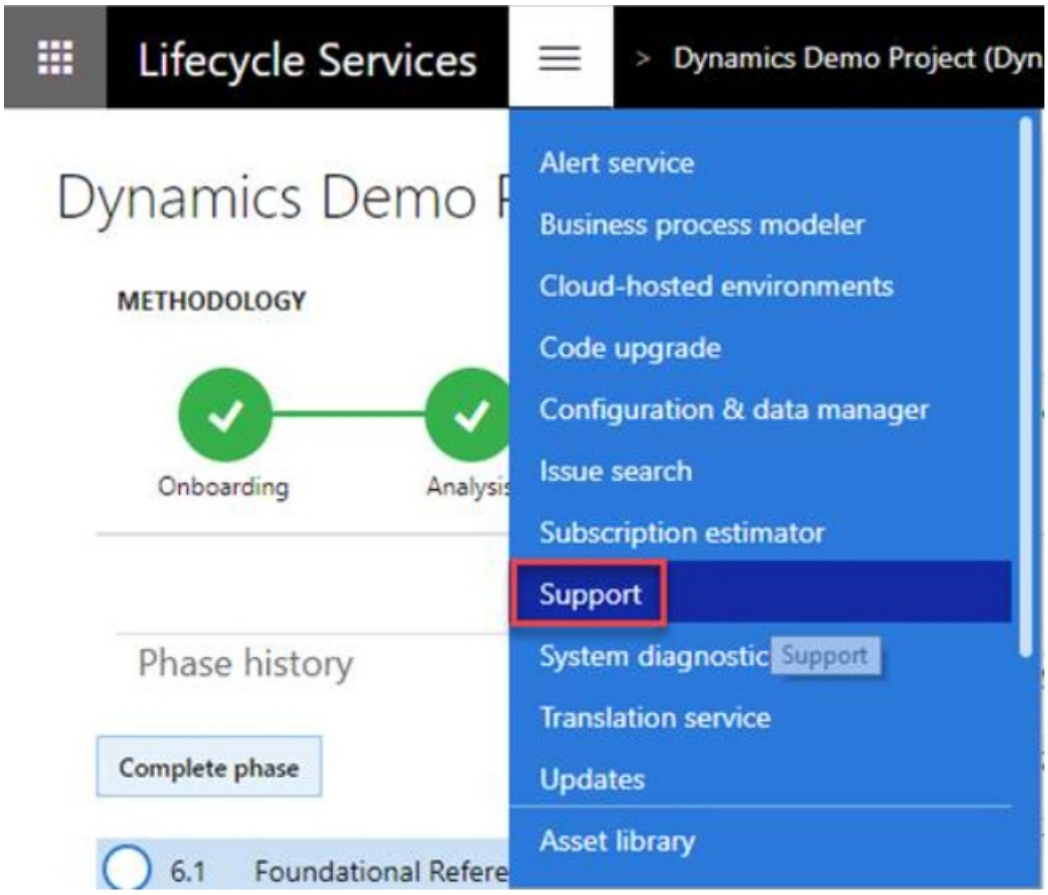
This feature is available for:

- Dynamics 365 Finance
- Dynamics 365 Supply Chain Management
- Environments that are managed by Microsoft
- A production environment in the LCS project
- All support plans

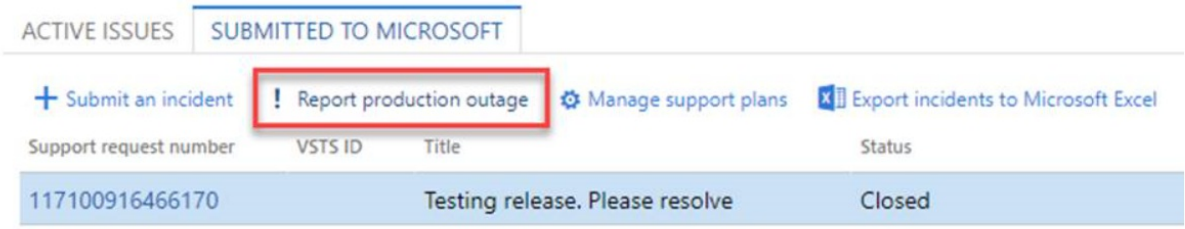
Report a production outage

To report a production outage, follow these steps:

1. Log in to your LCS project.
2. From the hamburger menu, click **Support**.



3. On the **Submitted To Microsoft** tab, click **Report production outage**.



4. Confirm the production outage, select the outage scenario from the drop-down list, and then click **Continue**.
5. Add a title and details about the outage, and then click **Next**.
6. Provide contact information, and then click **Next**.
7. Click **Done**.

If you're unable to report a production outage in LCS, phone support is available.

> **NOTE**
>
> If you don't see your situation listed in the outage scenarios, enter a support incident through LCS. During the initial investigation by a Microsoft support engineer, if it is found that the situation does not meet the current list of production outage scenarios, the support incident will be transferred to the correct support team and service-level agreement (SLA) based on your current support plan.

Configure database logging

2/18/2021 • 3 minutes to read • Edit Online

Database logging provides a way to track specific types of changes to the tables and fields in Finance and Operation apps. Changes that can be tracked include insert, update, delete, and rename key operations. When you configure logging for a table or field, a record of every change to that table or field is stored in the database log table, **sysdatabaselog**, in the environment database.

Database logging can be used for these purposes:

- Create an auditable record of changes to specific tables that contain sensitive information.
- Monitor the use of electronic signatures. By default, all transactions that have been signed by using electronic signatures are logged.

Database logging is intended to track individual transactions. It isn't intended to track automated transactions that are run in batch jobs.

Security for database logging

Database logs can contain sensitive data. By default, any user who has database access can query the database log table (**sysdatabaselog**) by using X++ or alerts, or by querying the database directly. To help protect data, you should restrict permissions on the **sysdatabaselog** table for on-premises deployments.

Database logging and performance

Although database logging can be valuable from a business perspective, it can be expensive with regard to resource use and management. Here are some of the performance implications of database logging:

- The database log table can grow quickly and can increase the size of the database. The amount of growth depends on the amount of logged data that you decide to retain.
- When logging is turned on for a transaction type, each instance of that transaction type causes multiple records to be written to the Microsoft SQL Server transaction log file. Specifically, one record is written for the initial transaction, and one record logs the transaction in the database log table. Therefore, the transaction log file will grow more quickly and might require additional maintenance.
- Database logging can adversely affect long-running automated processes, such as inventory close, calculations for bills of materials (BOMs), master planning, and long-running data imports.
- When logging is turned on for a table, all set-based database operations are downgraded to row-based operations. For example, if you're logging inserts for a table, each insert is done as a row-based insert.

Here are some practices that Microsoft recommends:

- Create a plan for how long you will retain logged data, and how you will archive or delete data.
- Limit log entries, and help improve performance by selecting specific fields to log instead of whole tables.

  > **NOTE**
  > Only updates can be logged for individual fields.

- After you configure database logging, consider increasing the frequency of backups of the SQL Server transaction log.

> **NOTE**
>
> This recommendation applies only to on-premises deployments that have a local instance of SQL Server.

Set up database logging

You can use the **Logging database changes** wizard to set up database logging. This wizard provides a flexible way to set up logging for tables or fields.

1. Go to **System administration** > **Setup** > **Database log** > **Database log setup**.
2. Select **New** to open the **Logging database changes** wizard.
3. Complete the wizard.

Clean up database logs

You can delete database logs as required. You can delete logs for specific tables, delete specific types of database logs, or delete logs based on the date and time when they were created.

> **NOTE**
>
> Records that have been electronically signed can't be deleted from logs.

1. Go to **System administration** > **Inquiries** > **Database** > **Database log**.
2. Select **Clean up log**.
3. Select the method that should be used to select the logs that are deleted. Enter the table ID that the logs refer to, the type of log, or the creation date and time.
4. Use the **Database log cleanup** tab to specify when the log cleanup task should be run.

Consistency check for database log triggers

In Platform update 34, functionality for a consistency check was added. The consistency check is run as part of the **Database log** wizard. It's run after you select **Finish** or after you select **Consistency check** on the **Database log setup** page.

The consistency check will re-create any missing database log triggers. It will also drop any "orphaned" database log triggers that no corresponding configuration is found for. In this way, the consistency check quickly detects and fixes any inconsistencies between the current configuration and the database triggers that are used to implement the logging functionality.

1. Go to **System administration** > **Inquiries** > **Database** > **Database log**.
2. On the **Database log** page, select **Consistency check**.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Build OData metadata cache when the AOS starts

2/18/2021 • 2 minutes to read • Edit Online

With the release of Platform update 32, we have introduced the ability to build OData metadata cache when the Application Object Server (AOS) starts, instead of when the first OData request is made. This significantly decreases the response time for the first OData call after an AOS process restart.

This option is useful if your business process can't wait for the OData metadata cache to be built each time that the AOS process restarts. Follow these steps to turn on this feature.

1. Go to **System administration** > **Setup** > **System parameters**.
2. On the **General Tab**, select **Build metadata cache when AOS starts**, and then select **Save**.

> **NOTE**
> When you enable this functionality, the AOS should already be running and should have served one OData request. This means that the cache is already built. This new functionality will take effect during the next AOS restart.

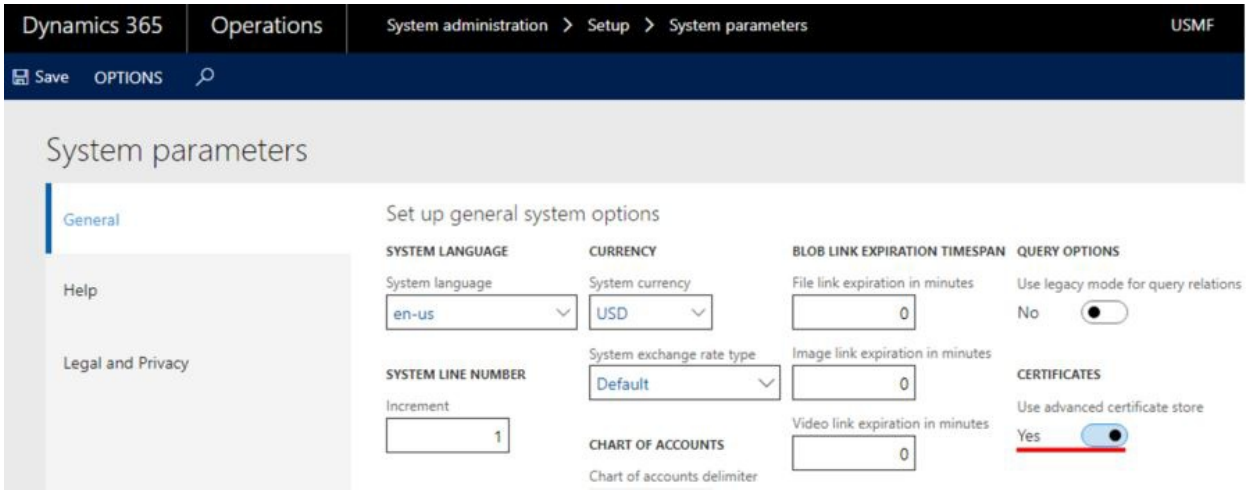
Set up the Azure Key Vault client

2/18/2021 • 3 minutes to read • Edit Online

The functionality for storing advanced certificates lets you define the type of certificate storage that is used in Finance and Operations apps.

The functionality provides two options for storing certificates: local storage and Microsoft Azure Key Vault storage. You can define the option that is used by setting the new **Use advanced certificate store** option on the **General** tab of the **System parameters** page (**System administration** > **Setup** > **System parameters**).

- **Local storage** – This storage option can be used with on-premises deployments and any kind of on-premises development environment. To use it, set the **Use advanced certificate store** option to **No**. This storage option is recommended for development environments that are used for development and validation purposes, where it's necessary to validate the certificate and work with it.
- **Azure Key Vault storage** – This storage option is required for cloud deployments, but it can also be used with on-premises deployed environments and any kind of on-premises development environment. To use it, set the **Use advanced certificate store** option to **Yes**. This storage option is the only option for a production environment in the Azure cloud.



Some setup is required before you can work with certificates that are stored in Key Vault. For information about the required settings, see the following Microsoft Knowledge Base (KB) article: 4040294 - Maintaining Azure Key Vault storage. After you set up the Key Vault storage, you should link to the certificates in Finance and Operations apps.

After the certificate is installed in Key Vault, it must be set up in the application.

1. Go to **System administration** > **Setup** > **Key Vault parameters**.
2. Select **New** to create a new instance.
3. Enter a name and description, and then, on the **General** FastTab, set the fields that are required for the integration with Key Vault storage:

- **Key Vault URL** – Enter the default Key Vault URL if it isn't already defined by the secret reference.
- **Key Vault client** – Enter the interactive client ID of the Azure Active Directory (Azure AD) application that is associated with the Key Vault storage for authentication.
- **Key Vault secret key** – Enter the secret key that is associated with the Azure AD application that is used

for authentication with the Key Vault storage.

> **NOTE**
> If several Key Vault storages are used, you should set up a separate instance for each instance on the **Key Vault parameters** page.

4. On the **Certificates** FastTab, select **Add** to add your certificates. For each certificate, set the following fields:

- **Name**
- **Description**
- **Key Vault certificate secret** – Enter a secret reference to the certificate.

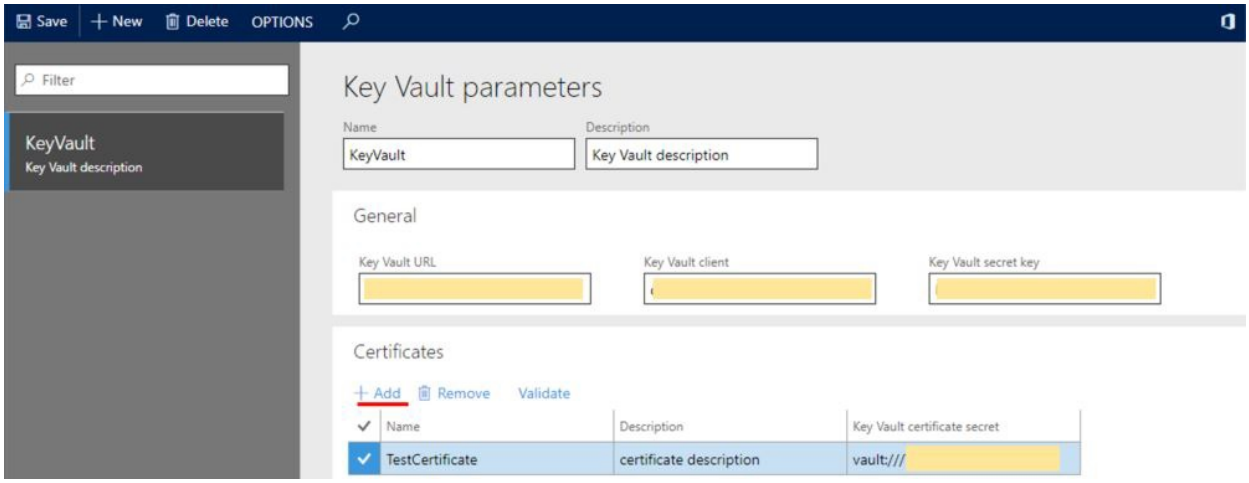
The format of a Key Vault certificate secret must resemble the following example:

vault://<KeyVaultName*>/<SecretName>/<SecretVersion*>

Attributes that are marked with an asterisk (*) are optional. However, the **<SecretName>** attribute is required. In most cases, you can define a Key Vault secret key in the following format:

vault:///<SecretName>

If the secret version isn't defined in the Key Vault secret key, the system retrieves the active certificate that has the latest expiration date.



> **NOTE**
> The Key Vault storage functionality has been extended so that it includes caching of certificates. The following configuration is recommended:

- Specify a secret version in the Key Vault certificate secret.
- After you upload a new version of the existing certificate to the Key Vault storage, update the **<SecretVersion>** attribute in the **Key Vault certificate secret** field.

Use the **Validate** function to verify that you've correctly defined the reference to the certificate, and that the certificate is valid.

> **NOTE**
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Cleanup routines in Dynamics 365 Finance and Dynamics 365 Supply Chain Management

2/18/2021 • 10 minutes to read • Edit Online

In Microsoft Dynamics 365 Finance and Dynamics 365 Supply Chain Management, cleanup routines are available in various modules. This topic provides an overview of the routines that are currently available. The information is organized by module.

> **IMPORTANT**
>
> These cleanup routines should be run only after the business has done detailed analysis and confirmed that the data is no longer required.
>
> Always test each cleanup routine in a test environment before you run it in a production environment.

System administration

PATH	DESCRIPTION
System administration > Periodic tasks > Notification clean up	This cleanup routine is used to periodically delete records from the EventInbox and EventInboxData tables. **Recommendation:** If you don't use alert functionality, turn off the alert from the batch job.
System administration > Periodic tasks > Batch job history clean-up	This regular version of the batch job history cleanup routine lets you quickly clean all history entries that are older than a specified number of days. Any entry that was created earlier will be deleted from the BatchJobHistory table, and also from linked tables that have related records (BatchHistory and BatchConstraintsHistory). This version has improved performance optimization, because it doesn't have to run any filtering.
System administration > Periodic tasks > Batch job history clean-up (custom)	This custom batch job history cleanup routine should be used only when specific entries must be deleted. You can clean up selected types of batch job history records, based on criteria such as status, job description, company, or user. You can add other criteria by using the **Filter** button.
System administration > Inquiries > Database > Database Log > Clean up log	This cleanup routine lets you delete database logs as you require. You can delete logs for specific tables, delete specific types of database logs, or delete logs based on the date and time when they were created. **Note:** Records that have been electronically signed can't be deleted from logs.

Data management

PATH	DESCRIPTION
In the **Data management** workspace, select **Job history cleanup**.	This cleanup routine is available in Platform update 29 and later. To use it, you must turn on the **Execution history cleanup** feature in Feature management. In Data management, this routine must be used to schedule a periodic cleanup of the execution history. It replaces the earlier Staging cleanup routine, which is now obsolete (deprecated). The following tables will be cleaned up: • All staging tables • DMFSTAGINGVALIDATIONLOG • DMFSTAGINGEXECUTIONERRORS • DMFSTAGINGLOGDETAIL • DMFSTAGINGLOG • DMFDEFINITIONGROUPEXECUTIONHISTORY • DMFEXECUTION • DMFDEFINITIONGROUPEXECUTION
In the **Data management** workspace, select the **Staging cleanup** tile.	This cleanup routine should no longer be used, because it's obsolete. Instead, use the Job history cleanup routine.

General ledger

PATH	DESCRIPTION
General ledger > Periodic tasks > Clean up ledger journals	This cleanup routine deletes General ledger, Accounts receivable, and Accounts payable journals that have been posted. When you delete a posted ledger journal, all information that is related to the original transaction is removed. **Note:** You should delete this information only if you're sure that you won't have to reverse the ledger journal transactions.

Sales and marketing

PATH	DESCRIPTION
Sales and marketing > Periodic tasks > Clean up > Delete sales orders	This cleanup routine deletes selected sales orders.
Sales and marketing > Periodic tasks > Clean up > Delete quotations	This cleanup routine deletes selected quotations.
Sales and marketing > Periodic tasks > Clean up > Delete return orders	This cleanup routine deletes selected return orders.
Sales and marketing > Periodic tasks > Clean up > Sales update history cleanup	This cleanup routine deletes old update history transactions. All updates of confirmations, picking lists, packing slips, and invoices generate update history transactions. You can view these transactions on the **History on update** page.

PATH	DESCRIPTION
Sales and marketing > Periodic tasks > Clean up > Order events cleanup	This cleanup routine cleans up order events. The next step is to open the **Order event setup** page and clear the check boxes for any order events that aren't required.

Procurement and sourcing

PATH	DESCRIPTION
Procurement and sourcing > Periodic tasks > Clean up > Purchase update history cleanup	This cleanup routine is used to delete all updates of confirmations, picking lists, product receipts, and invoices that generate update history transactions.
Procurement and sourcing > Periodic tasks > Clean up > Delete requests for quotations	This cleanup routine is used to delete requests for quotation (RFQs) and RFQ replies. The corresponding RFQ journals aren't deleted but remain in the system.
Procurement and sourcing > Periodic tasks > Clean up > Draft consignment replenishment order journal cleanup	This cleanup routine is used to clean up draft consignment replenishment order journals.

Warehouse management

PATH	DESCRIPTION
Warehouse management > Periodic tasks > Clean up > Work creation history purge	This cleanup routine is used to delete work creation history records from the WHSWorkCreateHistorytable table. In the dialog box, you specify the number of days to keep the history.
Warehouse management > Periodic tasks > Clean up > Containerization history purge	This cleanup routine is used to delete containerization history from the WHSContainerizationHistory table. In the dialog box, you specify the number of days to keep the history.
Warehouse management > Periodic tasks > Clean up > Wave batch cleanup	This cleanup routine is used to clean up batch job history records that are related to the wave processing batch group.
Warehouse management > Periodic tasks > Clean up > Cycle count plan cleanup	This cleanup routine is used to clean up batch job history records that are related to cycle count plan configurations.
Warehouse management > Periodic tasks > Clean up > Mobile device activity log cleanup	This cleanup routine is used to delete mobile device activity log records from the WHSMobileDeviceActivityLog table. In the dialog box, you specify the number of days to keep the history.
Warehouse management > Periodic tasks > Clean up > Work user session log cleanup	This cleanup routine is used to delete work user session records from the WHSWorkUserSessionLog table. In the dialog box, you specify the number of hours to keep records.

Inventory management

PATH	DESCRIPTION
Inventory management > Periodic tasks > Clean up > Calculation of location load	The WMSLocationLoad table is used to track the weight and volume of items and pallets. The Summation of load adjustments job can be run to reduce the number of records in the WMSLocationLoad table and help improve performance.
Inventory management > Periodic tasks > Clean up > Inventory journals cleanup	This cleanup routine is used to delete posted inventory journals.
Inventory management > Periodic tasks > Clean up > Inventory settlements cleanup	This cleanup routine is used to group closed inventory transactions or delete canceled inventory settlements. By cleaning up closed or deleted inventory settlements, you can help free up system resources. Don't group or delete inventory settlements that are too close to the current date or fiscal year, because part of the transaction information for the settlements will be lost. Closed inventory transactions can't be changed after they have been grouped, because the transaction information for the settlements will be lost. If canceled inventory settlements are deleted, they can't be reconciled with finance transactions.
Inventory management > Periodic tasks > Clean up > Inventory dimensions cleanup	This cleanup routine is used to maintain the InventDim table. To maintain the table, delete unused inventory dimension combination records that aren't referenced by any transaction or master data. The records are deleted, regardless of whether the transaction is open or closed. An inventory dimension combination record that is still referenced can't be deleted, because when an InventDim record is deleted, related transactions can't be reopened.
Inventory management > Periodic tasks > Clean up > Dimension inconsistency cleanup	This cleanup routine is used to resolve dimension inconsistencies on inventory transactions that have been financially updated and closed. Inconsistencies might be introduced if the multisite functionality was activated during or before the upgrade process. Use this routine only to clean up the transactions that were closed before the multisite functionality was activated. **Note:** Don't use this routine periodically.
Inventory management > Periodic tasks > Clean up > On-hand entries cleanup	This cleanup routine is used to delete closed and unused entries for on-hand inventory that is assigned to one or more tracking dimensions. Closed transactions contain a value of **0** (zero) for all quantities and cost values, and they are marked as closed. By deleting these transactions, you can help improve the performance of queries for on-hand inventory. Transactions won't be deleted for on-hand inventory that isn't assigned to tracking dimensions.

PATH	DESCRIPTION
Inventory management > Periodic tasks > Clean up > Warehouse management on-hand entries cleanup	This cleanup routine deletes records in the InventSum and WHSInventReserve tables. These tables are used to store on-hand information for items that are enabled for warehouse management processing (that is, WHS items). By cleaning up these records, you can significantly improve of the on-hand calculations.
Inventory management > Periodic tasks > Clean up > On-hand entries aggregation by financial dimensions	Use This cleanup routine as a tool to aggregate InventSum rows that have 0 (zero) quantities. This routine basically extends the previously mentioned routine by also cleaning up records where the **Closed** field is set to **True**. Basically, this routine is needed to handle scenarios where there are no more quantities in the InventSum table for a combination of inventory dimensions, but there is still a value. Although these values will disappear in some cases, the current design occasionally allows values to remain. For example, if you use batch numbers, each batch number (and the combined site, warehouse, and so on) creates a new record in the InventSum table. When the batch number is sold, you will see that quantity fields are set to **0** (zero). In most cases, the **Financial cost amount** and **Physical cost amount** fields are also set to **0** (zero). However, in standard cost revaluation and other scenarios, the field might still show some amount. This behavior is valid, and it reflects the way that Finance and Supply Chain Management handle the costs at the financial inventory level (for example, the site level). In Finance and Supply Chain Management, inventory value is determined by records in the InventSum table. In some cases, when inventory values in the past are reported, it's determined by inventory transactions (the InventTrans table). Therefore, in the previously described scenario, when you run inventory value reports, Finance and Supply Chain Management initially look at the InventSum table, aggregate all records to the site level, and report the value for the item per site. The data from the individual records at batch number level are never used. Therefore, this routine goes through all InventSum records, finds the records where there is no more quantity (that is, **No open quantities** field is set to **True**). Because there is no reason to keep these records, Finance and Supply Chain Management find the InventSum record for the same item that has the same site, they copy the values from the batch number level to the site level, and they delete the record. Then, when you run inventory value reports, Finance and Supply Chain Management still find the same correct values. Therefore, this routine reduces number of InventSum records, significantly in some cases, and can have a positive impact on the performance of any function that queries that table.
Inventory management > Periodic tasks > Clean up > Cost calculation details	This cleanup routine is used to clean up cost calculation details.

Production control

PATH	DESCRIPTION
Production control > Periodic tasks > Clean up > Production journals cleanup	This cleanup routine is used to delete unused journals.
Production control > Periodic tasks > Clean up > Production orders cleanup	This cleanup routine is used to delete production orders that are ended.
Production control > Periodic tasks > Clean up > Clean up registrations	We recommend that you periodically clean up registrations. This cleanup routine deletes only data that has been processed. **Note:** Make sure that you don't delete registrations that might be required later for documentation purposes.
Production control > Periodic tasks > Clean up > Archive future registrations	This cleanup routine is used to remove future registrations from the raw registrations table.

Master planning

PATH	DESCRIPTION
Master planning > Master planning > Maintain plans > Plan version cleanup	Usually, this cleanup is done automatically. However, automatic cleanup sometimes malfunctions, and orphan data remains in the system. This orphan data slows down queries and causes the database size to grow. We recommend that you do a preventive run one time per month, when master resource planning (MRP) isn't running.

Upgrades, updates, and hotfixes resources

2/18/2021 • 2 minutes to read • Edit Online

Upgrades, updates, and hotfixes can include moving to new product versions, code migration and upgrade, moving to an update, or deploying a hotfix.

The processes for each type of upgrade are similar, but different enough that we think that you should review the topics for a specific task before you begin.

Upgrade from Microsoft Dynamics AX 2012 to Finance and Operations

To get started, review the following topics:

- Upgrade from AX 2012 to Finance and Operations
- Prepare to migrate code to Finance and Operations

Migration from Microsoft Dynamics AX 2009 to Finance and Operations

This Tech Talk video provides an introduction to migration from AX 2009 to Finance and Operations: Dynamics 365 for Operations – Tech Talk: Migration tools.

Upgrade from a previous version of Finance and Operations

The steps for applying updates and upgrading differ between cloud and on-premises implementations.

Cloud

If you are upgrading a cloud version of Finance and Operations, review the following topics:

- Process for moving to the latest update of Finance and Operations
- Apply the latest platform update to environments
- Download updates from Lifecycle Services (LCS)

On-premises

If you are applying updates to an on-premises version of Finance and Operations, review the following topic:

- Apply updates to on-premises deployments
- Redeploy on-premises environments

Hotfixes

- Download updates from Lifecycle Services (LCS)
- Apply updates to cloud environments
- Install metadata hotfixes in development environments
- Patch SQL Server Reporting Services (SSRS) in one-box environments
- Update the Visual Studio development tools

This Tech Talk video provides an introduction servicing (applying code updates, requesting sandbox database

refreshes, and filing support requests): Dynamics 365 for Operations – Tech Talk: Servicing.

For more information, see:

- Refresh database
- Set up technical support for Finance and Operations apps

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Upgrade from AX 2012 to Finance and Operations

2/18/2021 • 13 minutes to read • Edit Online

IMPORTANT

Upgrade is currently only supported from either Dynamics AX 2012 R2 or Dynamics AX 2012 R3. For each release, please update to the latest available cumulative update before upgrading to latest Finance and Operations application release.

Finance and Operations running Platform update 8 and the July 2017 Application release, and later provides an upgrade path that customers who currently run Microsoft Dynamics AX 2012 can use to move their data and code to Finance and Operations. Currently upgrade from Dynamics AX 2012 R3 and AX 2012 R2 are supported. The upgrade process is built on the following elements:

- Tools to help you bring forward existing custom application code from AX 2012.
- A data upgrade process that you can use to bring your database forward. Therefore, you can upgrade your full transactional history.

IMPORTANT

Dynamics AX 2012 implementations that are running some deprecated features cannot currently be upgraded. For example, upgrade is not possible from systems that are using either virtual companies or data partitions. If you aren't sure whether your system can be upgraded, run the Upgrade analyzer tool.

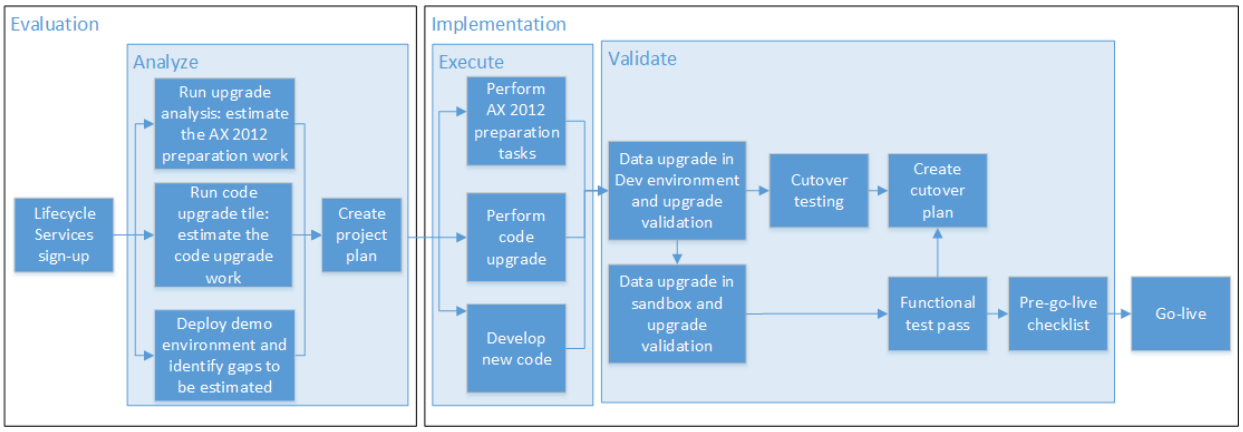
NOTE

Start your cloud migration journey with a no-charge, no-obligation migration assessment though the Dynamics 365 Migration Program.

Overview

The overall upgrade process can be visualized as three overarching phases: Analyze, Execute, and Validate.

The following diagram shows the end-to-end upgrade process, and the activities that we consider part of each phase.



Analyze

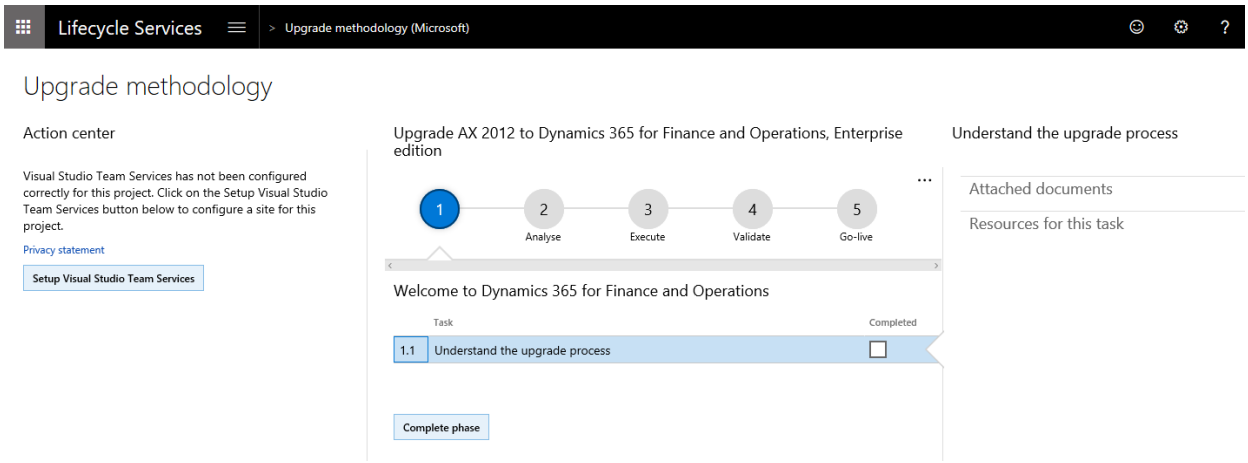
The activities in the Analyze phase help you estimate the effort that is required for the upgrade. They also help you prepare a project plan. These activities can be done before you buy Finance and Operations. They will help you make an informed purchase decision by providing a data point about the effort and resources that you will require.

Sign up for a preview subscription

To sign up for a preview subscription, see Sign up for preview subscriptions.

Select the upgrade methodology

In your new LCS project, set the project methodology to **Upgrade AX 2012 to Dynamics 365 for Finance and Operations**. This methodology is made specially for AX 2012 customers who are upgrading. It describes the three phases in detail and provides links to all the supporting documentation about the process.



Run the upgrade analyzer

The upgrade analyzer tool runs against your AX 2012 environment and identifies tasks that you should do to prepare the AX 2012 environment, to help make the upgrade experience smoother and less expensive:

- **Data cleanup** – This process helps you identify data that you can remove without causing loss of functionality. The tool identifies various types of data that you can reduce by running a cleanup process. For each type of data, an explanation is given about the impact of the cleanup. You then decide whether to run the cleanup process. Part of the cost of your subscription is based on database size. Therefore, by reducing the size, you reduce that component of the subscription cost and also help reduce the time that is required for the upgrade go-live process. A smaller database helps guarantee a faster upgrade.
- **SQL configuration** – This process reviews the SQL configuration and recommends optimizations. By making sure that SQL performs optimally, this process helps reduce the time that is required for the upgrade go-live process.
- **Deprecated features** – This process identifies features that you're currently using, but that aren't available in Finance and Operations. Therefore, the process helps you discover gaps in functionality early. It also provides suggestions for alternatives.

Additionally, as part of this step, you must install a pre-upgrade checklist in your AX 2012 environment. You can use this checklist to enter data that will be required for the upgrade procedure. For example, in one pre-upgrade checklist task, you provide the Microsoft Azure Active Directory (Azure AD) sign-in information for each current AX 2012 user, so that each user will be able to sign in to Finance and Operations.

- If upgrading from AX 2012 R3, install KB 4035163
- if upgrading from AX 2012 R2, install KB 4048614

The output of the upgrade analyzer tool becomes the workstream in the upgrade project plan for your AX 2012 system administrators. For more information, see Upgrade from AX 2012 - Plan by using the Upgrade analyzer tool.

Run the Code upgrade estimation tools

This step takes your code from AX 2012, converts it to the new format, and provides feedback about conflicts that a developer must resolve later. This step forms the basis for the estimate of the cost of your code upgrade.

To complete this step, you must export your code from AX 2012 as a model store export and upload it to the LCS Code upgrade tool. The Code upgrade tool will produce an upgraded version of your code and a report about the remaining conflicts that must be resolved. Your developer can then review both the upgraded code and the report to determine the effort that will be required in order to upgrade your code base.

The output of this step represents the workstream in the upgrade project plan for your Microsoft Dynamics AX developers.

For more information, see Upgrade from AX 2012 - Estimate effort by using the Code upgrade service.

Deploy a demo environment

Demo environments are default environments that contain demonstration data (not your own data) and standard code (no customizations). We recommend that you deploy a demo environment to evaluate new features, and to perform a basic fit gap analysis of standard processes that are used in AX 2012 but that might have changed in Finance and Operations. You can either deploy these demo environments in Azure or downloaded them as a virtual machine (VM) that you run on your own hardware. If you deploy them in Azure, you must provide your Azure subscription, because you're still using a public preview project and haven't yet purchased a subscription.

The output of this step represents the workstream in the upgrade project plan for your functional users or business users.

For more information, see Upgrade from AX 2012 - Deploy a demo environment for analysis

Create a project plan

A template for a project plan is provided in the upgrade methodology. In this step, the output from the previous steps of the Analyze phase is used to fill the project plan for the upgrade project. The project plan will also contain all testing details: data upgrade testing, cutover testing (mock cutover), the functional test pass iterations, and details about the various resource assignments for those tasks.

At this stage, the project plan provides a data point that can help you understand the time and cost of an upgrade.

Execute

During the Execute phase, you work through the tasks that you planned during the Analyze phase. To move to the Execute phase, you must purchase Finance and Operations apps, and you must have available resources that can work on the upgrade.

Switch to the LCS implementation project

The public preview project that you used for the Analyze phase has served its purpose. You can now discard it. For the remaining steps, you require only the project plan that you created in the final step of the Analyze phase.

When you purchase a subscription, you will receive details about how to sign up for a new LCS project. This project is known as an implementation project and will be the new permanent LCS project for your subscription, for as long as you have that subscription. This project differs from the public preview project in that it's managed by Microsoft. Therefore, this project has these characteristics:

- All environments in the project are hosted in Azure.
- The Azure subscription that is associated with the project is managed by Microsoft. Therefore, there is no separate billing for Azure costs. The costs are covered by your subscription.
- The production environment in the project is maintained by Microsoft. Therefore, code deployments,

upgrades, and infrastructure maintenance are run directly by Microsoft, not by your staff.

Identify the project as an AX 2012 upgrade

When you first sign in to your LCS implementation project, you're guided through the **Project Onboarding** wizard. You can always visit the **Project Onboarding** wizard later using the navigation menu next to **Project Settings** in your project.

While on the Project Onboarding wizard, in the **Project Scope** section, you can use the **Legacy System** field to identify the project as an AX 2012 upgrade. It's crucial that you identify the project in this way, so that the sandbox infrastructure that is deployed is compatible with the upgrade process that is outlined here. If this step isn't completed early in the project, you might accidentally deploy your Sandbox on a newer infrastructure that is incompatible with this process. In that case, the upgrade effort might be delayed.

Perform the AX 2012 preparation tasks

Complete the tasks that the upgrade analyzer tool discovered, and that are documented in your upgrade project plan. Your Microsoft Dynamics AX system administrator and database administrator (DBA) must complete these tasks.

Upgrade from AX 2012 - Pre-upgrade checklist for data upgrade

Perform code upgrade

Complete the tasks that were planned during the code upgrade estimation step of the Analyze phase. Your developers must run these tasks.

From this point onward, code changes in AX 2012 should be frozen. Only emergency code changes should be allowed in AX 2012. If a change is made, it must be ported manually to the new code base.

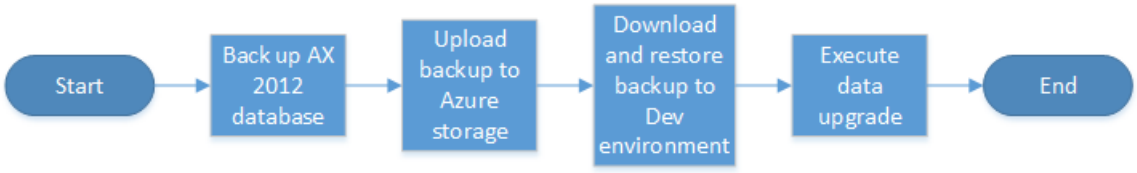
Develop new code

Complete the tasks from the fit gap analysis that was performed during the "Deploy a demo environment" step of the Analyze phase. These tasks will probably be a mixture of functional tasks that define the configuration and development tasks for customizations that are related to new features that are being taken up.

Data upgrade (development environment)

After your code upgrade tasks are completed, you can upgrade your database for the first time. This first upgrade occurs in a development environment, so that you can more easily remediate or debug any issues that are found at this stage. In a development environment, an issue can be debugged immediately, code can be adjusted, and the upgrade can be rerun within minutes. Larger sandbox environments don't offer this agility, and a minimum of several hours will be required in order to debug and remediate issues, update code, deploy the updated code, and rerun the upgrade.

The following illustration shows the process. Just back up the AX 2012 database, upload it to Azure, restore it to the Finance and Operations environment, and then run the data upgrade.



Data upgrade is done through a special type of deployable package. The same mechanism is used to deploy new code from one environment to another environment.

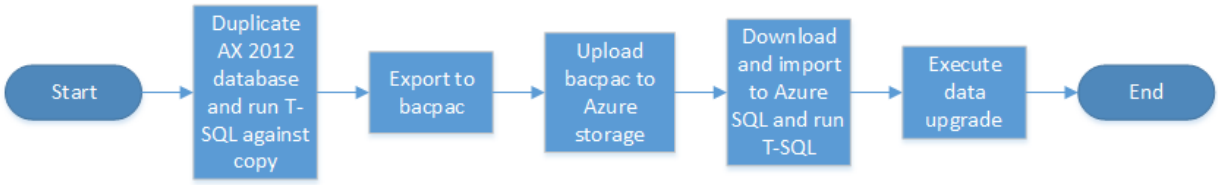
The underlying framework that is used to convert the data in the database during this process is largely the same as the upgrade framework in AX 2012 that is based on X++ batch jobs that run **ReleaseUpdatexxx** classes.

For details, see Upgrade from AX 2012 - Data upgrade in development environments.

Data upgrade (sandbox environments)

When data upgrade in a development environment is completed, the same process can be run in a sandbox environment. The sandbox environment is the environment where business users and functional team members can test business processes by using the upgraded AX 2012 data and code.

The following illustration shows the process for running data upgrade in a sandbox environment. The difference here is that the bacpac tool is used instead of a traditional SQL backup. This tool is required in order to convert between Microsoft SQL Server and Azure SQL Database. It's a standard SQL tool, and isn't specific to Finance and Operations.



For details, see Upgrade from AX 2012 - Data upgrade in sandbox environments.

Validate

When you enter the Validate phase, you will have available environments that include your upgraded custom code and your upgraded data. This phase describes the process of validating and testing that the upgraded environment works as desired. It also describes the process of preparing for go-live.

Perform cutover testing and create a cutover plan

The term *cutover* is used here to describe the final process of putting the new system live. This process consists of the tasks that occur after AX 2012 is turned off and before Finance and Operations is turned on.

The goal of the testing, or *mock cutover* is to practice the cutover process. In this way, you can help guarantee that everyone who is involved in the actual cutover to go-live will have a smooth experience.

There are two main workstreams:

- **Technical workstream** – This workstream is the process of running the data upgrade. Your business will enforce a limit on the amount of downtime that is allowed. During this downtime, neither product database will be available. The technical workstream might have to performance-tune its data upgrade procedure to meet the business's downtime limit.
- **Functional workstream** – After data upgrade, several configuration tasks will be required in the Finance and Operations environment. All these tasks must be documented and quantified, and a resource must be assigned to them, because they must fit together with the technical tasks within the business's downtime limit.

For details, see

- Upgrade from AX 2012 - Post-upgrade tasks
- Upgrade from AX 2012 - Cutover testing (Mock cutover)

Functional test pass

Complete a full functional test pass of all business processes. This test pass will be an extensive retest of all business processes that involve Finance and Operations. These business processes include both old processes that were brought forward from AX 2012 and new processes that involve new features that were taken up for the first time in Finance and Operations.

Depending on code quality, issue remediation and retesting might require several iterations of the functional test pass. When an issue is fixed, be sure to retest all processes that are involved, to help guarantee that the downstream or upstream process isn't affected by the change.

For details, see Upgrade from AX 2012 - Functional test passes.

Pre-go-live checklist

The pre-go-live checklist is a recommended procedure that can help reduce the chance of errors during the final cutover to go-live. One week before go-live is due, stop configuration changes in AX 2012 (that is, under <module>\Setup). This restriction on configuration changes is merely procedural. The Microsoft Dynamics AX system administrators just agree to put changes of this type on hold at this point.

We recommend that you also freeze code changes in the Finance and Operations code base. No further changes should be allowed unless they have been evaluated and have been shown not to block go-live.

After the configuration restriction and code freeze are in place, data upgrade should be run for the last time before cutover. In this way, you can make sure that everything still works as expected.

For details, see Validate: Prepare for go live.

Go live

After you have successfully completed upgrade testing in a Standard or Premier Acceptance Test environment (Sandbox Tier 2 or higher), and you have also completed a successful test cutover, the moment has arrived to upgrade your production environment and go live.

Cutover is the term that we use for the final process of getting a new system live. This cutover process consists of the tasks that occur after Dynamics AX 2012 is turned off but before Finance and Operations is turned on.

For details, see Upgrade from AX 2012 - Cutover process (Go live)

Supported upgrade paths

Upgrade to the cloud version of Finance and Operations is supported from AX 2012 R2 and AX 2012 R3, in private preview.

Upgrade from Dynamics AX 2012 RTM isn't currently supported. Upgrade to the on-premises version isn't currently supported, but support will be added in the future.

Upgrade from AX 2012 - Plan by using the Upgrade analyzer tool

2/18/2021 • 2 minutes to read • Edit Online

IMPORTANT

Upgrade is currently only supported from either Dynamics AX 2012 R2 or Dynamics AX 2012 R3. For each release, please update to the latest available cumulative update before upgrading to latest Finance and Operations application release.

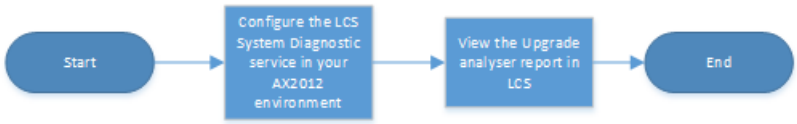
This topic explains how to use the Upgrade analyzer tool to plan your upgrade from Microsoft Dynamics AX 2012. This tool is run against an AX 2012 environment and identifies data that you should clean up in AX 2012 to help reduce the subscription cost for Finance and Operations. The tool also suggests SQL configuration optimizations that can help speed up the upgrade processes. Additionally, the tool warns you if any features that you use in AX 2012 are obsolete in the current version. Therefore, you can plan ways to replace or work around those features.

Upgrade analyzer gathers data from your AX 2012 environment as part of the regular System diagnostic service in Dynamics Lifecycle Services (LCS). For an overview of the System diagnostic service, and for information about how data is collected and pushed back into the cloud so that you can consume it through LCS, see System diagnostics in Lifecycle Services (LCS).

You can view the results of the System diagnostic service in a Microsoft Power BI report in LCS. The report presents a list of tasks that you should complete in the AX 2012 environment.

To access the Upgrade analyzer report, go to https://diag.lcs.dynamics.com/UpgradeAnalysisReport/Report/"ProjectID" (Replace "ProjectID" with your current project ID, which is an integer that can be found in the URL of your current LCS project).

The following illustration shows an overview of the procedure for using Upgrade analyzer.



If you already use the System diagnostic service in your AX 2012 environment, you must configure a new instance of the service on a machine that differs from the existing machine.

For information about how to configure the System diagnostic service in your AX 2012 environment, see Install and run System diagnostics.

Within a few minutes after you configure the System diagnostic service, the AX 2012 environment will appear in your LCS project.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Upgrade from AX 2012 - Estimate effort by using the Code upgrade service

2/18/2021 • 2 minutes to read • Edit Online

IMPORTANT

Upgrade is currently only supported from either Dynamics AX 2012 R2 or Dynamics AX 2012 R3. For each release, please update to the latest available cumulative update before upgrading to latest Finance and Operations application release.

This topic explains how to use the Code upgrade service in Microsoft Dynamics Lifecycle Services (LCS) to help estimate the tasks and effort that are required in order to upgrade a code base from Microsoft Dynamics AX 2012 Finance and Operations.

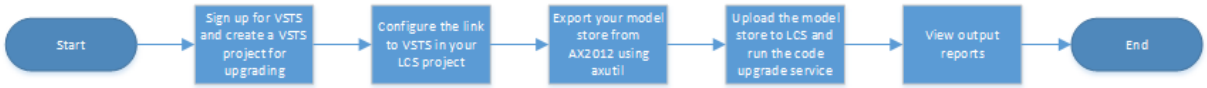
The Code upgrade service converts an export of your AX 2012 model store to the correct format. However, the new version of your code won't be fully functional until a developer resolves any issues that the service identifies but can't resolve itself.

The Code upgrade service performs these actions:

- Directly resolve some types of conflict issues.
- For other issues, log Microsoft Azure DevOps tasks.
- Create a version of your code in the correct format, and check the new version into a new branch of your Azure DevOps project.

In the Analyze phase, we use the report to help estimate the effort that is required in order to complete code conversion activities.

The following illustration shows an overview of the process for configuring the Code upgrade service.



For information about how to configure the Code upgrade service, see Configure the code upgrade service in Lifecycle Services (LCS).

The output of the Code upgrade service is designed to be consumed by a developer. This output will help the developer estimate the effort that is required in order to complete the code upgrade tasks. To form an estimate, the developer must review the tasks that the service generates in Azure DevOps and the new version of the code that the service generates.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Upgrade from AX 2012 - Deploy a demo environment for analysis

2/18/2021 • 2 minutes to read • Edit Online

IMPORTANT

Upgrade is currently only supported from either Dynamics AX 2012 R2 or Dynamics AX 2012 R3. For each release, please update to the latest available cumulative update before upgrading to latest Finance and Operations application release.

This topic explains why and how you should deploy a demo environment during the Analyze phase of your project for upgrading from Microsoft Dynamics AX 2012 to Finance and Operations.

By deploying a demo Finance and Operations environment, you gain hands-on experience with the program and can explore new features that you might be interested in. You also have an opportunity to validate differences in the program for your business processes, so that you can identify potential gaps or confirm that no gaps exist. At this stage in your upgrade project, your data and code won't be available in the environment. Therefore, you will have limited ability to validate that everything conforms to your business process. However, this step is the first step in that work stream.

If you haven't yet purchased licenses, and you're using a free trial, you can follow the steps in Deploy a demo environment to deploy a demo environment to a Microsoft Azure subscription that you bring yourself.

If you've already purchased licenses, you received a link to configure a special type of project in Microsoft Dynamics Lifecycle Services (LCS): an implementation project. The implementation project will let you deploy a dev/test environment and a sandbox environment. For more information about this type of environment deployment, see Upgrade from AX 2012 - Data upgrade in sandbox environments.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Upgrade from AX 2012 - Data upgrade in development environments

2/18/2021 • 5 minutes to read • Edit Online

IMPORTANT

Upgrade is currently only supported from either Dynamics AX 2012 R2 or Dynamics AX 2012 R3. For each release, please update to the latest available cumulative update before upgrading to latest Finance and Operations application release.

This is an exciting moment in the upgrade project. The output of this task provides the first upgraded dataset from Microsoft Dynamics AX 2012 to the latest Finance and Operations development environment.

Before you run this process in a shared sandbox environment, we recommend that you run it in a development environment. There are two reasons for this approach:

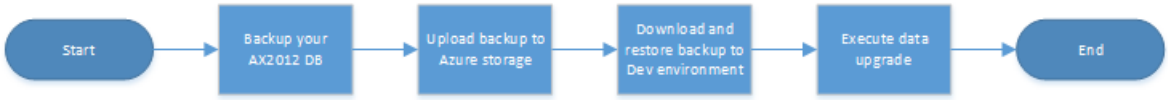
- It provides local data that developers can write and test their custom data upgrade scripts against.
- It helps reduce the overall time that is spent on iterations of the data upgrade process. In a development environment, an issue can be debugged immediately, code can be adjusted, and the upgrade can be rerun within minutes. However, larger sandbox environments don't allow for this level of agility. In those environments, a minimum of several hours will be required to debug and remediate issues, update code, deploy the updated code, and rerun the upgrade.

We strongly recommend that you run the Upgrade analyzer and respond to the issues it identifies before running data upgrade - this will help ensure that your data upgrade is quicker and easier.

NOTE

It's very important that you install the latest version of SQL Server Management Studio before you start this process: Download SQL Server Management Studio.

End-to-end data upgrade process



Back up your AX 2012 database

To back up your AX 2012 database, use the standard Microsoft SQL Server process to produce a BAK file. If you use the compression option when you create the backup, the file size will be smaller, and less time is required in order upload it to and download it from Microsoft Azure Storage.

Upload the backup to Azure Storage

If your developer environment is hosted as a VM locally or in Azure you will need to transfer the 2012 database backup to it. With a local VM you may be able to transfer the file directly across the network (if you have configured the virtual network to allow that) but for an Azure hosted VM we recommend that you upload your backup to Azure Storage (using your own secure file transfer service or SFTP is also a valid option). You would need to provide your own Azure storage account for this. There are free tools to help you to move files between Azure storage, from a command line you can use Azcopy, or for a GUI experience you can use Microsoft Azure storage explorer. Use one of these tools to first upload the backup from your on-premises environment to Azure

storage and then on your download it on your development environment.

Download and restore the backup to the development environment

> **NOTE**
>
> Developer environments that are hosted by Microsoft have limited drive space. We recommend that most AX 2012 customers host their own developer environment by using cloud-hosted environments. By using cloud-hosted environments, you can increase the drive space so that it meets your own specifications.

When you restore the backup to the new development environment, don't overwrite the existing AXDB database. Instead, restore the AX 2012 database next to the original databases. You might also consider using drive D for the data and log files, to help improve performance. However, there is a potential downside to using drive D. If the underlying virtual machine (VM) is deallocated in Azure and then reallocated, drive D will be wiped. In practice, this scenario rarely occurs. Therefore, you might find that the risk is acceptable. To learn more about how to use drive D, see Understanding the temporary drive on Windows Azure Virtual Machines.

To speed up the database restore process, you can change the SQL Server service account to **axlocaladmin**. The restore process can then use instant file initialization. For more information, see Database Instant File Initialization.

After the database is restored, stop the following services:

- World wide web publishing service
- Dynamics 365 for Finance and Operations Batch Management service
- Management Reporter 2012 Process service
- Microsoft Dynamics Lifecycle Services Diagnostic Service
- Data Import / Export service

Next, rename the original AXDB database **AXDB_orig**. This database might be useful as reference later, when you develop code.

```
ALTER DATABASE AXDB SET SINGLE_USER WITH ROLLBACK IMMEDIATE
GO
ALTER DATABASE AXDB MODIFY NAME = AXDB_Orig
GO
ALTER DATABASE AXDB_Orig SET MULTI_USER
GO
```

Finally, rename the newly restored AX 2012 database **AXDB**.

Run the data upgrade deployable package

To get the latest data upgrade deployable package for a target environment that is running the latest update, download the latest binary updates from Microsoft Dynamics Lifecycle Services (LCS) Shared asset library.

1. Sign in to LCS.
2. Select the **Shared asset library** tile.
3. In the **Shared asset** library, under **Select asset type**, select **Software deployable package**.
4. In the list of deployable package files, find the data upgrade package that corresponds to your upgrade. For example, if you're upgrading from AX 2012, the package name starts with AX2012DataUpgrade. Select the package that corresponds to the release you are upgrading to. For example: AX2012DataUpgrade-July2017.

For more information, see Upgrade data in development or demo environments.

Troubleshooting data upgrade script errors

There are options that you let you resume the data upgrade where it last stopped. You can also record any data upgrade script errors with call stacks to a table in the database. For development scenarios, you can skip failed scripts and continue to run the upgrade.

For more details, see the main data upgrade topic.

Recommendation for the first data upgrade run

When you run the data upgrade against your dataset for the first time, and especially when there many customizations or many custom data upgrade scripts, you might find the feature to skip failed scripts useful. By using this feature, you gain visibility into as many errors as possible in one run. Otherwise, only one critical issue is discovered per run. Be aware that, because dependencies exist between scripts, you might receive errors in related child scripts if you skip the parent script. These errors occur only because the parent wasn't run correctly. They will be resolved when the issue in the parent script is resolved.

Upgrade from AX 2012 - Pre-upgrade checklist for data upgrade

2/18/2021 • 4 minutes to read • Edit Online

IMPORTANT

Upgrade is currently only supported from either Dynamics AX 2012 R2 or Dynamics AX 2012 R3. For each release, please update to the latest available cumulative update before upgrading to latest Finance and Operations application release.

This topic describes each task in the Microsoft Dynamics AX 2012 checklist that is associated with data upgrade to Finance and Operations.

Installation

Use the pre-upgrade checklist to enter data that will be required for the upgrade procedure.

- If upgrading from AX 2012 R3, install KB 4035163
- if upgrading from AX 2012 R2, install KB 4048614

Prepare model metadata

During data upgrade, one goal is to maintain element IDs between the existing AX 2012 environment and the upgraded Finance and Operations environment. To accomplish this goal, you must bring a copy of the element IDs from the AX 2012 environment into the Finance and Operations environment. AX 2012 stores element IDs in a table that is named ModelElement. This table is in the model database, which is a separate database from the AX 2012 business data database. During an upgrade to Finance and Operations, you must copy the AX 2012 database to Microsoft Azure. This process can be time consuming.

To avoid copying the whole model database to Azure SQL Database, use the following procedure to replicate the ModelElement table in the business data database. Later, during data upgrade runs, the database synchronization process will retrieve the required information from this replicated table and make sure that element IDs are maintained in the upgraded Finance and Operations environment.

1. In the Finance and Operations data upgrade checklist, click **Prepare model metadata**.
2. When you're prompted, click **Yes**.
3. Wait for the copy process to be completed.

If the process is successful, the task is marked as completed.

Prepare security role metadata

Another goal during data upgrade is to preserve security role assignments. This task resembles the previous "Prepare model metadata" task. Security role information that is stored in the AX 2012 model database must be copied to the AX 2012 business data database, so that the information is preserved in the Finance and Operations environment after upgrade. During data upgrade runs, the same security role will be restored in the upgraded Finance and Operations environment.

1. In the Finance and Operations data upgrade checklist, click **Prepare security role metadata**.
2. When you're prompted, click **Yes**.

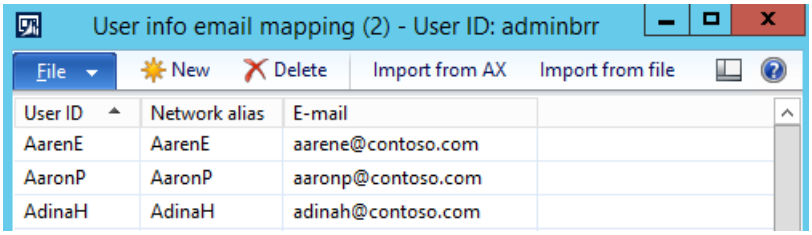
3. Wait for the copy process to be completed.

If the process is successful, the task is marked as completed.

Set up user mapping

In AX 2012, users are authenticated against an on-premises Active Directory server. However, in Finance and Operations, users are authenticated against Azure Active Directory (Azure AD). This task provides a form where you can map existing AX 2012 users to equivalent Azure AD users. The AX 2012 users will then be able to access Finance and Operations.

1. In the Finance and Operations data upgrade checklist, click **Set up user mapping**.
2. The **User info email mapping** form appears. Follow one of these steps to fill in the grid:
   - Import users from AX 2012, and then manually fill in the Azure AD email address:
     a. Click **Import from AX**. The grid is filled with existing users.
     b. For each user, enter the corresponding Azure AD email address, as shown in the following illustration.



   - Import users from a file. This option is faster. We recommend that you use this option when many users must be updated.
     a. In a comma-separated values (CSV) file, create the mapping between AX 2012 users and Azure AD email addresses. Your IT department can export a similar mapping from your on-premises Active Directory Domain Services (AD DS). The file should have two columns: **UserId** and **EmailAddress**.

> **NOTE**
> The first row in the file is treated as a header row and will be ignored during the import.

     b. After the file is ready, click **Import from file**, browse to the file, and import it.

        The grid should be filled with the mappings that you specified in the file.

If the imported file contains an entry that isn't valid, an error file is generated.

Validate baseline version

Run this task to validate that the current version can be upgraded.

- In the Finance and Operations data upgrade checklist, click **Validate baseline version**.

If the baseline version is one of the supported baseline versions, the task is marked as completed.

Archive retail salt data

This task is used to migrate the registry key that RetailSaltUtility uses. This tool is used for some deployments where the customer wants to inject a specific random value into the hash that is used to authenticate channel

users.

- In the Finance and Operations data upgrade checklist, click **Archive retail salt data**.

If the process is successful, the task is marked as completed.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Upgrade from AX 2012 - Dacpac process to upgrade data in Sandbox Tiers 2-5 environments

2/18/2021 • 10 minutes to read • Edit Online

IMPORTANT

Upgrade is currently only supported from either Dynamics AX 2012 R2 or Dynamics AX 2012 R3. For each release, please update to the latest available cumulative update before upgrading to latest Finance and Operations application release.

This topic is a process guide that will help customers who no longer have Remote Desktop Protocol (RDP) access to their Tier-2 through Tier-5 sandbox environments when they upgrade from Microsoft Dynamics AX 2012 to Finance and Operations apps. By using a new file format this is based on data-tier application packages (DACPACs), and the Database movement toolkit, customers can bring their AX 2012 schema and data into an existing empty database in a sandbox environment.

We **strongly recommend** that you run the data upgrade process in a development environment before you run it in a shared sandbox environment. This approach will help reduce the overall time that is required for a successful data upgrade. For more information, see Upgrade from AX 2012 - Pre-upgrade checklist for data upgrade.

In this process guide, you will learn how to complete the following steps:

- Open firewall access to your sandbox environment database.
- Clear the sandbox database of all objects.
- Publish the schema from your AX 2012 database to the sandbox database.
- Transfer the data from the source database to the target database.
- Apply the data upgrade package from Microsoft Dynamics Lifecycle Services (LCS).
- Copy the database to production for mock go-live validation and actual go-live.

Before you get started

Network latency is a critical component in data transfer. If your source SQL Server is geographically far away or has poor network latency to your sandbox Azure datacenter, then the process can take additional hours or potentially time out. Latency under 75 milliseconds is preferred. You can find your latency using a tool such as Azure Speed Test. If your latency is poor, you should consider backing up your AX 2012 database and restoring it on a cloud-hosted DevTest environment deployed to the same Azure datacenter as your target sandbox. After which you can perform the upgrade steps.

Open firewall access to your sandbox environment database

By default, all sandbox Standard Acceptance Test environments use Azure SQL Database as their database platform. The databases for these environments are protected by firewalls that restrict access to the Application Object Server (AOS) that they were originally deployed with.

However, your public IP address can be used to grant access to your source system. Different methods can be used to grant this access, depending on your environment type and RDP access. Regardless of the method, connection to the database requires that the server and the database be explicitly specified. You can find this information on the environment details page in LCS.

Find the user name record for **axdbadmin**, and make a note of the server and the database in the format **{sqlServer\sqlDatabase}**, such as **spartan-nam-opsprod365433\DBOpsProd365_SandboxUAT_d2145fe**. In this case, the server value for Microsoft SQL Server Management Studio (SSMS) is **spartan-nam-opsprod365433.database.windows.net**, and the database value is **DBOpsProd365_SandboxUAT_d2145fe**. To specify the database, you must select **Options** in the SSMS connection dialog box. Use the **axdbadmin** credentials (user name and password).

Microsoft-managed environments where RDP access is available

If you have RDP access to your sandbox, sign in to the sandbox AOS virtual machine (VM), and open SSMS. In SSMS, enter the server, user name, and password. On the **Options** tab, explicitly enter the database name from the **axdbadmin** record in LCS.

After you're connected, open a query against the database, and enter your IP address in the following Transact-SQL (T-SQL) command.

```
-- Create database-level firewall setting for IP a.b.c.d
EXECUTE sp_set_database_firewall_rule N'AX 2012 Upgrade via RDP', 'a.b.c.d', 'a.b.c.d';
```

Back in your source environment, open SSMS, and try to connect by using the same **axdbadmin** credentials against the User Acceptance Testing (UAT) database. You must verify that you can connect before you move on to the next steps.

Note that firewall rules are deleted whenever you do a database refresh or whenever you do a database import from LCS.

Microsoft-managed environments without RDP access

Follow the process for Enable just-in-time database access to allow-list your IP address to the database.

Before the allow-list entry expires, connect to the sandbox database by entering the server, user name, and password. On the **Options** tab, explicitly enter the database name from the **axdbadmin** record in LCS.

After you're connected, open a query against the database, and enter your IP address in the following Transact-SQL (T-SQL) command.

```
-- Create database-level firewall setting for IP a.b.c.d
EXECUTE sp_set_database_firewall_rule N'AX 2012 Upgrade without RDP', 'a.b.c.d', 'a.b.c.d';
```

A second entry is created in the firewall rules. This entry won't expire. Note that these firewall rules are deleted whenever you do a database refresh or whenever you do a database import from LCS.

Self-service environments

Unfortunately, sandbox environments of the self-service type aren't supported for AX 2012 upgrade scenarios, because they don't currently support data upgrade packages. Microsoft is working to add this support and will update this topic when more information is available.

Clear the sandbox database of all objects

In versions of the Database movement toolkit prior to Version 5, this was an explicit step that customers had to take via SSMS on the target database. However, this is now incorporated into the toolkit in Version 5 and later, and is handled via the PowerShell scripts.

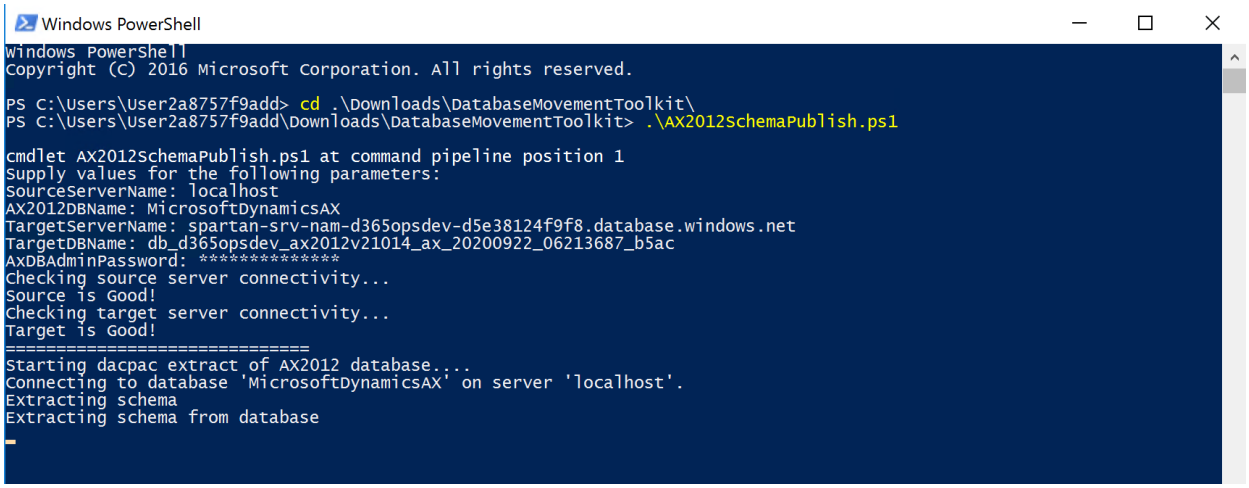
Publish the schema from AX 2012 to the sandbox database

Now that the database is empty, you can publish your non-upgraded 2012 schema. For this step, you should

download the latest version of the Database movement toolkit into your source environment.

Use Windows PowerShell to change the directory to the folder location where you unzipped the Database movement toolkit (for example, C:\dbmovement-toolkit\). Then run the **AX2012SchemaPublish.ps1** script. Use the following parameters:

- **SourceServerName** – The location where your source AX 2012 database is hosted. This location can be **localhost**.
- **AX2012DBName** – The name of your AX 2012 database (for example, **MicrosoftDynamicsAX**).
- **TargetServerName** – The server value of your sandbox database server (for example, **spartan-srv-12345.database.windows.net**). You retrieved this value in a previous step.
- **TargetDBName** – The name of your AXDB database (for example, **d365opsprod-12345**). You retrieved this value in a previous step.
- **AxDBAdminPassword** – The password for the **axdbadmin** database account.



During execution, the script performs a cleanup on the source and target databases. For the source database, it removes any non-essential users and non-essential schemas. If there are errors, it will stop the script and those errors will need to be reviewed. The source for the script is available in the toolkit called **Step1_CleanupSourceDB.sql**. For the target database, it removes all objects in a more thorough manner than was previously documented via SSMS. Any errors will need to be reviewed, and would stop the script. The source for the target cleanup is available in the toolkit called **Step0_CleanupTargetDB.sql**.

After cleanup, the toolkit uses SqlPackage.exe to extract only the database and schema definitions from your AX 2012 database as a 2012DBSource.dacpac file in the working directory. Next, it publishes this file to your sandbox environment by using the Profile.publish.xml publishing profile that is included in the Database movement toolkit. This publishing profile will skip several object types, such as SQL views, SQL users, statistics, and other objects that aren't required for the upgrade. If there are issues with the dacpac publish, you can find a detailed log in the working directory called **PublishDiag.log**.

When the script has finished running, open SSMS, and verify that you can see the tables in the sandbox database. Also confirm that the tables are empty and have no data.

> **NOTE**
>
> If you must start over because of an error, this script is designed to be run again as it will clean up both the source and target databases.

Populate legacy stored procedures

The data upgrade scripts require the legacy stored procedures and functions to be in place prior to running the upgrade. To transfer these scripts use SSMS.

Open SSMS, and connect to your source AX 2012 database. Right-click the database name, and then select **Tasks > Generate scripts**. On the **Introduction** page, select **Next**. On the **Choose Objects** page, select the **Select specific database objects** option, and then select the following check boxes:

- Stored procedures
- User-defined functions

On the **Set Scripting Options** page, select **Advanced**, and ensure the value is set to **Script DROP and CREATE**. Select **OK**, and then select **Save to new query window**. Select **Next** to move through the summary. The query window that appears contains a script that lists all the objects in the correct order so that the objects can be dropped from the database.

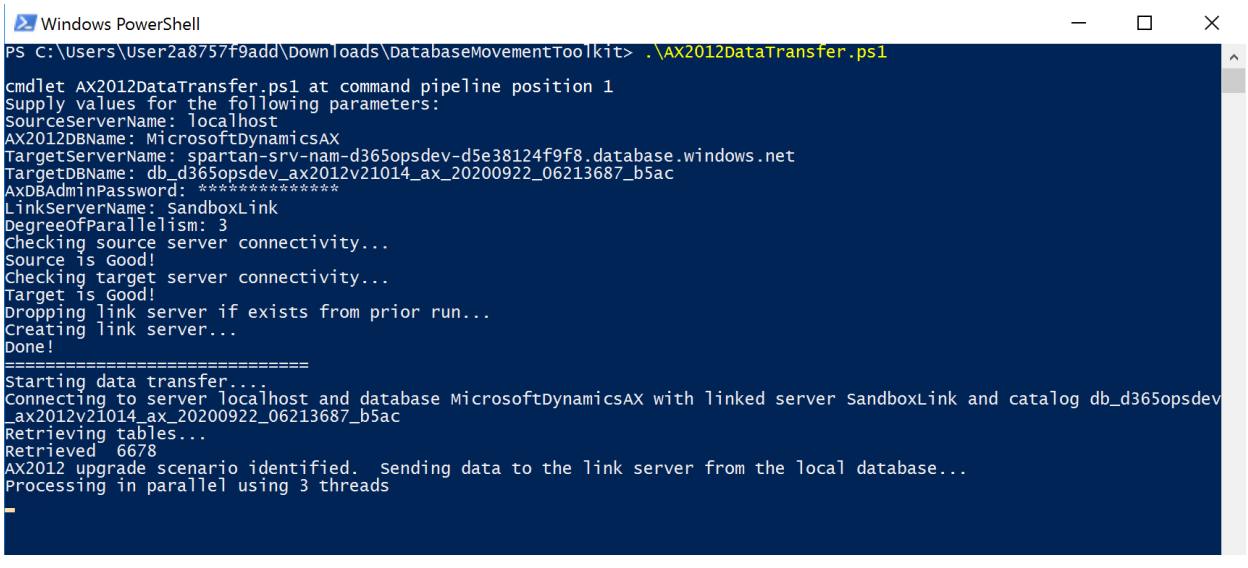
Using the script that was generated, connect to your target database using SSMS, and execute the script. This will create the AX 2012 stored procedures and functions in the target spartan database.

Transfer data from AX 2012 to the sandbox database

Now that the schema is in place, you must transfer the data to the target tables. A SQL-linked server will be used for this transfer. The transfer will also take advantage of parallelism features that are available in the instance of Windows PowerShell version 7.0 or later that is included in the Database Movement toolkit.

Use Windows PowerShell to change the directory to the folder location where you unzipped the Database movement toolkit (for example, C:\dbmovement-toolkit\). Then run the **AX2012DataTransfer.ps1** script. Use the following parameters:

- **SourceServerName** – The location where your source AX 2012 database is hosted. This location can be **localhost**.
- **AX2012DBName** – The name of your AX 2012 database (for example, **MicrosoftDynamicsAX**).
- **TargetServerName** – The server value of your sandbox database server (for example, **spartan-srv-12345.database.windows.net**). You retrieved this value in a previous step.
- **TargetDBName** – The name of your AXDB database (for example, d365opsprod-12345). You retrieved this value in a previous step.
- **AxDBAdminPassword** – The password for the axdbadmin database account.
- **LinkedServerName** – The name of the SQL-linked server to create (for example, **AX2012Link**).
- **DegreeOfParallelism** – The number of tables that should be processed in parallel (for example, **3**). The value should be no more than half of the number of cores that are available in the source environment, because the script is CPU-intensive and RAM-intensive.



During execution, if a linked server between the source server and the sandbox server doesn't already exist, the

script creates one. It then copies data from all the AX 2012 tables to the target database. It uses the **DegreeOfParallelism** parameter to process multiple tables at the same time.

> **NOTE**
> If you must start over because of an error, go back to the Publish the schema from AX 2012 to the sandbox database step.

Apply the data upgrade package from LCS

Now that the schema and the data have been moved to the sandbox environment, you can start the data upgrade process. As a prerequisite, make sure that your LCS project has been configured for AX 2012 upgrade. For more information, see Identify the project as an AX 2012 upgrade.

On the environment details page for your environment, select **Maintain** > **Apply updates**, and wait for the list of packages to be loaded. If you scroll to the bottom of the list, you will see that data upgrade packages are added to it as they are pulled in from the Shared asset library. It might take some time for all the available upgrade packages to be loaded.

Select the upgrade package that matches the version that you want to upgrade to. For example, to upgrade to version 10.0.14, select **AX2012DataUpgrade-10-0-14**.

Troubleshooting data upgrade errors

There are several ways to troubleshoot data upgrade errors. In some cases, you can view the error in the deployable package logs from LCS. For example, you can use this approach when the error is related to starting/stopping services and database synchronization.

If the upgrade fails while an upgrade script is running, you can view the errors in the ReleaseUpdateScriptsErrorLog table in the sandbox database. To access this table, connect to the sandbox database by using the information in previous steps.

If you can fix the data, you can resume the upgrade from LCS. However, note that you can't resume from LCS more than eight times. Any attempt to resume more than eight times will cause another failure, because the servicing systems don't allow more attempts. In this case, you can use the **Abort** button to cancel the upgrade package and try again later.

> **NOTE**
> If you must start over because of an error, go back to the Publish the schema from AX 2012 to the sandbox database step.

Copy the database to production for mock go-live and actual go-live

After the data upgrade is completed, apply the same customization packages from your sandbox environment to your production environment. You can then copy your sandbox environment AXDB database to the production environment.

For information about how to perform the copy operation, see Copy the sandbox database to production.

> **NOTE**
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Upgrade from AX 2012 - Data upgrade in sandbox environments

2/18/2021 • 14 minutes to read • Edit Online

IMPORTANT

Upgrade is currently only supported from either Dynamics AX 2012 R2 or Dynamics AX 2012 R3. For each release, please update to the latest available cumulative update before upgrading to latest Finance and Operations application release.

NOTE

This topic is being phased out in favor of a new process that is based on the dacpac file format. For information about the new process, see Upgrade from AX 2012 - Dacpac process to upgrade data in Sandbox Tiers 2-5 environments.

The output of this task is an upgraded database that you can use in a sandbox environment. In this topic, we use the term *sandbox* to refer to a Standard or Premier Acceptance Testing (Tier 2/3) or higher environment connected to a SQL Azure database. On this environment business users and functional team members can validate application functionality. This functionality includes customizations and the data that was brought forward from Microsoft Dynamics AX 2012.

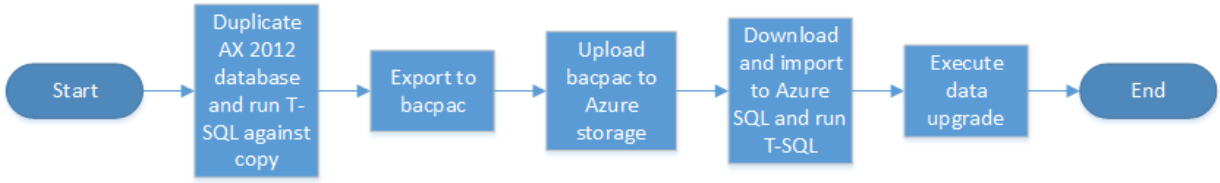
We strongly recommend that you run the data upgrade process in a development environment before you run it in a shared sandbox environment, because this approach will help reduce the overall time that is required for a successful data upgrade. For more information, see Upgrade from AX 2012 - Pre-upgrade checklist for data upgrade.

NOTE

It's very important that you install the latest version of SQLPackage before you start this process. To do this, go to Download and Install SQLPackage. We recommend that you use the .NET Core version. This can be saved and extracted to C:\temp.

Overview of the sandbox data upgrade process

Before you start to upgrade data in a sandbox environment, you will have already upgraded data in a development environment, as explained in Upgrade from AX 2012 - Pre-upgrade checklist for data upgrade. The two processes are very similar. The main difference is that a sandbox environment uses Microsoft Azure SQL Database for data storage, whereas a development environment uses Microsoft SQL Server. This technical difference in the database layer requires that you modify the data upgrade procedure slightly in a sandbox environment, because a backup from the AX 2012 database instance can't just be restored to SQL Database.



Here are the high-level steps in the upgrade process.

1. Turn off the AX 2012 AOS instances.
2. Create a copy of the AX 2012 database. We strongly recommend that you use a copy, because you must delete some objects in the copy that will be exported.
3. Export the copied database to a bacpac file by using a free SQL Server tool that is named SQLPackage.exe. This tool provides a special type of database backup that can be imported into SQL Database.
4. Optional, if you are running import from a cloud-hosted VM, you should upload the bacpac file to Azure storage.
5. Import the bacpac file by using SQLPackage.exe. This can be run from a local server or from a cloud-hosted VM. If you're using a local server, you will need to request JIT access to the Dynamics 365 database and set the required firewall rules. If you're using a cloud-hosted VM, download the bacpac file. You must then run a script against the imported database to reset the SQL database users.
6. Run the appropriate data upgrade package against the imported database.

Turn off the AX 2012 AOS instances

This task involves the AX 2012 system administrator and the server administrators.

Before you turn off AOS instances, you must stop all batch jobs that are running. A long-running batch job that has already started to run will prevent the system from stopping. Plan ahead, so that you can stop your AOS instances when needed. You might have to schedule batches so that they're completed some time before you turn off AX 2012.

You might have other systems that are integrated with the AX 2012 environment. You must also factor these systems into your plan to turn off AX 2012. For example, you might have to turn off the integrated systems some time before you turn off AX 2012 itself, so that any remaining in-flight transactions can be completed. The requirements for integrated systems vary widely from business to business. Therefore, your team of experts must plan for this scenario independently.

Create a copy of the AX 2012 database

You must create a copy of the AX 2012 database that you're upgrading, because you must delete some objects from the database. These objects include any Microsoft Windows authentication users. These changes make the modified database unusable for AX 2012. During this step, you will create a copy of the database and delete these objects.

This step must be done by the database administrator (DBA) or a person who has similar knowledge and experience.

To create a database copy, make a backup of the original database, and restore it under a new name. Make sure that enough space is available for both databases. You can create the copy on a different server. The version of the SQL Server instance that runs the database isn't important.

To perform this action, right-click the source database, and then select **Tasks** > **Backup**. Create a full-copy backup. To avoid overwriting your existing database backups, you might want to save the file under a different name in the backup directory.

Run the T-SQL script to prepare the database

This script prepares the database by removing users, removing procedures related to the AX 2012 RTM model store, cleaning up schemas, dropping views, and dropping references to tempDB.

After the copy is created, run the following Transact-SQL (T-SQL) script against it.

```
--remove NT users as these are not supported in Azure SQL Database
declare
@SQL varchar(255),
@UserName varchar(255)

set quoted_identifier off

declare     userCursor CURSOR for
select name from sys.sysusers where (isntuser = 1 or isntgroup =1) and name <> 'dbo'

OPEN userCursor
    FETCH userCursor into @UserName
    WHILE @@Fetch_Status = 0
       BEGIN
           set @SQL = 'DROP USER [' + @UserName + ']'
           exec(@SQL)
           FETCH userCursor into @UserName
       END
CLOSE userCursor
DEALLOCATE userCursor

go

--remove any AX 2012 RTM model store procedures that still exist
declare
@SQL varchar(255),
@procname varchar(255)

set quoted_identifier off

declare     procCursor CURSOR for
select name from sys.procedures where name like 'XI_%' or name like 'XU_%'

OPEN procCursor
    FETCH procCursor into @procname
    WHILE @@Fetch_Status = 0
       BEGIN
           set @SQL = 'DROP PROCEDURE [' + @procname + ']'
           exec(@SQL)
           FETCH procCursor into @procname
       END
CLOSE procCursor
DEALLOCATE procCursor

go

--If you receive a message that you cannot delete users because they own a schema, then check which schema
the user owns.
--Either change the ownership to another user (for example to dbo) or drop the schema if it does not contain
any objects.
--The examples below are for an AX 2012 demo environment. You will need to edit this for your specific
environment.

    if exists (select 1 from sys.schemas where name = 'contoso\admin')
begin
    drop schema [contoso\admin]
end
if exists (select 1 from sys.schemas where name = 'contoso\Domain Users')
begin
    drop schema [CONTOSO\Domain Users]
end
go

--drop all views in the current database because some refresh the tempDB, which is not a supported action in
Azure SQL Databases
declare
```

```
@SQL2 varchar(255),
@ViewName varchar(255)

set quoted_identifier off

declare     viewCursor CURSOR for

select viewname = v.name
from sys.views v
order by v.name

OPEN viewCursor

FETCH viewCursor into @ViewName
    WHILE @@Fetch_Status = 0
       BEGIN
           set @SQL2 = 'DROP VIEW ' + @ViewName
           exec(@SQL2)
           FETCH viewCursor into @ViewName
       END
CLOSE viewCursor
DEALLOCATE viewCursor
go

-- Drop the following procedure because it contains a tempDB reference that is not supported in Azure SQL
Database
If exists (select 1 from sys.procedures where name = 'MaintainShipCarrierRole')
begin
    drop procedure MaintainShipCarrierRole
end
```

Export the copied database to a bacpac file

> **NOTE**
>
> This topic is being phased out in favor of a new process that is based on the dacpac file format. For information about the new process, see Upgrade from AX 2012 - Dacpac process to upgrade data in Sandbox Tiers 2-5 environments.

Export the copied database to a bacpac file by using the SQLPackage.exe tool. This step should be done by the DBA or a team member who has equivalent knowledge.

> **IMPORTANT**
>
> It's very important that you install the latest version of SQL Server Management Studio before you start this step. Although SQLPackage is present in earlier versions of Management Studio, it won't work correctly for this step unless you first install the latest version.

This step is important, because the export will have to be done again during the downtime before go-live. Here are some tips:

- The bacpac process is very I/O and CPU intensive. Therefore, run the export on a high-powered machine.
- SQLPackage should be run locally on the machine that hosts the database. Don't run SQLPackage on a local laptop that you connect to the database machine, because this process is also network intensive.

Next, open a **Command Prompt** window as an administrator, and run the following commands.

```
cd C:\Program Files (x86)\Microsoft SQL Server\130\DAC\bin\

SqlPackage.exe /a:export /ssn:localhost /sdn:<database to export> /tf:D:\Exportedbacpac\my.bacpac
/p:CommandTimeout=1200 /p:VerifyFullTextDocumentTypesSupported=false
```

Here is an explanation of the parameters:

- **ssn** (source server name) – The name of the SQL Server to export from. For this process, the parameter should always be set to **localhost**.
- **sdn** (source database name) – The name of the database to export.
- **tf** (target file) – The path and name of the file to export to. The folder should already exist, but the file will be created by the process.
- **/p:CommandTimeout** – The per-query timeout value. This parameter enables larger tables to be exported without a timeout.

Upload the bacpac file to Azure storage or the LCS Asset Library

The bacpac file you have created will need to be copied to the AOS machine in your Azure hosted sandbox environment. There are several reasons for this:

1. The Azure SQL Database instance used by your Tier 2 (or higher) sandbox environment has firewall rules preventing access from outside of the environment itself.
2. Performance of bacpac import is multiple times faster when importing from a machine within the same Azure datacenter as the Azure SQL database instance.

You can choose how you would like to move the bacpac file to the AOS machine - you may have your own SFTP or other secure file transfer service. We recommend to use our Azure storage, which would require that you acquire your own Azure storage account on your own Azure subscription (this is not provided within the Dynamics subscription itself). There are free tools to help you to move files between Azure storage, from a command line you can use Azcopy, or for a GUI experience you can use Microsoft Azure storage explorer. Use one of these tools to first upload the backup from your on-premises environment to Azure storage and then on your download it on your development environment.

Another option is to use the Asset library in Microsoft Dynamics Lifecycle Services (LCS). However, the upload and download will take longer than Azure storage. To use this option:

1. Sign in to your project in LCS and go to your Asset library.
2. Select the Database backup tab.
3. Upload the bacpac file. Since the removal of RDP access to Sandbox AOS servers, we recommend that you use a cloud-hosted environment running in the same region as the sandbox environment to import the file. You can download the bacpac onto the cloud-hosted VM by signing in to LCS on that machine and downloading it from the LCS Asset library.

Import the bacpac file into SQL Database

> **NOTE**
>
> This topic is being phased out in favor of a new process that is based on the dacpac file format. For information about the new process, see Upgrade from AX 2012 - Dacpac process to upgrade data in Sandbox Tiers 2-5 environments.

During this step, you will import the exported bacpac file into the SQL Server database instance that your sandbox environment uses. As stated above, since the removal of RDP access to Sandbox AOS servers, we recommend that you use a cloud-hosted environment running in the same region as the sandbox environment

to import the file. You can request JIT access to the database and run from a local server. To do that, follow the process for Enable just-in-time database access to allow-list your IP address to the database.

You must first install the latest version of Management Studio on your cloud-hosted VM or local server. You will then import the file by using the SQLPackage.exe tool.

If you are using a cloud-hosted environment, for performance reasons, we recommend that you put the bacpac file on drive D on the AOS machine. On Azure virtual machines (VMs), drive D is a physical disk that typically has higher performance than other available disks.

Open a **Command Prompt** window as an administrator, and run the following commands.

```
cd C:\Program Files (x86)\Microsoft SQL Server\130\DAC\bin\

SqlPackage.exe /a:import /sf:D:\Exportedbacpac\my.bacpac /tsn:<azure sql database server
name>.database.windows.net /tdn:<New database name> /tu:sqladmin /tp:<password from LCS> /mp:64
/p:CommandTimeout=1200 /p:DatabaseEdition=<Edition> /p:DatabaseServiceObjective=<Service objective>
/p:DatabaseMaximumSize=<Maximum size>
```

Here is an explanation of the parameters:

- **sf** (source file) – The path and name of the file to import from.
- **tsn** (target server name) – The name of the SQL Azure server to import to. The name can be found in LCS. Suffix it with **.database.windows.net**.
- **tdn** (target database name) – The name of the database to import to. The database should not already exist. The import process will create it.
- **tu** (target user) – The SQL user name for the target SQL Database instance. We recommend that you use **sqladmin**. You can retrieve the password for this user from your LCS project.
- **tp** (target password) – The SQL password for the target SQL database instance.
- **mp** (max parallelism) - Specifies the degree of parallelism for concurrent operations running against a database. The default value is 8. A value of 64 provides the best performance in most scenarios.
- **/p:CommandTimeout** – The per-query timeout value. This parameter enables larger tables to be exported without a timeout.
- **/p:DatabaseEdition** – Specifies the edition of the database such as Basic, Standard, Premium, GeneralPurpose, BusinessCritical, or Hyperscale. To meet performance requirements and comply with your service agreement, use the same service objective level as the current Finance and Operations database (AXDB) on this environment. You can check the value for the existing database by using Management Studio. Right-click the database, and then select **Properties**.
- **/p:DatabaseServiceObjective** – Specifies the performance level of the database such as S1, P2, P4, or GP_Gen5_8. To meet performance requirements and comply with your service agreement, use the same service objective level as the current Finance and Operations database (AXDB) on this environment. You can check the value for the existing database by using Management Studio. Right-click the database, and then select **Properties**.
- **/p:DatabaseMaximumSize** – Defines the maximum size in GB of an Azure SQL database. You may need to use this parameter to allow a large database to be imported.

After you run the commands, you may receive the following warning. You can safely ignore it.

If you receive an error message about a non-valid value, double-check your parameters. If they are okay, you may need to use a newer version of SqlPackage. You can use the Windows .NET Core version without the need to install. It is a .zip file that can be extracted to C:\Temp\Sqlpackage-dotnetcore, for example. Now when you import the database, instead of using the Sqlpackage.exe under C:\Program Files (x86) you can use the

Sqlpackage.exe in C:\Temp\Sqlpackage-dotnetcore.

Run a T-SQL script to update the database

Run the following script against the imported database. The script performs the following actions:

- Recreates database users
- Sets the correct performance parameters
- Enables the SQL Query Store feature

```
CREATE USER axdeployuser FROM LOGIN axdeployuser
EXEC sp_addrolemember 'db_owner', 'axdeployuser'

CREATE USER axdbadmin WITH PASSWORD = 'password from lcs'
EXEC sp_addrolemember 'db_owner', 'axdbadmin'

CREATE USER axruntimeuser WITH PASSWORD = 'password from lcs'
EXEC sp_addrolemember 'db_datareader', 'axruntimeuser'
EXEC sp_addrolemember 'db_datawriter', 'axruntimeuser'

CREATE USER axmrruntimeuser WITH PASSWORD = 'password from lcs'
EXEC sp_addrolemember 'ReportingIntegrationUser', 'axmrruntimeuser'
EXEC sp_addrolemember 'db_datareader', 'axmrruntimeuser'
EXEC sp_addrolemember 'db_datawriter', 'axmrruntimeuser'

CREATE USER axretailruntimeuser WITH PASSWORD = 'password from lcs'
EXEC sp_addrolemember 'UsersRole', 'axretailruntimeuser'
EXEC sp_addrolemember 'ReportUsersRole', 'axretailruntimeuser'

CREATE USER axretaildatasyncuser WITH PASSWORD = 'password from lcs'
EXEC sp_addrolemember 'DataSyncUsersRole', 'axretaildatasyncuser'

ALTER DATABASE SCOPED CONFIGURATION  SET MAXDOP=2
ALTER DATABASE SCOPED CONFIGURATION  SET LEGACY_CARDINALITY_ESTIMATION=ON
ALTER DATABASE SCOPED CONFIGURATION  SET PARAMETER_SNIFFING= ON
ALTER DATABASE SCOPED CONFIGURATION  SET QUERY_OPTIMIZER_HOTFIXES=OFF
ALTER DATABASE imported-database-name SET COMPATIBILITY_LEVEL = 130;
ALTER DATABASE imported-database-name SET QUERY_STORE = ON;
```

Run the data upgrade deployable package

For Tier-2 sandbox environments, you can now run the data upgrade package directly from LCS, just as you can do for Tier-1 DevTest environments. To apply the package, go to your environment in LCS, and select **Maintain** > **Apply updates**. Scroll to the bottom of the list, and wait for the data upgrade packages to be loaded from the Shared asset library. It might take some time for the packages to be loaded. If no data upgrade packages appear in the list, go to your project settings in LCS, and make sure that your legacy system is set to **AX2012 Upgrade**. You can then apply the packages.

The name of the package that you select should match your version. For example, to upgrade to version 10.0.14, select **AX2012DataUpgrade-10-0-14**.

For more information, see Upgrade data in development or demo environments.

Upgrade a copy of the database in a development environment

It is highly recommended to have upgraded the same database in a development environment. If you have a copy of the database available for development environments, it will be much easier to investigate bugs that are found in the upgraded sandbox environment.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Data upgrade process for AX 2012 to Dynamics 365 Finance + Operations (on-premises)

2/18/2021 • 13 minutes to read • Edit Online

This topic describes the process for upgrading Microsoft Dynamics AX 2012 databases to Dynamics 365 Finance + Operations (on-premises) version 10.0.x. Currently, upgrade is supported only from either Dynamics AX 2012 R2 or Dynamics AX 2012 R3.

> **IMPORTANT**
>
> This topic explains the process for doing a data upgrade only. For information about how to do a code upgrade, see the upgrade guides that are available for cloud versions. The code upgrade tooling is available only through Microsoft Dynamics Lifecycle Services (LCS).

AX 2012 upgrade to Dynamics 365 Finance + Operations (on-premises)

Two upgrade methods are currently supported:

- **Upgrade from inside the VHD** – This method involves copying your database into the virtual hard disk (VHD) and running the upgrade from inside the VHD. Overall, this method is easier.
- **Upgrade where the VHD points to your database** – This method involves pointing the VHD upgrade process to your database. The upgrade process is still run from inside the VHD.

> **NOTE**
>
> The VHD doesn't require external network access to run the upgrade process.

Prerequisites

1. Sign up for a preview subscription.
2. For each AX 2012 release, update to the most recent cumulative update that is available before you upgrade to the most recent Finance + Operations application release.
3. Install the pre-upgrade checklist. For more information, see Installation.
4. Go through the data upgrade preparation steps. You can skip the "Set up user mapping" step. This step is relevant only for cloud-hosted upgrades.
5. Make a backup of your database (MicrosoftDynamicsAX). For more information, see Create a Full Database Backup.
6. In LCS, go to the Shared asset library by selecting the tile on the right side of the page. Then, under **Select asset type**, select **Downloadable VHD**, and download all parts of the VHD package that most closely matches the version that you will upgrade to in your on-premises environment. The image requires a large amount of disk space. Therefore, be sure to download and extract the package on a drive that has enough free space.
7. The files that you downloaded are a self-extracting zip file. Extract the VHD to a location that has a good amount of free space.

8. Use Hyper-V to start a virtual machine (VM) and attach the VHD. (Note that the VM must be Generation 1.)
9. Connect to the VM. For information about the credentials, see Running the Virtual Machine (VM) locally.
10. Depending on your planned on-premises target version of 10.0.x and the VHD image that you downloaded, you might have to download and apply the required application update and platform update from the Shared asset library. Under **Select asset type**, select **Software deployable package**. For more information, see Install deployable packages from the command line.

    > **IMPORTANT**
    > In any case, make sure that you've applied the most recent quality update to your VHD, to ensure that it contains the most recent fixes for doing data upgrades.

11. If you have any extensions or customizations, install them on the VHD now. Otherwise, the upgrade process will remove any data that is related to customizations. If you must prepare your environment before the upgrade, check with your independent software vendor (ISV) or value-added reseller (VAR).

Upgrade from inside the VHD

1. Restore the backup that you created to the OneBox VM. For more information, see Restore a Database Backup Using SSMS.
2. Optional: If the name of your restored database isn't **AXDB**, open Windows PowerShell as an administrator, and run the following script.

   ```
   .\Configure-On-Premises-Upgrade.ps1 -DatabaseName '<DB-name>'
   ```

   > **NOTE**
   > Replace **<DB-name>** with the name of your database (for example, **AXDB**). If you want to edit more values, see the appendix later in this topic.

   The script will run a database connection test to verify that the information that you provided is valid.
3. In LCS, go to the Shared asset library. Under **Select asset type**, select **Software deployable package**, and then select **AX2012DataUpgrade-10-0-8** to download the **MajorVersionDataUpgrade.zip** file.
4. Copy the file, paste it in the desired location (for example: **c:\D365FFOUpgrade**), and unzip it.
5. Open a Command Prompt window as an administrator, change the directory to the folder that you just unzipped, and run the following commands.
   a. `AxUpdateInstaller.exe generate -runbookid=upgrade -runbookfile=upgrade.xml -topologyfile=defaulttopologydata.xml -servicemodelfile=defaultservicemodeldata.xml`
   b. `AxUpdateInstaller.exe import -runbookfile=upgrade.xml`
   c. `AxUpdateInstaller.exe execute -runbookid=upgrade`
6. After the upgrade process is successfully completed, back up the newly upgraded database. If you have customizations from ISVs or VARs, check whether you must run some post–data upgrade scripts.
7. Restore the database into your on-premises environment's SQL Server, but give it a name that differs from the name of the AX 2012 database (for example, name it **AXDBupgraded**). The restored database must be configured. Follow the steps in Configure the Finance + Operations database.
8. Deploy a new Dynamics 365 Finance + Operations (on-premises) environment.

- If you have customizations, follow these steps:

    a. In LCS, go to the Shared asset library.

    b. Under **Select asset type**, select **Model**, and then download **Dynamics 365 Finance + Operations on-premises, Version 10.0.x Demo Data**. Select the version that is closest to the 10.0.x environment that you will deploy as the on-premises baseline.

    c. Use the restore backup option for SQL Server to create a new database from this file. (Typically, this database is named **AXDB**.) For more information, see Restore a Database Backup Using SSMS.

    d. The demo database must be configured. Follow the steps in Configure the Finance + Operations database.

    e. In LCS, set up a new environment, and deploy it with version 10.0.x. For more information, see Set up and deploy on-premises environments (Platform update 12 and later). When you deploy the environment, the name of the database that you specify should be the name of the database that you created earlier (typically **AXDB**).

    f. Apply your own customizations, and ISV and VAR modules, to the newly created 10.0.x environment. Otherwise, when the environment is initially synced with the database, it will delete any customization-related or extension-related data.

    g. Shut down on-premises Application Object Server (AOS), Business Intelligence (BI), and Management Reporter (MR) servers, or stop the services from the Azure Service Fabric portal by selecting **Deactivate (Restart)**.

    h. Rename or delete the demo database (typically **AXDB**) that you used for deployment, and then rename your new database (typically **AXDBupgraded**) to the name that the demo database had (typically **AXDB**).

    i. The renamed database must be configured. Follow the steps in Configure the Finance + Operations database.

    j. Start on-premises AOS, BI, and MR servers, or start the services from the Service Fabric portal by selecting **Activate**.

    > **NOTE**
    > The Database synchronization process will be triggered when the AOS nodes start up, and your environment will be unavailable until the process is completed.

- If you don't have customizations, follow these steps:

    a. Optional: Rename your old database (typically **AXDBold**), and then rename your new database (typically **AXDB**). In the next step, make sure that you enter the name of the upgraded database.

    b. In LCS, set up a new environment, and deploy it with version 10.0.x (Redeploy). For more information, see Set up and deploy on-premises environments (Platform update 12 and later).

Upgrade where the VHD points to your database

1. Back up the database from your on-premises environment (typically **AXDB**). For more information, see Create a Full Database Backup (SQL Server).

2. Restore the backup that you just created into the database server, and give it a different name (for example, **AXDBtoupgrade**). For more information, see Restore a Database Backup Using SSMS.

3. Open Windows PowerShell as an administrator, and run the following script.

```
.\Configure-On-Premises-Upgrade.ps1 -DatabaseName '<DB-name>' -DatabaseServer '<SqlServerName>' -
DatabaseUser '<User>' -DatabasePassword '<Password>'
```

NOTE

- Replace **<DB-name>**, **<SqlServerName>**, **<User>**, and **<Password>** with the values that you require.
- Only SQL Server authentication is officially supported for this upgrade. For more information, see Create a Database User.
- You must add the certificate authority certificate that signed your SQL Server certificate to the trusted certificate authorities store in your Onebox VHD. For more information, see Installing the trusted root certificate.
- Make sure that the database user that you use has the **sysadmin** server role, or at least **All Privileges**, assigned on the database that you want to upgrade. Also make sure that the user has permissions to access tempDB. Step 6 of the upgrade process will fail if these conditions aren't met.
- When you install the certificate authority certificate in the OneBox VHD, make sure that you use the fully qualified domain name (FQDN) or IP address to connect to the database that appears there. If you can't access the database by using the domain name, because it doesn't point to that server, edit your hosts file, and add the FQDN and the IP address that the FQDN should be resolved to.

4. In LCS, go to the Shared asset library. Under **Select asset type**, select **Software deployable package**, and then select **AX2012DataUpgrade-10-0-8** to download the **MajorVersionDataUpgrade.zip** file.
5. Copy the file, paste it in the desired location (for example: **c:\D365FFOUpgrade**), and unzip it.
6. Open a Command Prompt window as an administrator, change the directory to the folder that you just unzipped, and run the following commands.
   a. `AxUpdateInstaller.exe generate -runbookid=upgrade -runbookfile=upgrade.xml -topologyfile=defaulttopologydata.xml -servicemodelfile=defaultservicemodeldata.xml`
   b. `AxUpdateInstaller.exe import -runbookfile=upgrade.xml`
   c. `AxUpdateInstaller.exe execute -runbookid=upgrade`
7. If you have customizations from ISVs or VARs, check whether you must run some post–data upgrade scripts.
8. Run the **Configure-OnpremUpgrade.ps1** script by using the values that are stated in the Resetting the VHD database (Optional) section later in this topic.
9. Configure your upgraded database for Finance + Operations by following the steps in Configure the Finance + Operations database.
10. Deploy a new Dynamics 365 Finance + Operations (on-premises) environment.
    - If you have customizations, follow these steps:
      a. In LCS, go to the Shared asset library.
      b. Under **Select asset type**, select **Model**, and then download **Dynamics 365 Finance + Operations on-premises, Version 10.0.x Demo Data**. Select the version that is closest to the 10.0.x environment that you will deploy as the on-premises baseline.
      c. Use the restore backup option for SQL Server to create a new database (typically **AXDB**) from this file. For more information, see Restore a Database Backup Using SSMS.
      d. The demo database must be configured. Follow the steps in Configure the Finance + Operations database.
      e. In LCS, set up a new environment, and deploy it with version 10.0.x (Redeploy). For more

information, see Set up and deploy on-premises environments (Platform update 12 and later). When you deploy the environment, the database that you should specify should be the database that you configured earlier (typically **AXDB**).

f. Apply your own customizations, and ISV and VAR modules, to the newly created 10.0.x environment. Otherwise, when the environment is initially synced with the database, it will delete any customization-related or extension-related data.

g. Shut down on-premises AOS, BI, and MR servers, or stop the services from the Service Fabric portal.

h. Rename or delete the demo database (typically **AXDB**) that you used for deployment, and then rename your new database (typically **AXDBupgraded**) to the name that the demo database had (typically **AXDB**).

i. Start on-premises AOS, BI, and MR servers, or start the services from the Service Fabric portal.

> **NOTE**
> The Database synchronization process will be triggered when the AOS nodes start up, and your environment will be unavailable until the process is completed.

- If you don't have customizations, follow these steps:

  a. Optional: Rename your old database (typically **AXDBold**), and then rename your new database (typically **AXDB**). In the next step, make sure that you enter the name of the upgraded database.
  b. Set up a new environment, and deploy it with version 10.0.x. For more information, see Set up and deploy on-premises environments (Platform update 12 and later).

Configuring existing users

If you followed either of the previous procedures, you can sign in by using the Administrator user that you specified in LCS. However, none of your other users can sign in until they have been configured for the new system. Run a **Select** statement against your USERINFO table, and make a note of the value in the **NETWORKDOMAIN** field for the Administrator user (for example, `https://adfs.contoso.com/adfs` or `http://adfs.contoso.com/adfs/services/trust`). Then, for all interactive users who should be able to sign in, set the **NETWORKDOMAIN** field to the same value that the Administrator user has. The **NETWORKALIAS** field must also be modified. In Finance + Operations, this field is set to the user's email address (for example, `testuser@contoso.com`).

Resetting the VHD database (Optional)

If you used the **Configure-On-Premises-Upgrade.ps1** script, run the following command to reset your database to the default configuration.

```
.\Configure-OnPremUpgrade.ps1 -DatabaseName 'AxDB' -DatabaseServer 'localhost' -DatabaseUser 'axdbadmin' -
DatabasePassword 'AOSWebSite@123'
```

Appendix

Using the Configure-On-Premises-Upgrade.ps1 script

IMPORTANT

This script is intended to be run only from a OneBox VHD environment.

The script requires that you pass at least the **DatabaseName** parameter. If you don't pass this parameter, the script automatically requests it.

You can pass an additional parameter, such as **DatabaseServer** or **DatabaseUser**, if you want. However, in this case, the script will request all additional parameters. This behavior occurs because the script will assume that you want to point the database connection to a machine outside the VM. Therefore, those parameters are required to correctly establish the connection.

The following parameters can be passed to the script:

- **-DatabaseName** – The name of the database to upgrade.
- **-DatabaseServer** – The database server that contains the Finance + Operations database.
- **-DatabaseUser** – The user name for SQL Server Authentication.
- **-DatabasePassword** – The password for SQL Server Authentication.

After the configuration has been passed, the script uses the new parameters to run a database connection test. If the script can't connect to the database, we recommend that you debug the connection from SQL Server Management Studio or another tool.

Configure-On-Premises-Upgrade.ps1

```
<#
.Synopsis
    Configures a OneBox deployment to upgrade an OnPrem 7.x database to OnPrem 10.0.x

.DESCRIPTION
    This must be executed before the upgrade process is carried out.

.EXAMPLE
    .\Configure-OnPremUpgrade.ps1 -DatabaseName 'AxDB'

    .\Configure-OnPremUpgrade.ps1 -DatabaseName 'AxDB' -DatabaseServer '127.0.0.1' -DatabaseUser 'axdbadmin'
-DatabasePassword 'secretPass'
#>
[CmdletBinding()]
param
(
    # Database server containing Microsoft Dynamics 365 for Operations, on-premises database.
    [AllowNull()]
    [string] $DatabaseServer,

    # Database name that you want to upgrade.
    [Parameter(Mandatory = $true)]
    [string] $DatabaseName,

    # Username for SQL Authentication.
    [AllowNull()]
    [string] $DatabaseUser,

    # Password for SQL Authentication.
    [AllowNull()]
    [string] $DatabasePassword
)

$webroot = "C:\AOSService\webroot"

$commandParameter = " -decrypt `"$webroot\web.config`""
```

```
$command = Resolve-Path "$webroot\bin\Microsoft.Dynamics.AX.Framework.ConfigEncryptor.exe"

Start-Process $command $commandParameter -PassThru -Wait

if([string]::IsNullOrEmpty($DatabaseUser) -and [string]::IsNullOrEmpty($DatabasePassword) -and
[string]::IsNullOrEmpty($DatabaseServer)) {

    [xml]$web = Get-Content $webroot\web.config

    $web.SelectSingleNode("configuration/appSettings/add[@key='DataAccess.Database']").value =
[string]$DatabaseName

}
else {

    if([string]::IsNullOrEmpty($DatabaseServer)){

        $DatabaseServer = if($value = Read-Host 'What is the IP or FQDN of the Database server?
[127.0.0.1]') {$value} else {'127.0.0.1'}

    }

    if([string]::IsNullOrEmpty($DatabaseUser)){

        $DatabaseUser = if($value = Read-Host 'What is the SQL Authentication username? [axdbadmin]')
{$value} else {'axdbadmin'}

    }

    if([string]::IsNullOrEmpty($DatabasePassword)){

        $dbPassEn = if($value = Read-Host 'What is the SQL Authentication password?' -AsSecureString)
{$value} else {''}

        $BSTR = [System.Runtime.InteropServices.Marshal]::SecureStringToBSTR($dbPassEn)

        $DatabasePassword = [System.Runtime.InteropServices.Marshal]::PtrToStringAuto($BSTR)

    }

    [xml]$web = Get-Content $webroot\web.config

    $web.SelectSingleNode("configuration/appSettings/add[@key='DataAccess.DbServer']").value =
[string]$DatabaseServer

    $web.SelectSingleNode("configuration/appSettings/add[@key='DataAccess.Database']").value =
[string]$DatabaseName

    $web.SelectSingleNode("configuration/appSettings/add[@key='DataAccess.SqlUser']").value =
[string]$DatabaseUser

    $web.SelectSingleNode("configuration/appSettings/add[@key='DataAccess.SqlPwd']").value =
[string]$DatabasePassword

}
#Save Configuration to webroot config
$web.Save("$webroot\web.config")

#Reloading the configuration to run test
[xml]$web = Get-Content $webroot\web.config

$TestDbServer = $web.SelectSingleNode("configuration/appSettings/add[@key='DataAccess.DbServer']").value

$TestDbName = $web.SelectSingleNode("configuration/appSettings/add[@key='DataAccess.Database']").value

$TestDbUser = $web.SelectSingleNode("configuration/appSettings/add[@key='DataAccess.SqlUser']").value

$TestDbPass = $web.SelectSingleNode("configuration/appSettings/add[@key='DataAccess.SqlPwd']").value
```

```
#Setting up connection test.

$dbConn = New-Object System.Data.SqlClient.SqlConnection

$dbConn.ConnectionString = "Data Source=$TestDbServer;User
ID=$TestDbUser;Password=`"$TestDbPass`";Database=$TestDbName"

try{

    $dbConn.Open()

    $result = $true

}

catch{

    $result = $_.Exception.Message

}
Finally{

    $dbConn.Close()

}

$commandParameter = " -encrypt `"$webroot\web.config`""

Start-Process $command $commandParameter -PassThru -Wait

if($result -ne $true){

    Write-Host "`nThe connection to the Database Server failed:" -ForegroundColor Red
    Write-Host $result -ForegroundColor Red

}
else{

    Write-Host "`nThe connection to the Database Server was successful!" -ForegroundColor Green

}
```

Troubleshooting

- Exception calling "Open" with "0" argument(s): "Cannot open database "AxDB1" requested by the login. The login failed. Login failed for user 'axdbadmin'." You supplied the Wrong database name or the user doesn't have access to that database.
- Exception calling "Open" with "0" argument(s): "A network-related or instance-specific error occurred while establishing a connection to SQL Server. The server was not found or was not accessible. Verify that the instance name is correct and that SQL Server is configured to allow remote connections. (provider: Named Pipes Provider, error: 40 - Could not open a connection to SQL Server)". The script could not establish a connection with the SQL Server specified. Check the ip/fqdn and port that you used.
- Exception calling "Open" with "0" argument(s): "Login failed for user 'axdbadmin'." The supplied login credentials are not correct.

Upgrade from AX 2012 - Cutover testing (Mock cutover)

2/18/2021 • 5 minutes to read • Edit Online

IMPORTANT

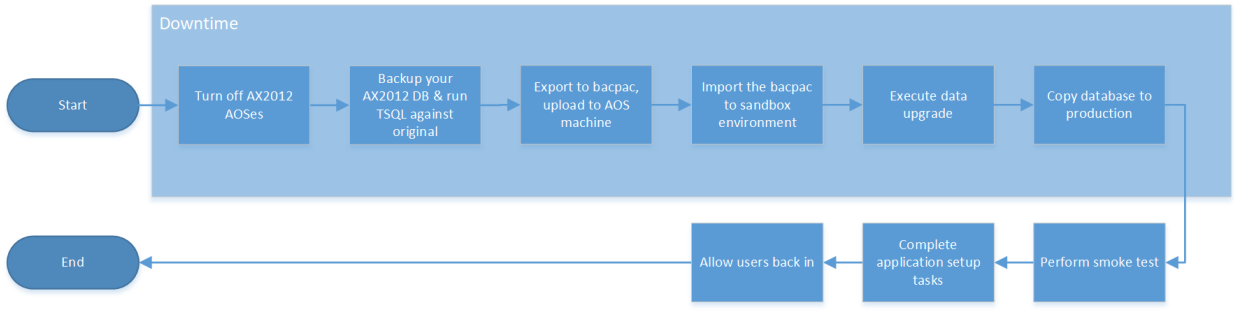
Upgrade is currently only supported from either Dynamics AX 2012 R2 or Dynamics AX 2012 R3. For each release, please update to the latest available cumulative update before upgrading to latest Finance and Operations application release.

Cutover is the term that we use for the final process of getting a new system live. The cutover process consists of the tasks that occur after Microsoft Dynamics AX 2012 is turned off, but before Finance and Operations is turned on. The purpose of upgrade cutover testing (mock cutover) is to practice the cutover process, to help guarantee a smooth experience for everyone who is involved during the actual cutover to go-live.

There are three main workstreams during a cutover:

- **Technical workstream** – This workstream includes the data upgrade execution process. Your business will enforce a limit on the amount of downtime that is allowed. During this downtime, neither AX 2012 nor Finance and Operations will be available. This workstream might have to tune the data upgrade procedure to meet the business's downtime limit.
- **Functional workstream** – This workstream includes the configuration tasks that are performed after the data upgrade is completed. All these tasks must be documented and quantified, and a resource must be assigned, because both the functional workstream and the technical workstream must fit within the business's downtime limit.
- **AX 2012 rollback** - This workstream includes rolling back to an AX 2012 environment. Although it's unlikely that you will have to roll back, it's very important that you have a tested process in case you require it.

The following illustration shows the overall process for cutover to go-live as it will occur in the production environment.



The mock cutover process is very similar to a basic data upgrade validation in a sandbox environment. We assume that you are familiar with that process, and have already performed it. Mock cutover differs in the following ways:

- After you perform a data upgrade in the sandbox environment, a LCS other type service requests is needed to copy your upgraded database from the data upgrade sandbox environment into your production environment. The email template below is provided for your use

> [!Copy] This is a request for 2012 data upgrade database copy from the sandbox environment to production. I acknowledge that this will overwrite the database currently in production.

- We added the following tasks:
  - Perform a smoke test.
  - Complete application setup tasks. This step can be large, depending on the functionality that is used. During this step, the functional team configures new application functionality so that it's ready to be used in the upgraded system.
  - Allow users back in. Notify your user base that the upgrade is completed and that they can use the system again.

> **NOTE**
> In this article, we use the term *sandbox* to refer to a Standard or Premier Acceptance Testing (Tier 2 or 3) or higher environment connected to a SQL Azure database.

Technical workstream

The technical workstream involves various technical team members: the database administrator (DBA), the AX 2012 system administrator, server administrators, and developers who are familiar with AX 2012 and Finance and Operations.

During cutover testing, the technical team is focused on performance and reliability testing of the data upgrade process, to make sure that it meets the business's downtime limit. Many elements of hardware and software are involved in this process. Some of these elements are on-premises, whereas others are in the Microsoft cloud. In addition, many elements of custom application code and standard code are involved. The result of this testing should be confidence in the cutover process for your environment.

Technical workstream process

> **NOTE**
> For the technical workstream, the cutover testing process is the same as the high-level steps of the actual/go-live cutover process.

For the technical workstream, the cutover testing process is the same described in Upgrade from AX 2012 - Data upgrade in sandbox environments.

Functional workstream

After data upgrade, several configuration tasks will be required in the new environment. The goal of this workstream is to document and quantify all configuration tasks, and to assign a resource to each task, to help guarantee that these tasks can be done together with the technical workstream during the downtime window.

Typically, functional tasks involve changing the values of specific system parameters or other configuration data. These tasks are identified through the full functional test pass, which is a separate activity from the cutover testing. When a task of this type is identified, it should be reviewed together with the functional resource and your developer.

Larger changes might require that a new custom data upgrade script be written to update the data during the data upgrade process. However, the functional resource can manually run smaller changes through the new system after data upgrade.

Larger changes that have new data upgrade scripts must be tested. Therefore, one or more additional iterations of the MajorVersionDataUpgrade.zip package will have to be run. It's important that you weigh the cost of running the package again against the cost of manual data entry.

For each manual change, a task must be added to the cutover plan document. This task must show the following details:

- What is the task, and what must be done?
- Who must do it?
- How long does it take?

Add users, and perform functionl tests

When you have fully configured your environment, add users, and perform appropriate testing.

Roll back to AX 2012

The goal of this task is to restore the database by using the backup that was made when AX 2012 was turned off, and then turn AX 2012 back on. The state of integrated systems might also have to be restored. However, because integrated systems vary from business to business, you must plan for this scenario independently, based on your specific circumstances. Although it's unlikely that you will have to roll back, it's very important that you have a tested process in case you require it.

Upgrade from AX 2012 - Post-upgrade tasks

2/18/2021 • 3 minutes to read • Edit Online

IMPORTANT

Upgrade is currently only supported from either Dynamics AX 2012 R2 or Dynamics AX 2012 R3. For each release, please update to the latest available cumulative update before upgrading to latest Finance and Operations application release.

This topic describes the tasks that you might have to perform in Finance and Operations apps, like Dynamics 365 Finance and Dynamics 365 Supply Chain Management, after you complete a code and data upgrade from Microsoft Dynamics AX 2012. A process data package (PDP) that is available in Microsoft Dynamics Lifecycle Services (LCS) includes links to the following menu items. This PDP will fill in the **Data validation checklist** workspace. The **Data validation checklist** workspace lets users track a project and monitor the tasks that are required in order to complete it.

Document management

If you use document management, existing documents or attachments that are stored in the database should be migrated to Microsoft Azure Blob storage. To complete this migration, use the **Migrate files** button on the **Migrate files** tab on the **Document management parameters** page. This operation is not critical as document management can still access file stored in the database, but the files can take considerable database storage and the retrieval is less efficient. The file migration process will migrate all possible database files to Microsoft Azure Blob storage, reporting on any failures and continuing. If any errors are reported, attempt running the file migration process again.

If the file migration process isn't able to complete without failure, this may be that the files stored in the database are corrupt, which Microsoft is unable to repair. If this is the case, you can request a non-business critical support case be opened to enable conversion of the attachments into note records, which will retain any previous notes as well as the names of the files that were stored in the database. Note that the files themselves cannot be recovered.

Print management

If you use Print management, the references to network printers from AX 2012 won't be valid. You must set up and reference network printers on the **Document routing** page. For more information, see Install the Document Routing Agent to enable network printing.

Commerce

After you complete the upgrade from AX 2012, you must configure registers and devices.

To configure a register, click **Retail and Commerce** > **Channels** > **Stores**. Select the row for the channel, and then expand the **Registers** FactBox. Click **More**, click **New**, and complete the setup of the register.

To configure a device, click **Retail and Commerce** > **Channel setup** > **POS Setup**, and then click **New**.

Additionally, you must run all jobs (9999) for the channel database. Click **Retail and Commerce** > **Headquarters setup** > **Commerce scheduler** > **Channel database**. Select the row for the appropriate channel database, and then click **Full data sync**. Select the **9999** (**All jobs**) distribution schedule, and then click **OK**. Click **OK** again to run the job.

Service industries

After you complete the upgrade from AX 2012, you must set up resource capacity roll-up and project ledger intercompany posting.

To run the Resource capacity roll-up batch job, click **Project management and accounting** > **Inquiries and reports** > **Capacity synchronization**. You must run this batch job to set up the resource and resource calendar reservation data. This data will be required if you use project resource scheduling. For more information, see Project resourcing.

To enable project ledger intercompany posting, click **Project management and accounting** > **Setup** > **Posting** > **Ledger posting setup**. On the **Cost accounts** tab, in the **Ledger account types** field, select **Intercompany cost**, and then enter the details of the lending legal entity. On the **Revenue accounts** tab, in the **Ledger account types** field, select **Intercompany revenue**, and then enter details of the borrowing legal entity.

Budget planning

After you complete the upgrade from AX 2012, you must set up Budget planning columns and layouts. To complete this setup, click **Budgeting** > **Setup** > **Budget planning** > **Budget planning configuration**.

Additionally, you must update Budget planning processes so that they use the appropriate layout for each budget stage. To update Budget planning processes, click **Budgeting** > **Setup** > **Budget planning** > **Budget planning process**.

For more information about Budget planning upgrade, see Upgrade budget planning.

Upgrade from AX 2012 - Functional test passes

2/18/2021 • 2 minutes to read • Edit Online

IMPORTANT

Upgrade is currently only supported from either Dynamics AX 2012 R2 or Dynamics AX 2012 R3. For each release, please update to the latest available cumulative update before upgrading to latest Finance and Operations application release.

After data upgrade is completed, we recommend that you complete a full functional test pass of all business processes. In a full functional test pass, you do an extensive retest of all business processes that are performed by using Finance and Operations. Tests should include both processes that have been brought forward from Microsoft Dynamics AX 2012 and new processes that use features from Finance and Operations.

Depending on your code quality, bug remediation and retesting might require several iterations of the functional test pass. After a bug is fixed, take care to retest all processes that are involved, to make sure that no downstream or upstream processes are affected by the change.

Old data vs. new data

As you create a list of tests to perform, consider the impact of old data versus new data. This type of testing will help you uncover data-related bugs. The code that created the old and new data might be very different, and this difference can be a common root cause of bugs.

For example, if the business process that you're testing is cancellation of a purchase order, can you cancel a purchase order that was started before it was upgraded to the new system? Can you also cancel a purchase order that was started after the upgrade to the new system?

We used a very simple example here, but the testing requirement can be more complex, because many business processes in the system are interconnected, and the effect of old data versus new data effect is cumulative.

For example, here are the stages of a production test flow:

1. The item master is designed and released to a legal entity.
2. Item requirements are created.
3. Production orders are generated.
4. Purchase orders are generated.
5. Production order processing occurs (shop floor).
6. Vendor payment (payment of purchase orders) occurs, and so on.

In this production test flow, each stage can be performed by using either new records or old records as input. The result is a matrix of tests that covers every combination of old and new data. For some processes, test matrices might seem excessive, and they might actually be excessive in practice. Therefore, you can decide to focus on certain combinations that you predict will be used the most. However, it's still helpful for you to know what you aren't covering. Make a conscious decision, where you know what you have, what you're going to focus most of the testing on, and what you're not going to focus on.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Upgrade from AX 2012 - Prepare for go-live

2/18/2021 • 2 minutes to read • Edit Online

> **IMPORTANT**
> Upgrade is currently only supported from either Dynamics AX 2012 R2 or Dynamics AX 2012 R3. For each release, please update to the latest available cumulative update before upgrading to latest Finance and Operations application release.

As the go-live date approaches, it's important that you implement this series of steps to help ensure that the source Microsoft Dynamics AX 2012 system and the upgrade process both remain stable and consistent for go-live.

As part of this process, you must lock down any further code changes or application setup changes before you run the final cutover test.

Code freeze

All code changes in the AX 2012 environment should be frozen. Implement an escalation process to handle any critical issues that appear in AX 2012. By default, any new code changes that are required should be implemented only in the new system, not in the AX 2012 environment. Implementation of proposed code changes in the AX 2012 environment should be discussed at the management level. If a code change is made in AX 2012, the same change must be made in the new system. In that case, another iteration of cutover testing and functional testing might be required.

Application configuration freeze

All application configuration changes should be frozen in the AX 2012 environment, because these changes could affect how the new system behaves or how the data upgrade scripts behave. Configuration changes are changes in the AX 2012 application that are related to the configuration of system functionality. By freezing these changes, you help guarantee the stability of the data upgrade process.

Because configuration changes are typically controlled by the AX 2012 system administrator or a small group of trusted super users, we don't recommend that you enforce the freeze by changing security access. Instead, implement the freeze through a business process that is communicated to those users. Changes to security access might require a code change (changes to the role definitions themselves) or a configuration change (reassignment of users), and these changes could affect the upgrade process.

Running the final cutover test

After no further code or setup changes will occur, run a final cutover test to make sure that all data and code upgrade tasks still run as expected.

> **NOTE**
> You must complete this step even if you freeze code and setup changes at the beginning of the upgrade project, because the data itself changes every day. This final cutover test also validates that the current data is upgraded successfully.

Make sure that functional testing is performed against this last upgraded copy.

At this point in the upgrade project, we recommend that you categorize any bugs that are found:

- **Blocking** – The upgrade project can't proceed until every bug of this type is fixed. The upgrade must be postponed, and can proceed only after the bug is remediated and the cutover test is run again. For a bug to be classified as blocking, it must meet these conditions:
  - It prevents a critical business process from being completed.
  - No workaround or mitigation is available for it.
- **Non-blocking** – The upgrade project can proceed. Bugs of this type can be fixed in the upgraded system.

Upgrade from AX 2012 - Go live (Cutover)

2/18/2021 • 3 minutes to read • Edit Online

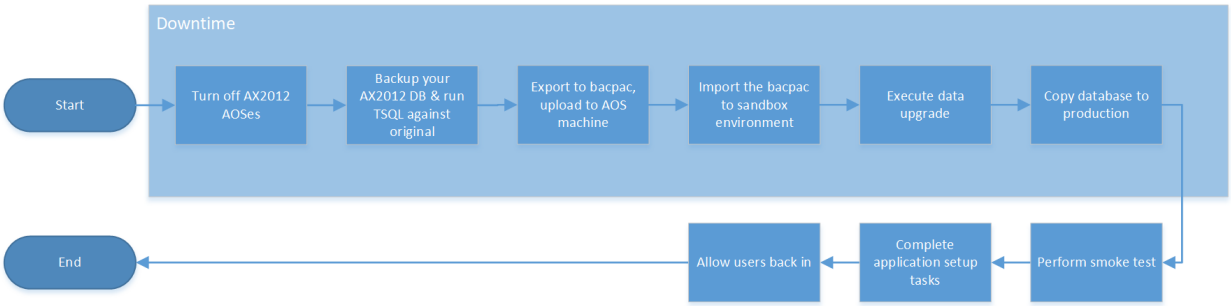
IMPORTANT

Upgrade is currently only supported from either Dynamics AX 2012 R2 or Dynamics AX 2012 R3. For each release, please update to the latest available cumulative update before upgrading to latest Finance and Operations application release.

After you have successfully completed upgrade testing in a Standard or Premier Acceptance Test environment (Sandbox Tier 2 or higher), and you have also completed a successful test cutover, the time has arrived to upgrade your production environment and go live.

Cutover is the term that we use for the final process of getting a new system live. This cutover process consists of the tasks that occur after Microsoft Dynamics AX 2012 is turned off but before Finance and Operations is turned on. Before you plan your final cutover, you need to successfully complete one successful mock cutover as described in Cutover testing

The following illustration shows the overall process for cutover to go-live as it will occur in the production environment.



NOTE

In this topic, we use the term *sandbox* to refer to a Standard or Premier Acceptance Testing (Tier 2 or 3) or higher environment connected to a SQL Azure database.

Overall process

The high-level steps of the production environment upgrade process are the same as the Mock cutover process, refer to Upgrade from AX 2012 - Cutover testing (Mock cutover) for detailed instructions.

1. Turn off the AX 2012 AOS instances
2. Create a copy of the AX 2012 database. We strongly recommend that you use a copy, because you must delete some objects in the copy that will be exported.
3. Export the copied database to a bacpac file by using a free SQL Server tool that is named SQLPackage.exe. This tool provides a special type of database backup that can be imported into SQL Database.
4. Upload the bacpac file to Azure storage.
5. Download the bacpac file to the Application Object Server (AOS) machine in the sandbox environment, and then import it by using SQLPackage.exe. You must then run a script against the imported database to reset the SQL database users.
6. Run the appropriate data upgrade package against the imported database.

Prerequisites

Before you can perform an upgrade in the production environment the following prerequisites must be met:

- Complete the code upgrade and data upgrade in a sandbox environment and successfully completed a functional test pass
- Deploy the production environment. Before the option to request the deployment of the production environment lights up you must have completed:
  - The Subscription estimator in LCS. We use this to help us size your production environment because it provides details of the throughput you'll require.
  - The Test phase of the methodology in LCS. This is to help ensure that you're at the stage in your project where you're ready to start testing in the production environment.
  - After a request is submitted to Microsoft to deploy the production environment, it will take roughly 24 hours to deploy, so ensure that you leave enough time for this to happen.
- Apply all necessary updates and customizations (AOT deployable packages) to the production environment. There should not be any code change after signing off on a Mock cutover.
- To schedule an upgrade, request a timeslot with the DSE team by submitting "Other" type service requests from LCS as described in the Upgrade from AX 2012 - Cutover testing (Mock cutover). This is to ensure that the preferred timeslots will be available for you. Be aware that there is significantly higher demand for slots during the weekend, so requesting these as far in advance as possible will help attain your preferred schedule.

Related articles

- Onboarding
- Submit service requests to the Dynamics Service Engineering team.
- Upgrade from AX 2012 - Cutover testing (Mock cutover)

Make the chart of accounts delimiter unique

2/18/2021 • 2 minutes to read • Edit Online

In Microsoft Dynamics AX 2012, you could use the same delimiter for your chart of accounts and dimension values. In current versions of Finance and Operations, you cannot have the same delimiter for the chart of accounts and dimension values. If there is a duplicate delimiter, you can change it after upgrade.

This feature is available in the following versions:

- Finance and Operations version 8.0
- Finance and Operations version 7.1, KB 4094701 Cannot enter the financial dimensions when the dimension values contain the chart of accounts delimiter
- Finance and Operations version 7.2, KB 4092967 Cannot choose sub-project as dimension when sub-project format contains the dimension delimiter

Update delimiter

If there is a conflict with the chart of accounts, the chart of accounts delimiter and the project/subproject ID format can be changed. No other dimension delimiters can be changed.

- You can change the chart of accounts delimiter after upgrade in **General ledger parameters** > **Chart of accounts and dimensions** > **Change delimiter**.
- If the only conflict is with the project/subproject ID format, you can change that value in **Project management and accounting parameters** > **General** > **Modify subproject format**.

How to determine if your environment requires updated delimiters

If delimiters in your upgraded environment are conflicting, you may experience instability when entering values in a segmented entry control or dimension entry control. This means that you will need to always use lookups or a flyout menu when entering account and dimension combinations.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Upgrade single-voucher journals and currency revaluations

2/18/2021 • 2 minutes to read • Edit Online

Some organizations enter journals that contain a single voucher that has more than one customer or vendor, and they also run the Accounts receivable or Accounts payable foreign currency revaluation process. This topic describes the steps that these organizations should follow when they upgrade to Microsoft Dynamics 365 for Operations version 1611.

Follow these steps when you upgrade to Microsoft Dynamics 365 for Operations version 1611.

1. Before you upgrade to Finance and Operations, run the foreign currency revaluation processes for Accounts receivable and Accounts payable. Set the **Method** field to **Invoice date**. A revaluation transaction is created that reverses the last foreign currency revaluation. Therefore, the open transactions are valued at their original accounting currency.
2. Upgrade to version 1611.
3. Run the Accounts receivable and Accounts payable foreign currency revaluation processes again. This time, set the **Method** field to **Standard**. A new revaluation transaction is created that is based on the current exchange rates. This transaction records the unrealized gain/loss and the correct summary ledger account.

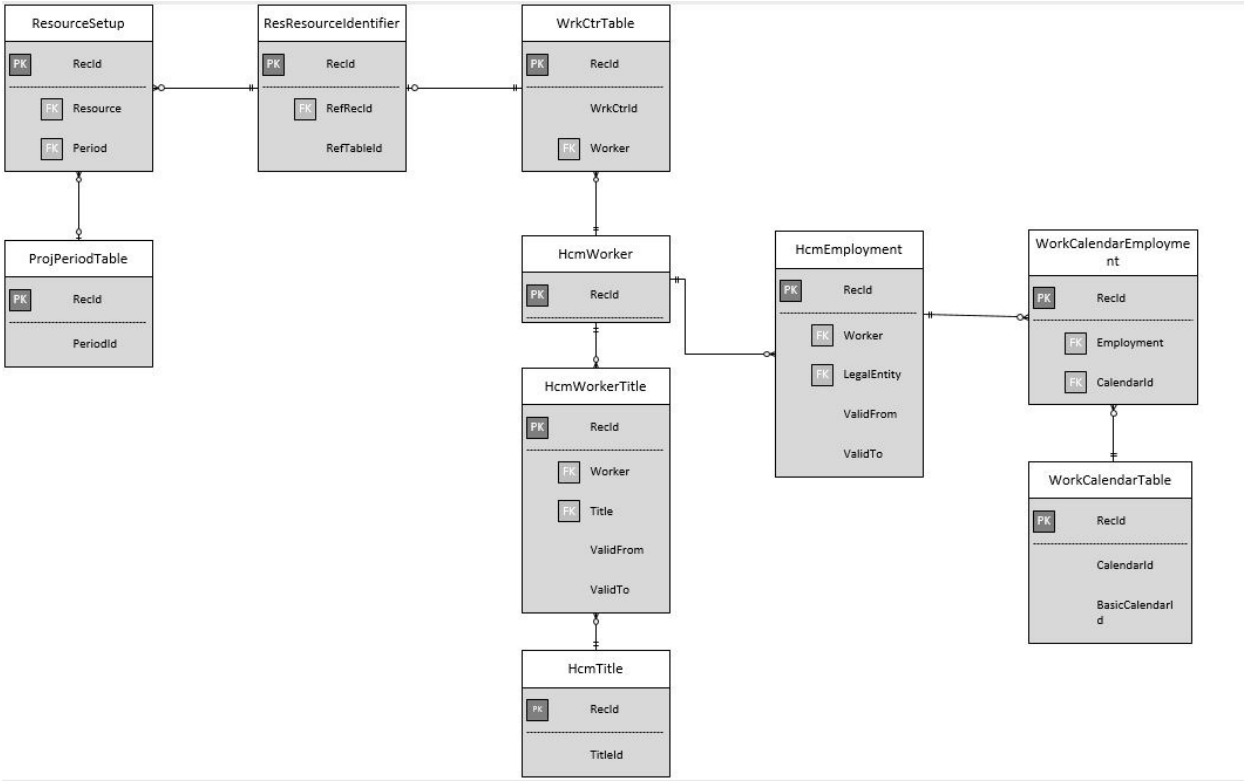
Project resource scheduling data model

2/18/2021 • 7 minutes to read • Edit Online

This topic provides information about the Project resource scheduling data model.

Physical data model for Project resource scheduling

The following diagram represents the data design structure of the Project resource scheduling physical data model.



Tables

The following table provides a list of additional tables that support the Resource management data model.

TABLE	DESCRIPTION
ResResourceIdentifier	Stores all resources and a subset of records from the **WrkCtrTable** table that are identified as resources. When a resource is added, a record will be added to the **WrkCtrTable **table. A record will also be added to this table with a Foreign Key reference to the new record in the **WrkCtrTable** table.
ResourceSetup	Specifies a resource's property. This table replaced the **ProjWorkerSetup** table in Dynamics AX 2012.
ResBooking	Stores all resource booking type reservations. This table replaced some of the information stored in the **PSASchedEmplReservation** table in Dynamics AX 2012.

TABLE	DESCRIPTION
ResAssignment	Stores all resource assignment type reservations. This table replaced some of the information stored in the **PSASchedEmplReservation** table in Dynamics AX 2012.
ResourceResourceCategorySetup	Stores the resource default project role association that is validated by a time range. This is a time effective table which has a Foreign Key to **ResResourceIdentifier.RecId** and a Foreign Key to **PSASchedRole.RecId** with **ValidFrom** and **ValidTo** fields. The **PSASchedRole** table is used to store the project role.
ResCalendarCapacity	This is a data denormalized table based on **WorkCalendarTable**, **WorkCalendarDate**, and **WorkCalendarDateLine** to calculate resource capacity. The calculation is based on the resource associated calendar date line definition, which is specified in the **WorkCalendarDataLine** table. The data in this table can be generated by running the **Synchronize resource capacity roll-ups** batch job.
ResRollUpCalendar	This is a data denormalized table based on **ResCalendarCapacity**. The data in this table is used by the **ResRollUpWriter** class to speed up the process of adding resource records to the **ResRollup** table. The data in this table can be generated by running the **Synchronize resource capacity roll-ups** batch job.
ResRollUp	This is a data denormalized table based on **ResBooking**, **ResAssignment**, and **ResRollUpCalendar** for performance improvement. This table contains all of the capacity and reservation data for time scales, including Day, Week, Month, Quarter, and Half year, for all resources. This data is accessed by the **AvailabilityView** control, which is included with any resource scheduling X++ form that includes this control, like the **Resource availability** form. The data in this table can be generated by running the **Synchronize resource capacity roll-ups** batch job. Data is updated for every resource reservation action such as adding a new resource, booking, assignment, and reservation cancellation.

Views

The following table provides a list of the most informative views that you can use to access the Resource management data model.

VIEW	DESCRIPTION
ResResourceWorkCenterView	This view is based on **ResResourceIdentifier** joining to **WrkCtrTable**, which captures only the fields that are required for resource management.
ResourceView	This view is based on **ResResourceIdentifier**, which is only based on **ResResourceWorkCenterView**. In a future release it can be extended to more than **ResResourceWorkCenterView** if a resource record will be saved on more than just **ResResourceWorkCenterView**.

VIEW	DESCRIPTION
ResourceWorkerView	This view replaces the **HcmWorker** table, which is based on **ResResourceIdentifier**. **WrkCtrTable** has a worker field which is a Foreign Key to **HcmWorkerTable**. This view includes all **WrkCtrTable** resource which worker field is not 0.
ResourceLegalEntityView	This view replaces the **HcmEmployment** table. This is the resource legal entity (LE) view, which is a union of **ResourceWorkerLegalEntityView** and **ResourceOtherLegalEntityView**. The purpose of this view is to look up the resource legal entity as well as the associated valid from and valid to date. For example, an **HcmWorker** resource can be hired by multiple LEs with different ValidFrom and ValidTo dates. If the worker is hired by multiple LEs, then this worker resource will have multiple records in this view. If the resource is not a **HcmWorker**, then this resource will have only one record and **validFrom** = **minDate** and **ValidTo** = **maxDate**.
ResourceCalendarView	This view replaces **WorkCalendarTable**. This view contains the calendar **RecId** of the resource. This is a union view of **ResourceWorkerCalendarView**, R**esourceOtherCalendarView**, and **ResourceGroupResourceCalendarView**. To look up the resource calendar, users should first look up the **ResourceLegalEntityview**. This is because a resource should have a different calendar depending on the resource's legal entity. First, locate the record from **ResourceLegalEntityView** and get the **RefRecId** and **RefTableId** fields. Then, look up **ResourceCalendarView** by setting **ResourceLegalEntityRefRecId** = **ResourceLegalView.RefRecId** and R**esourceLegalEntityRefTableId** = **ResourceLegalView.RefTableId**.
ResourceCategoryView	This view shows the resource category (the resource role for a worker resource), which is based on the **PSASchedRole** table.
ResSkillHcmView	This view is based on **HcmSkill** to capture all the skills that are defined in the **HcmSkills** table.
ResCertificateHcmView	This view is based on **HcmCertificateType** to capture all the certificates that are defined on the **HcmCertificateType** table.
ResEducationHcmView	This view is based on **HcmEducationDiscipline** to capture all the education disciplines that are defined in the **HcmEducationDiscipline** table.
ResProjectHcmView	This view is based on **ProjTable** to capture all projects that exist in the **ProjTable**.
ResRoleView	This view is based on **ResourceCategoryView** to capture all resource categories.

VIEW	DESCRIPTION
ResCharacteristicView	This view is the union view of **ResSkillHcmView**, **ResCertificateHcmView**, **ResEducationHcmView**, **ResProjectHcmView**, and **ResRoleView**. This view is used to gather all the characteristics that are defined as selectable criteria for the resource search feature.
ResResourceCharacteristicView	This view is the union view of **ResResourceCharacteristicHcmView**, **ResResourceResourceCategoryView**, **ResResourceNameView**, and **ResResourceRoleCharacteristicView**. This view is used to identify the characteristics that are associated with a resource for the resource search feature.
ResCalendarCapacityView	This view is based on **ResCalendarCapacity**, which contains calendars that are specified in **WorkCalendarTable** with capacity details.
ResCapacityView	This view shows the resource's capacity per hour.
ResResourceCapacityWorkDaysView	This view is based on **ResCapacityView**, which shows the resource capacity per day.
ResAssignmentView	This view is based on the table **ResAssignment**, which stores the activity resource assignments to either project or quotation work breakdown structure (WBS) task.
ResBookingView	This view is based on the table **ResBooking**, which stores the activity resource bookings to a project or quotation.

Changes that will affect tables and fields

These sections contain information regarding code changes to tables and fields that are part of the feature implementation related to Project resource scheduling.

Resource scheduling

The table **PSASchedEmplReservation** is no longer used to store a resource's reservations. Instead, reservations are stored in the **ResAssignment** and **ResBooking **tables. Both tables use the **Activity resource** field Foreign Key for **PSAProjSchedRole.RecId** to store a resource's reservation. The **PSAProjSchedRole** table is the project team table that has the **Resource** field Foreign Key to **ResourceView.RecId** and the **ResourceLegalEntity** field Foreign Key to **CompanyInfo.RecId** to identify which resources are the project's or quotation's team members. If the **PSAProjSchedRole.Resource** field = 0, then this activity resource is a planned resource. A planned resource is a shadow resource that is not backed by an actual resource. **PSAProjSchedRole.ResourceCategory** is a Foreign Key to **PSASchedRole** that stores the role of this team member. The **ResourceResourceCategorySetup** table stores the default time effective resource/role association. However, the resource can be reserved to any role defined by **PSAProjSchedRole.ResourceCategory,** ignoring the default role definition on the **ResourceResourceCategorySetup** table. Regarding WBS versioning, the tables **ProjPlanVersions** and **ProjPlanVersionDetails** store the WBS tasks versions. Initially, all WBS task data will be stored on these tables while the user is editing the WBS tasks content. After the user clicks the **Publish** button, the task data will be pushed to the original hierarchy tables (**smmActivities** and **PSAActivitySetup**). The resource management feature requires data in the original hierarchy tables and requires a published WBS.

Price by resource and resource category

The pricing tables, **ProjCostPriceExpense**, **ProjCostSalesPrice**, and **ProjRevenueSalesPrice** have a Foreign Key from **ResourceView.RecId**. The tables **ProjHourCostPrice**, **ProjHourSalesPrice**, and **ProjTransferPrice** have a Foreign Key from **ResourceView.RecId**. The field **ResourceCategory** is part of the Foreign Key for **ResourceCategoryView.RecId**. This is done so that pricing is based on a resource instead of a worker and enables pricing setup by resource category.

Common methods to get resource field and lookup resources

There are several resource methods in the **ResourceFacade** class. The following table includes some of the most common methods.

CLASS.METHOD	DESCRIPTION
ResourceFacade.findByWorker()	Look up the resource record ID by **HcmWorker** record ID.
ResourceFacade.findOrCreateByWorker()	Look up the resource record ID by **HcmWorker** record ID. If the resource record ID is not found, then a resource record ID will be added to the **ResResourceIdentifier** table and the resource record ID will be returned.
ResourceFacade.findByResourceID()	Look up the resource record ID by resource ID.
ResourceFacade.get...	There are many get methods supported for the resource in the **ResourceFacade** class. Resource related values like **Resource ID**, **Resource calendar**, **Resource legal entity**, and **Resource period** can be queried from the **ResourceView**, **ResourceLegalEntityView**, **ResourceCalendarView**, and **ResourceSetup** tables.

Facade classes

The following table lists the facade classes that you can use as a starting point to interact with the Resource management data model.

CLASS	DESCRIPTION
ActivityFacade	Contains the methods to retrieve an activity's properties, including **ID**, **Quotation ID**, **Project ID**, **Booked/assigned capacity**, **Remaining capacity**, **Calendar**, **Activity number**, and **Root project activity**.
ActivityResourceFacade	Contains the methods to retrieve an activity's resource properties including **Calendar**, **Name**, **Resource category**, **Resource legal entity**, **Is generic resource**, and **Is team member**.
PeriodFacade	Contains the methods to retrieve period properties including **Start and End date**, **Period ID**, and **Period name**.
ResourceCalendarFacade	Contains the methods to retrieve the resource's calendar properties including **Calendar data area ID**, **Calendar ID**, **Calendar RecID**, **Capacity**, and **Dates**.
ResourceCategoryFacade	Contains the methods to retrieve the resource's default resource **Category**, **ID**, **Name**, and **Type**.

CLASS	DESCRIPTION
ResourcePeriodFacade	Contains the methods to retrieve the resource's period date range and update the period method.
ResourceWorkerFacade	Contains the methods to retrieve the **Employment** type.

Workflow subsystem updates in Finance and Operations

2/18/2021 • 2 minutes to read • Edit Online

This topic reviews the workflow system in Finance and Operations. It describes the changes that have been implemented since Microsoft Dynamics AX 2012 and also includes links to more information about the workflow system.

The workflow system in Finance and Operations will be familiar to you if you've used Dynamics AX 2012. For more information about the workflow subsystem in Dynamics AX 2012, see the following topics.

TO LEARN ABOUT THIS SUBJECT	SEE THIS TOPIC
The workflow system	https://technet.microsoft.com/library/dd309672.aspx
Workflow types by module	https://technet.microsoft.com/library/dd362043.aspx
Workflow elements	https://technet.microsoft.com/library/dd309626.aspx
Workflow actions	https://technet.microsoft.com/library/dd362144.aspx
Workflow participants	https://technet.microsoft.com/library/dd309598.aspx
Workflow examples	https://technet.microsoft.com/library/dd309636.aspx
Developing a workflow	https://msdn.microsoft.com/library/cc967389.aspx
Implementing a workflow	https://msdn.microsoft.com/library/cc585061.aspx

Primary changes to the workflow system

Here are the primary changes that have been implemented in Finance and Operations:

- Integration with the new Application State Machine feature enables workflow events to be bound to state transitions on the underlying entity's state machine. This binding enables business logic to be centralized within the state machine and also enables the workflow system to be a declarative consumer of that state machine. The workflow metadata can reference a state transition that is performed when a specific workflow event occurs. Therefore, you can do state transitions within a workflow without writing any additional code.
- The workflow editor is now a program that you click one time to download. The editor communicates with Finance and Operations by using services, which means that you can carry forward the rich, graphical workflow design experience from Dynamics AX 2012.
- Workflow development wizards have been ported into Microsoft Visual Studio.

Additional resources

Technical Concepts Guide for Developers

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Transition from Analysis Services cubes to aggregate models

2/18/2021 • 2 minutes to read • Edit Online

This topic explains how in-memory, real-time aggregate models are used for analytics, and why we transitioned from using Server Analysis Services (SSAS) cubes.

The world is moving to real-time, proactive analytics. Reporting and trending on historical data is being replaced by up-to-the-second visualizations and proactive guidance. In-memory, real-time aggregate models now replace the perspectives that were previously used for analytics.

A historical look at perspectives and cubes

We envision embedded insights playing a key role in the Finance and Operations user experience. This vision has driven us to invest in building analytic capabilities within the product. In Dynamics AX 4.0, we introduced the concept of *perspectives*. The objective was to present a simpler view of the ERP schema, specifically modeled for reporting. This simpler view was referred to as perspectives. In Dynamics AX 4.0, the system generated reporting models (SMDL models) that enabled you to create ad-hoc reports with SQL Server Report Builder. In Dynamics AX 2009, we added the capability to generate SQL Server Analysis Services (SSAS) projects using metadata definitions in perspectives. These projects become cubes when deployed to an SSAS server. In Dynamics AX 2012, we improved modeling in perspectives and improved tooling support for managing the lifecycle of SSAS projects. You could use Excel, as well as Power View, to explore data and create reports with cubes in Dynamics AX 2012. The SMDL technology was also deprecated. In Dynamics AX 2012, we stopped generating SMDL models.

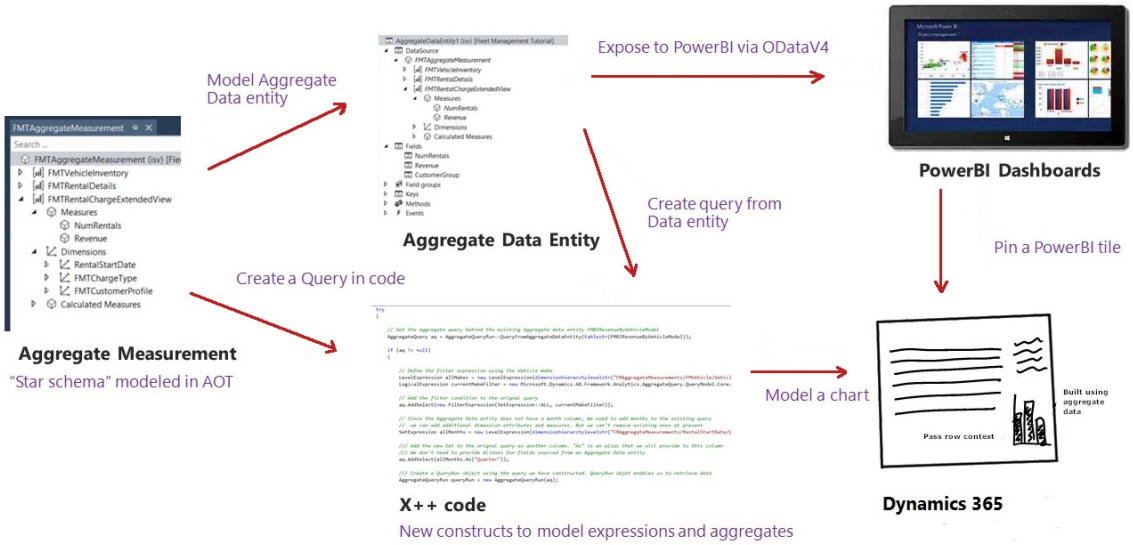
How perspectives are used now

As a developer, your "contract" with the system was a perspective. The system generated "stuff" to help you achieve your end goal. In Dynamics AX 2012, the "stuff" that was generated was SSAS projects. So the contract between you (the developer) and the system (the BI framework), was as follows:

- You modeled perspectives.
- The system generated the "stuff" needed to enable you to build visuals and reports.

Perspectives are now modeled using add-ins for Visual Studio. (Visual Studio is now the development environment.) Perspectives are comprised of *aggregate measurements* and *aggregate dimensions*. As a developer, you have the ability to model simpler schemas for answering business questions using aggregate measurements and aggregate dimensions. Aggregate measurements can be used to define data entities (called *aggregate data entities*) which can be directly bound to Finance and Operations forms as a data source. Aggregate data entities can also be used to

- Expose data to PowerBI.
- Access data programmatically using the AXQuery object.



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

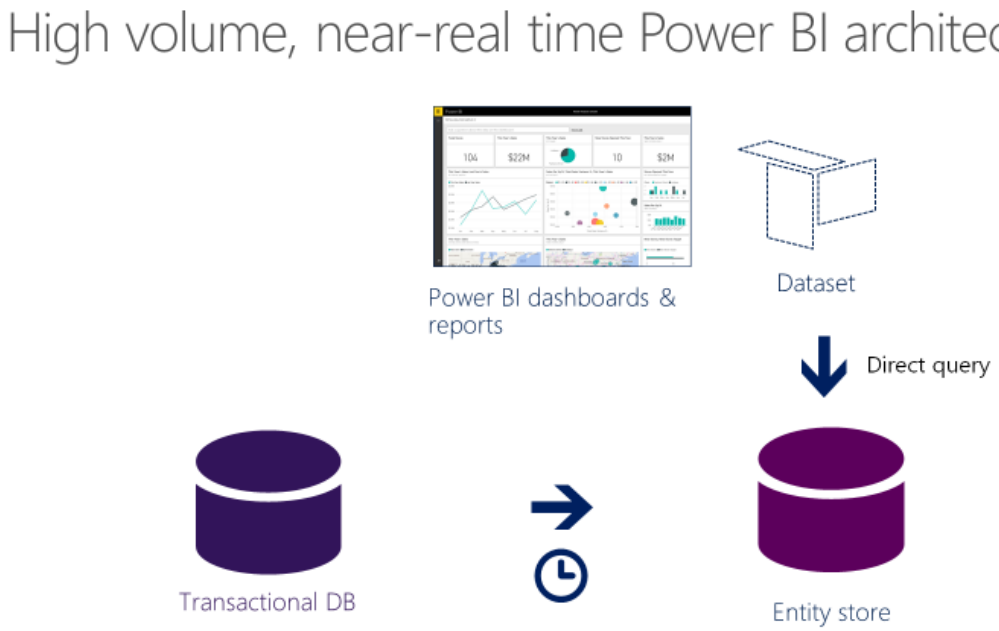
The survey will take about seven minutes. No personal data is collected (privacy statement).

Migrate upgraded AX 2012 R3 sales cubes to the entity store

2/18/2021 • 13 minutes to read • Edit Online

In this tutorial, you'll migrate an upgraded Microsoft Dynamics AX 2012 R3 cube schema to the entity store in a Finance and Operations application. You'll use the sales cube that was included in Dynamics AX 2012 R3 as an example.

The entity store will support near real-time Microsoft Power BI integration scenarios, as shown in the following diagram. For an overview of Power BI integration with entity store, see Power BI integration with entity store.



New Power BI features included in the May 2016 and November 2016 updates

This tutorial requires the Dynamics 365 for Operations May 2016 update or later. You will use the following new capabilities in this tutorial:

- Stage an aggregate measurement in the entity store and refresh the data from Dynamics AX. You might prefer this option over in-memory real time aggregate measurements when:
  - You upgrade a Dynamics AX 2012 cube.
  - Your aggregate measurements are very large.
  - Data freshness (latency) from a few minutes up to a few hours is acceptable for reporting.
- Use the batch framework to schedule a recurring refresh. For this release, only a full refresh is enabled.
- Create reports using Power BI desktop in a developer/test environment.
- Leverage the direct query option when creating Power BI content. For example, you can create larger models without relying on OData as the data refresh mechanism.
- Migrate reports from your development environment to a production environment using Lifecycle Services (LCS).
- As a partner or an ISV you can distribute Power BI content as part of an LCS solution to your customers.
- **If you're using the November update (platform release 1611)** or later, some steps in this document

are part of the process to refresh the entity store - you do not need to perform them manually.

Change upgraded aggregate measurement properties

As part of the code upgrade process, analysis services projects from the Application Object Tree (AOT) in Dynamics AX 2012 can be migrated to the new aggregate measurements metadata format.

1. Launch Visual Studio and create a new project in Application Suite.

   > **NOTE**
   > You can create a model and include the customized aggregate measurement within that model. For more information, see Customize through extension and overlayering.

2. Open Application Explorer. Go to **Analytics** > **Perspectives** > **Aggregate measurements**. You will notice a set of aggregate measurements that were upgraded from Dynamics AX 2012 R3, as well as the measurements that ship in the current version.

3. Select **SalesCube**. Right-click and select **Duplicate in project**.

4. An aggregate measurement with the name **SalesCubeCopy** will be added to the project.

5. Rename this measurement. Select **SalesCubeCopy** in Solution Explorer. Right-click and select **Rename**. Enter **SalesCubeV2** as the new name.

6. Double-click **SalesCubeV2** to launch the Aggregate measurement designer. Notice the structure of the aggregate measurement that was migrated from Dynamics AX 2012.

7. The Sales cube in Dynamics AX 2012 encompassed a broad subject area related to Sales. In this case, let's create a smaller, more focused Power BI model using the metadata that was upgraded. Expand the **Sales Order Lines** measure group and review the list of measures and dimension references.

   > **NOTE**
   > Leveraging the modeling capabilities you can quickly make a few enhancements to this model. Suggestions for improvements:
   >
   > - Replace views/tables that have been used to model the measure group (and/or dimensions) with an entity. You can model an entity using the underlying view and replace the view with the corresponding entity. This will enable you to leverage upcoming features such as incremental refresh and security.
   > - Remove unwanted dimension references by adding the corresponding field to the attributes node. For example, the Sizes dimension reference can be removed because the **Size** field in the measure group is sufficiently descriptive. This will improve the runtime performance of queries as well as refresh times.

8. Select the **SalesCubeV2** root node in the Aggregate measurement designer. Right-click and select **Properties**.

9. During upgrade, aggregate measurements are set to the legacy property flag, **SSASCube**. You need to change this property to one of two supported usage types. Previously, **InMemoryRealTime** was supported as usage for aggregate measurements. **StagedEntityStore** is supported as a new usage type.

   > **NOTE**
   > Modify the usage property to InMemoryRealTime if you plan to use the Aggregate measurement for embedded BI scenarios as well as Power BI integration. If you are using the Aggregate measurement only for Power BI or Cortana Intelligence Suite integration, select **StagedEntityStore**.

10. Save the project. Right-click the project in Solution Explorer and select **Rebuild**.
11. After the rebuild operation is finished, save the project, and then close Visual Studio. This completes the development work. You will author reports as a report developer or a power user.

Refresh the entity store

As an administrator you can configure the refresh of the aggregate measurement using the client.

1. Launch the Dynamics AX client and navigate to **System Administration** > **Setup** > **Entity Store**. The **Entity Store** form shows a list of aggregate measurements that are available for deployment to the entity store.
2. Notice that **Sales Cube** (which was upgraded from Dynamics AX 2012) is not available for deployment to the entity store. **SalesCubeV2**, which you created in the previous step, can be deployed to the entity store.
3. Select **SalesCubeV2** from the list, and click the **Refresh** button. The **Refresh** dialog box will display. Expand the **Run in the background** tab.
4. Provide a descriptive name in the **Task description** field. Optionally, you can select the **Recurrence** tab and create a recurring schedule instead of a one-time refresh. Click **OK**.
5. The system will create a batch job for refresh of the aggregate measurement in the entity store.

Authoring a report on Sales by State with Power BI desktop

This step requires that you the install Power BI desktop tool that can be downloaded from Microsoft Power BI Desktop.

1. Launch Power BI desktop. You may need to apply updates. A welcome page will display. Click **Get data**.
2. Alternatively, when Power BI desktop launches, on the **Home** tab select **Get Data** > **SQL Server**.
3. In the **SQL Server Database** dialog box, enter the server name and the name of the entity store database. If you deployed a developer environment, you can enter "." as the server name and **AxDW** as the database name. If you are working in a test environment, you need to get these parameters from your system administrator
4. Select the **DirectQuery** option. In this exercise, you will create Power BI reports that are executed directly on the entity store. If you had used the **Import** option, Power BI would cache data from the entity store and you would need to periodically refresh the Power BI model. **Import mode is currently not supported with reports written using entity store**. Click **OK**.
5. Next you will see the **Navigator** dialog box. Navigator enables you to select tables and views from the entity store that you want to report on. Enter **Sales** in the search box. The system will filter entities that are related to the **SalesCubeV2** aggregate measurement that was previously created.

   > **NOTE**
   > The entity store stages the aggregate measurements that have been created. While entities within each aggregate measurement are prefixed and stored as individual tables, Power BI desktop enables you to combine data from multiple aggregate measurements.

6. You will create a report that shows sales by state. Select **SalesCubeV2_Customer** and **SalesCubeV2_CustomerInvoices** from Navigator and click **Load**.
7. You will notice Power BI designer with **Fields** present in the entities that you have chosen (on the far right), as well as available visualization.

Create a surrogate key that links customers and invoices (applies to Platform versions before November 2016

update)

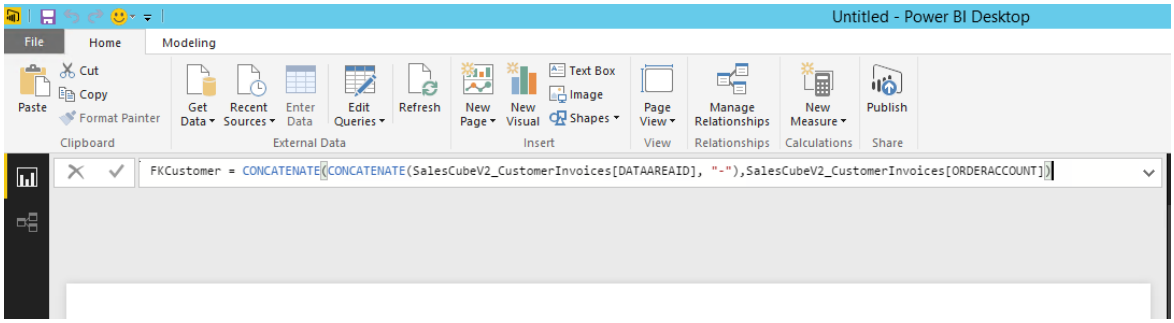
> **NOTE**
>
> Surrogate keys are generated in aggregate measurements staged into entity store. Power BI desktop does not enable you to relate table joins using multiple fields (also known as, composite keys). The **SalesCubeV2_Customer** entity does not have a surrogate key (such as AX RecID) defined in it. Next, you will create a surrogate key that enables relating a customer entity to invoices.

1. Select the ellipsis (...) icon next to the **SalesCubeV2_CustomerInvoices** entity. Right-click and select **New Column**.
2. Enter the following expression in the **Formula editor** window.

```
FKCustomer = CONCATENATE(CONCATENATE(SalesCubeV2_CustomerInvoices[DATAAREAID], "-"),
SalesCubeV2_CustomerInvoices[ORDERACCOUNT])
```

> **NOTE**
>
> When you enter the first few letters of the field name or function, the editor will display a list of candidate fields. This is called a type-ahead feature. You can either copy and paste this expression or use the type-ahead feature.

3. When completed, your formula should look similar to the following.



4. Notice that a new field, **FKCustomer**, is shown in the list of fields for the **SalesCubeV2_CustomerInvoices** table. Because this field is used to relate two tables, you can hide it from end users by right-clicking the field and selecting the **Hide** option.
5. Next, create a similar field in the **SalesCubeV2_Customer** table. Select the ellipsis (...) icon next to **SalesCubeV2_Customer** entity. Right-click and select **New Column**.
6. Enter the following expression in the **Formula editor** window.

```
FKCustomer = CONCATENATE(CONCATENATE(SalesCubeV2_Customer[DATAAREAID], "-"),
SalesCubeV2_Customer[CUSTOMER])
```

7. Notice that the field **FKCustomer** is shown in the list of fields for the **SalesCubeV2_Customer** table. Because this field is used for relating two tables, you can hide it from end users by right-clicking the field and selecting the **Hide** option.

Relate invoices and customers

NOTE

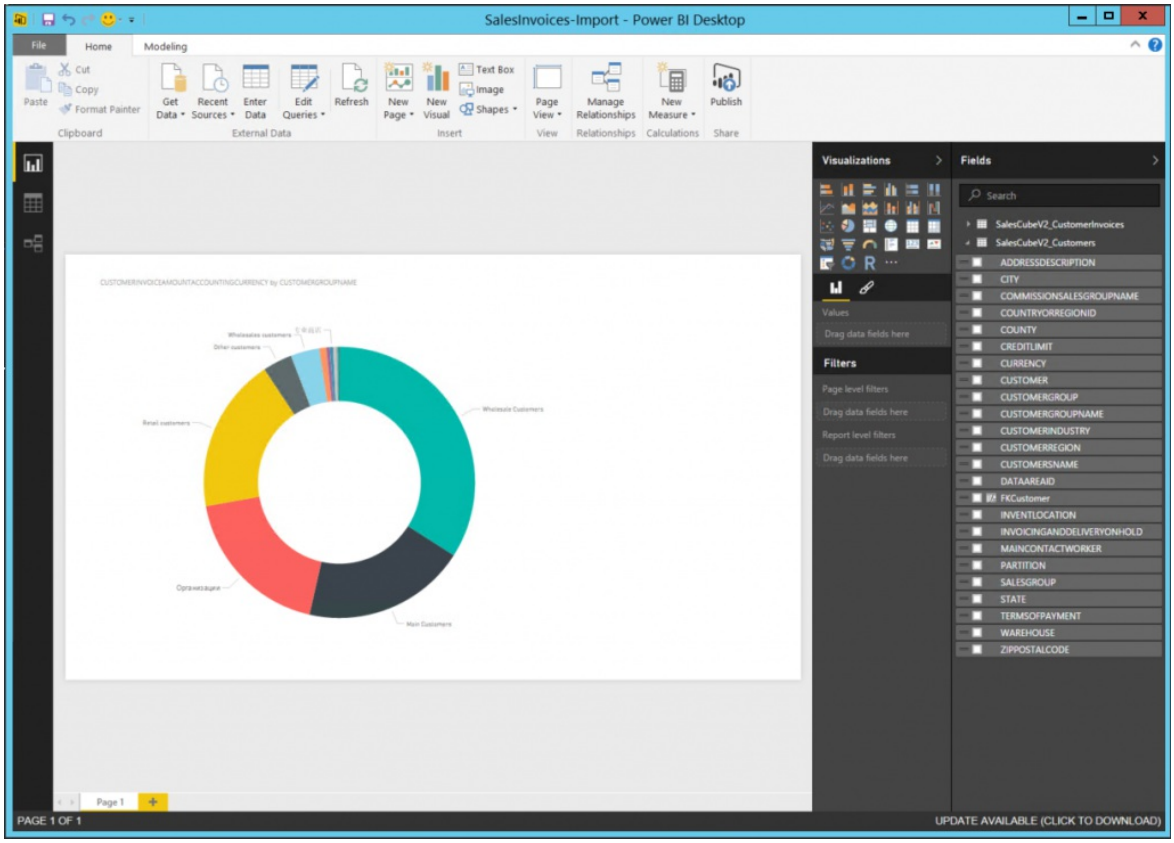
You can relate the surrogate keys already created within entity store. If not, you must relate the surrogate keys that you created manually. Next you will create a relationship between **SalesCubeV2_CustomerInvoices** and **SalesCubeV2_Customers** entities.

1. Click the **Manage Relationships** button on the Power BI ribbon. You will see the **Manage Relationships** dialog box. Click the **New** button.
2. In the **Create Relationship** dialog box, select **SalesCubeV2CustomerInvoices** as the first table in the drop-down list. Scroll to the right and select the **FKCustomer** field as the column to relate to.
3. In the second drop-down list select **SalesCubeV2Customer** as the table. Scroll to the right and select **FKCustomer** as the column to relate to.
4. Select the **Make this relationship active** option if it is not already selected. Click **OK** to continue.
5. You will notice the newly created relationship in the **Manage Relationships** dialog box. Click the **Close** button.

Create a Sales by state report

1. To create a report that shows sales by customer group, drag the **CustomerInvoiceAmountAccountingCurrency** field from the **SalesCubeV2_CustomerIncoices** table and drop it on the Power BI desktop canvas. Next, drag the **CustomerGroupName** field in the **SalesCubeV2_Customer** table to the same grid.

2. Change the chart type to a doughnut chart. You should see a report similar to the following.



3. You can create additional visuals using the Power BI desktop. When you save, you will notice that the file has a **PBIX** extension.

4. Save the report to your desktop.

5. At this point the report is fully functional (with data from your environment) and you can continue to use the Power BI desktop or upload this report to PowerBI.com and continue with data exploration.

6. Next, you will migrate this report to a production environment using LCS so that you can see this report with production data and share it with other users.

Publish the report and the model

Publishing a report and model requires uploading the report to Lifecycle Services, migrating the aggregate measurement to your production environment, configuring the client to point to the correct LCS library, and publishing your reports in your production environment.

Upload the report to Lifecycle Services

Microsoft Dynamics Lifecycle Services (LCS) is the tool used to migrate development artifacts from developer to production environments. In the May 2016 update, LCS supports migrating PBIX files (authored using the entity store) between environments.

1. Open LCS from the developer environment. If you haven't created a project in the LCS environment, create a project.
2. Scroll to the right and you will notice the **Asset Library** icon. Click the icon and launch **Asset Library**.

Notice that the asset library enables adding **PowerBI report models** (PBIX files) as implementation artifacts to a project.

1. Select the plus (+) icon to add a new asset.
2. Provide a name and a description. Click **Upload** and then locate the file that you saved in an earlier step.
3. After you successfully upload the file, click **Confirm**. Notice that the file is uploaded into LCS as an implementation asset. LCS supports managing versions and releases for Power BI reports. You can maintain several versions and publish reports to other environments, just as you would for other implementation artifacts. Because you added the PBIX files as an asset within an LCS project, environments that you deployed using that project will have access to this report.
4. Optionally, you can publish this report so that all of your projects can access the shared assets. If you are a partner or an ISV, and want to share this report with your customers, you would share this asset to your global library and enable your customers to import the asset into their respective LCS projects. To do this, select the **Save to my library** option.

Migrate the aggregate measurement to a production environment

1. You need to migrate the aggregate measurement that you modified in the developer environment to the production environment. You can follow the instructions in Generate a deployable package. create-apply-deployable-package.md.
2. After you successfully publish the model, perform the steps outlined in the **Refresh the entity store** section of this tutorial, so that the entity store is updated with data.

Configure an LCS project

If you haven't already done so, associate your environment with an LCS project so that Finance and Operations apps can consume assets within the project.

1. Launch the client from the instance that you want to use to deploy the Power BI reports. Typically this is the test or a production instance where you want to see a report with a different set of data than what you worked with as a report developer.

2. Open **System Administration** > **Setup** > **System parameters**. Select the **Help** tab. Using the **Lifecycle services help configuration** list box, select the LCS project that you uploaded the PBIX file to. Click **Save**.

> **NOTE**
>
> This form will only show the LCS projects that the current user has access to. If this step is being performed by an administrator, either the administrator needs to have access to the project, or the PBIX artifacts need to be imported into a project that the administrator has access to.

Publish Power BI reports to a production environment

1. Open **System Administration** > **Setup** > **Deploy PowerBI** from the client. You will see the file that you uploaded to LCS.
2. Select the **Sales Report** file and select the **Deploy Power BI files** option on the menu bar.

> **NOTE**
>
> You may be asked to consent publishing to the PowerBI.com service. Click the link to provide consent. When consent is complete, you need to go back to the original browser window and click the **Close** button.

3. After you successfully publish the file, the Power BI report will appear in your PowerBI.com subscription. You will notice that the report now points to the entity store in the production environment.

Continuing with PowerBI.com

As an administrator or a power user, you have successfully authored and published a Power BI report to the production environment using the entity store. You can perform several additional steps using Power BI functionality.

- Optionally, you can apply record-level security to the dataset to restrict users from seeing data they are not allowed to view in Power BI.
- You can create an organizational content pack and share it among users in a group.
  - You can export datasets, reports, and dashboards from your PowerBI.com instance as a new content pack to a selected group of users.
  - Note that organizational content packs adhere to any record-level security rules that you defined at the dataset level.
- Users can personalize their workspaces by adding Power BI tiles or reports.

Additional resources

Model aggregate data

Upgrade budget planning

2/18/2021 • 11 minutes to read • Edit Online

There are significant differences in budget planning between Microsoft Dynamics AX 2012 and Dynamics 365 Finance. Some features were not upgraded and therefore require reconfiguration. This topic explains what must be reconfigured and also describes new features that should be considered after the upgrade is completed.

Budget planning in Finance has many enhancements that weren't available in Dynamics AX 2012. This topic explains the changes that customers who upgrade must make. It also points out the new features that should be considered in the upgrade process. Because of the extent of the changes, any existing budget plans will not be able to be opened until the changes that are outlined in this topic are made. However, reports should continue to work and not require additional changes.

Overview of changes

Many significant changes have been made in Budgeting for Finance and Operations. These changes are intended to make Budget planning easier to configure and more reusable, to reduce year-over-year maintenance and setup. The following areas in AX 2012 no longer exist in Finance:

- Budget plan templates (Budget planning configuration)
- Budget plan folders (Budget planning configuration)
- Scenario constraints (Budget planning configuration)
- Templates for Budget planning stage rules and templates (Budget planning process)
- Matrix fields for worksheet templates
- Budget plan Microsoft Excel template wizard

Some new concepts can't be directly upgraded from the previous functionality. Therefore, you must complete some reconfiguration to address these new concepts. The following sections describe the concepts that have replaced the items in the preceding list.

Columns

Columns are a new concept that replace parts of the Excel template and also matrix fields. Columns can represent a period, month, quarter, year, or all time. The time reference is dynamic. It points to a relative period or year in reference to the budget process. For example, a **Prior Year January** column references fiscal period 1 for year -1. A column is specific to a budget plan scenario, such as actuals or budget request.

Layouts

Layouts are a new concept that replace the Excel template. Layouts contain the columns that define which budget or actuals data and periods should be shown. Layouts are also shared between the client and the Excel add-in. Therefore, the user experience when you enter or view data in the Finance and Operations client is better than the user experience in AX 2012. To enter data in the Finance client, you're no longer limited to viewing and entering a single scenario in a transaction view. Instead, a comparison view lets you easily view and enter amounts for multiple periods and accounts at the same time. Layouts can also be defined so that you can enter and view currency, comments, and other optional data. Layouts also let you define which ledger dimensions and dimension descriptions should be shown. Layouts also incorporate scenario constraints to define which columns in a template can be edited and which columns should be available in Excel. After you define a layout, a template is generated for it. This template, in turn, creates the corresponding Excel template. You can then edit the Excel template to incorporate more formulas and formatting, and then upload it again. Layouts are then assigned to each stage rule on the **Budget planning process** page. Therefore, the layouts replace templates, which were assigned and used in a similar manner.

Budget planning processes

Budget planning processes are mostly the same as in AX 2012. The most significant change is the replacement of templates with layouts. If any processes were previously completed in AX 2012, the processes are updated to a status of in-progress so that changes can be made. You must assign layouts will need for each stage rule to determine which scenarios and time periods appear when the plan is opened in the client. The layouts also determine which Excel template is opened outside Dynamic 365 Finance so that you can view the budget. **Default account structure** is a new required field for the Budget planning process. For each Budget planning process, assign the primary account structure that should be used for budgeting.

Attachments

In AX 2012, justification documents were saved to an attachment folder. No previous justification documents are upgraded. Justification documents are now stored in the database. If this information should be saved in the upgraded version, you can upload final justification documents for each plan as an attachment by using the **Justification** button on the Action Pane. In AX 2012, Excel worksheets for each budget plan were created based on the template. In Finance, all plans open a copy of the layout. However, no changes to the Excel file are saved. Any formulas or supporting information that were used on a per-plan basis must be added via comments, a justification document, or some other supplemental process.

Configuring an upgraded environment from AX 2012

To help you determine how to configure the upgraded system, the following example uses an upgraded budget process from AX 2012 demo data. Default configuration data for columns were created to help with the upgrade process. You can update or delete this default data if doesn't meet your configuration requirements. **Note:** There are new required fields that won't be set in the system. If you get stuck on a page, such as the **Budget planning configuration** page, and can't navigate away, you can close your browser and then reopen it to a different page to enter details in the correct order. There are required fields that aren't yet set. Therefore, issues might occur until everything is configured and all required fields have been set. This topic explains how to set these fields, as required. Here are some of these required fields:

- **Budget planning process** page: **Default account structure** field
- **Budget planning process** page: **Layout** field on the **Budget planning stage rules and layouts** FastTab

Define columns and layouts

1. On the **Budget planning configuration** page, click the **Columns** tab. As part of the upgrade, new columns are automatically created based on your budget plan lines. Columns now use dynamic dates, where the time and year are offset from the fiscal year that is defined in the Budget planning process. **Note:** For performance reasons during upgrade, it's assumed that all budget cycles represent calendar years, not fiscal years. If you use fiscal years, you must make edits to correctly map the columns to their fiscal year. For example, the following elements existed in AX 2012:

   - Budget plan scenarios: Actuals, Baseline, Budget Request, Budget Approved
   - Budget plan lines for all scenarios in 2017, and Actuals for both 2017 and 2016

   The following columns will be created in Finance and Operations:

COLUMN NAME	BUDGET PLAN SCENARIO	COLUMN TIME PERIOD	YEAR OFFSET
Jan Scenario 1	Actuals	1	0
Jan Scenario 2	Baseline	1	0
Jan Scenario 3	Budget Request	1	0
Jan Scenario 4	Budget Approved	1	0

COLUMN NAME	BUDGET PLAN SCENARIO	COLUMN TIME PERIOD	YEAR OFFSET
Jan Scenario 5	Actuals	1	-1
Feb Scenario 1	Actuals	1	0
...	...	...	...

In this example, a column that is named **Jan Scenario 1** is created for the most recent budget plan transaction data that is found where transactions exist in January. A similar column is created for each scenario that has data. After columns exist for all periods in that year, columns are created for previous years.

2. Change the column names and descriptions, and any other details, either manually in the client or by doing bulk updates through the Excel add-in that points to the budget plan columns data entity. Any filters that were previously set for matrix fields are now set within the columns.
3. Create a new budget plan layout. A layout points to several columns to define the view that appears in Excel and the client. The layout first requires that you specify a ledger dimension set to determine which financial dimensions can be entered. After you specify the dimension set, click **Descriptions** to select dimension descriptions to include in the layout.
4. On the **Layout elements** FastTab, click **Add** to add metadata for each row, such as a currency, a comment, or a budget class that determines revenue versus expense rows. Next, add columns for the time period, and scenarios that apply to this budget cycle and stage. You can make these changes manually in the client or through the Excel add-in that points to the budget plan layout elements data entity.
5. For each layout element, select whether the column should be editable, and whether the column should also appear in the Excel workbook for this layout. **Note:** For our historical plans, you might want to consider a layout that shows 12 monthly columns for any budget plan scenarios for that process.

Update budget planning processes to use the appropriate layout for each budget stage

1. On the **Budget planning process** page, select the process to configure.
2. On the Action Pane, click **Edit**.
3. Specify the default account structure for this budget process. **Default account structure** is a new required field that must be set.
4. On the **Budget planning stage rules and layouts** FastTab, in the **Layout** field, select a layout that was previously configured, and that is appropriate for this stage.
5. Continue to select the same or different layouts for the various budget planning stages, and then save your changes.

Additional features to consider in your budgeting process

Budget planning workspace

This workspace is designed for both the budget owner and individual budget contributors. It has links to any budget documents that require your attention. It also has reports and key performance indicators (KPIs) for the budget process. The budget administrator can define the current Budget planning process for all users on the **Budgeting parameters** page. On the **Workspace settings** tab, the **Budget planning** FastTab includes a field where you can select the budget planning process.

Alternate layouts

Alternate layouts are a new feature that lets you view plans in different layouts. One or more layouts can be

provided as options. You can then view a budget plan in a monthly layout or a quarterly layout. You define alternate layouts on the **Budget the planning stage rules and layouts** FastTab on the **Budget planning process** page.

Budget milestones

As part of the budget process, it's vital that you understand key dates and deadlines. You can now configure dates so that they have descriptions. Budgeting users will see these descriptions when they open budgets to edit or view anything that is assigned to them.

Copy from Budget Plan allocation

A new allocation method lets you distribute from a parent plan to a child plan without having to go through an intermediate level in the hierarchy. This method is especially useful for customers who previously created financial dimension just for budget distribution and approvals.

Generating budget plans from new budget sources

The following options were added as periodic processes. These options let you generate a budget plan by using existing data from another module as the starting point:

- Generate Budget Plan from Demand Forecast
- Generate Budget Plan from Supply Forecast
- Generate Budget Plan from Project
- Generate Budget Plan from Budget Register

More complete tracking of amounts

In AX 2012, budget planning had a single plan amount that was stored for each value. In Finance, the data model has been expanded. There are now accounting currency, transaction currency, and reporting currency amounts for each value. During the upgrade, these new columns are automatically filled in for existing data.

Do not convert currency in aggregation

Typically, when a child plan is aggregated to a parent level, the amounts are automatically converted from the transaction currency to the accounting currency for the organization. When you set the **Do not convert currency in aggregation** option to **No** the aggregated amounts remain in the original currency. Therefore, this option allows for more accurate adjustments that are affected by exchange rate fluctuations.

Looking back from a budget plan to other modules that contributed to the budget

Budget plans can be generated from demand or supply forecasts, project, and other areas. The Budget plans by dimension set inquiry includes several options that let you run queries to identify the data that was the source for the budget plan.

Overwrite or append to plan for allocation schedules

If there are multiple sources for amounts that must be distributed, you can specify that the amounts should be additive. In this case, the amounts don't overwrite any existing amounts. Instead, they are appended to the existing amounts.

Default financial dimension set for budget planning configuration

The **Budget planning configuration** page now includes a field where you can specify the default financial dimension set. Although this field is an optional field, it might be required for certain inquiries. It might also be required if you want to group or filter reports grouping by dimension set.

Data entities

Several data entities have been added to enable rapid implementation of Budget planning. The entities also let you make many changes through Excel. Therefore, you don't have to create items one at a time through the client. Here is a list of the new data entities:

- Entity name

- Budget parameters
- Budget plan parameter
- Budget plan scenarios
- Budget plan stages
- Budget plan workflow stage
- Budget plan allocation schedules
- Budget plan stage allocations
- Budget plan priorities
- Budget plan columns
- Budget plan layout elements

AX 2009 upgrade - Use the Data migration tool to migrate from Dynamics AX 2009 to Finance and Operations

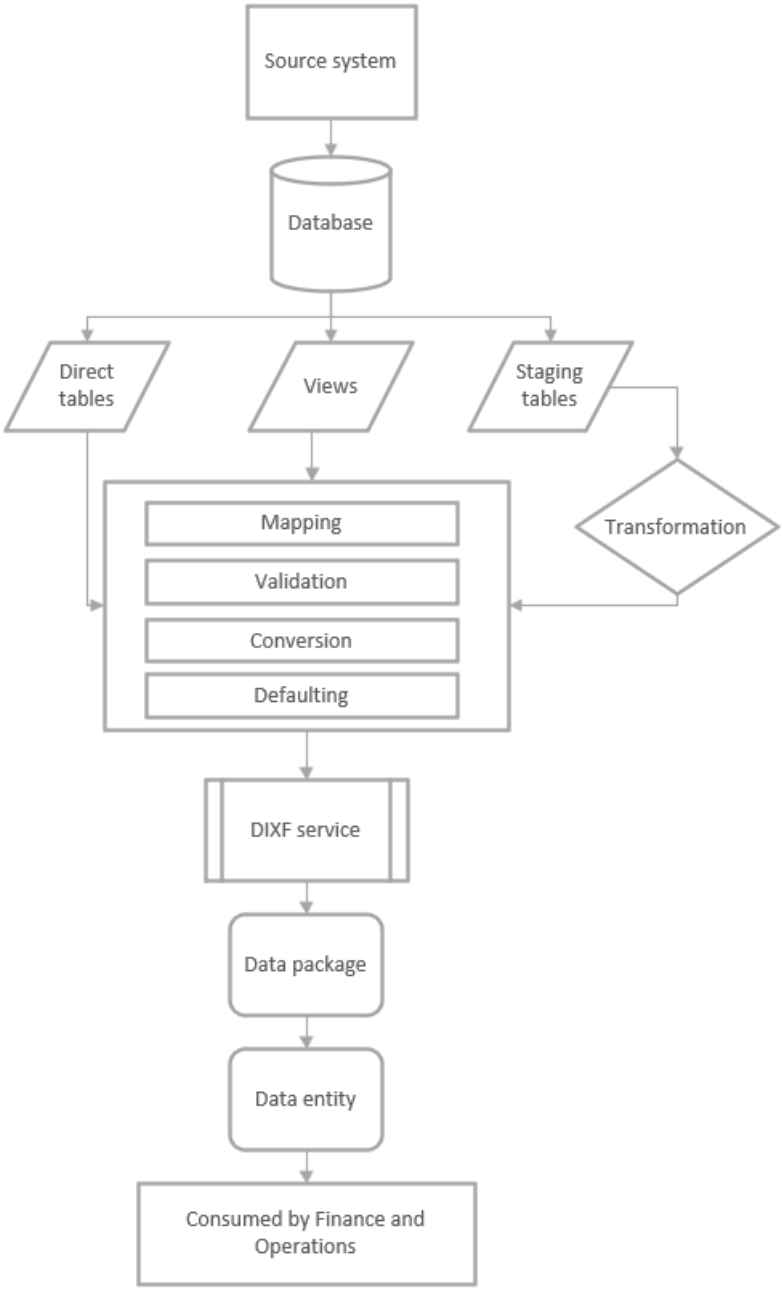
2/18/2021 • 2 minutes to read • Edit Online

You can use the Microsoft Dynamics AX 2009 Data migration tool (DMT) to migrate your data from AX 2009 to Finance and Operations. Using the DMT is the only supported upgrade path from AX 2009. The DMT helps you find and fill gaps between the table schemas for each version, as well as helping you move your data.

> **NOTE**
>
> Start your cloud migration journey with a no-charge, no-obligation migration assessment though the Dynamics 365 Migration Program.

Architecture

The following illustration describes the architecture of the DMT, and how data from the source system (AX 2009) is processed and moved to the target system (Finance and Operations).



Data migration process

The following illustration shows the overall process of collecting and preparing the data in your AX 2009 instance and then importing that data into your new environment.

Before you can use the DMT to export data from the source environment (AX 2009), you must complete the following pre-processing tasks:

- Mapping the table fields between the source and target environments
- Applying conversions to the source data
- Setting up default values for the source data
- Applying query filters

Because there can be multiple legal entities in the source system, you must select the legal entities that contain

the data to migrate. For the selected legal entities, you can review the source tables and their row counts. You can also view any virtual companies. Finally, you can analyze virtual companies that legal entities are attached to and the related tables.

Successful migration of exported data requires that a source table be mapped to an equivalent target data entity. You set up the mapping by using a Microsoft Excel mapping file that allows for automatic mapping of the source and target fields of each table. The mapping file also includes the data from the schema of the new data entities and the default data that is required in some tables.

Before you can migrate data from AX 2009, you must complete the following tasks to meet the migration requirements:

- Select target dimensions that correspond to the source dimensions that are populated based on the selected legal entities.
- Review the inventory dimensions that are included with the selected legal entities.
- Select the chart of accounts for each legal entity, or consolidate multiple legal entities into a single chart of accounts.
- Complete the basic ledger setup.
- Apply data conversions to the source data, based on the extended data type (EDT) of the field.

AX 2009 migration – Install the Data migration tool

2/18/2021 • 5 minutes to read • Edit Online

This topic explains how to set up the Data migration tool (DMT) so that you can migrate data from Microsoft Dynamics AX 2009 to Finance and Operations.

> **IMPORTANT**
> At this time, the DMT is in private preview. If you are interested you can sign up for the Preview Program. The public release date for the DMT has not been set.

Prerequisites

- Microsoft SQL Server 2008/2012/2014/2016.
- The Microsoft .NET Framework version 4.5 or later.
- Microsoft SQL Server machine that has Microsoft SQL 2012 Native Client installed.
- The Microsoft SQL Server Integration Services (SSIS) service is installed and running on the machine where the DMT service will be installed.
- SQL Server authentication must support both SQL authentication and Microsoft Windows authentication.
- Microsoft Access database engines that follows the version guidance in the following table.

	SQL SERVER 2008	SQL SERVER 2012 AND LATER
No Microsoft Office on the VM	Access engine 32-bit	Access engine 64-bit
Microsoft Office 32-bit	Access engine 32-bit	Access engine 64-bit
Microsoft Office 64-bit	Access engine 32-bit and 64-bit	Access engine 64-bit

- Microsoft Dynamics AX 2009 SP1 5.0.1000.52 or later.
- The prerequisite patch (axpatch.exe) installed. To find the patch, from the location where you downloaded and extracted the zip file, go to <pre-requisiteforpatch><application>.

Install DIXF service

1. Go to the location where you extracted the zip file, and then, in the **DIXF msi** folder, right-click **DIXF_Service_x64.msi**, and select **Run**.
2. When the wizard starts, select **Next**.
3. Accept the license terms, and then select **Next**.
4. Select an account for the service, and then select **Next**. The account should have admin rights. If you select the **Network Service** check box, verify that the network service account has admin rights. Otherwise, clear the check box, and enter an admin account user name and password. Then select **Next**.
5. Select the SQL Server version, and then select **Next**.
6. Select **Install**, and then, when the wizard is completed, select **Finish**.

Copy binaries

Go to the location where you extracted the zip file, and copy the following files to the **Program Files (x86)\Microsoft Dynamics AX\50\Client\Bin** folder:

- Microsoft.Dynamics.AX.Framework.Tools.DMT.dll
- Interop.Shell32.dll

Install DMT components for AX 2009

There are two ways to install the DMT. You can use the combined XPO file or an application hotfix. If you're using a Microsoft Dynamics Lifecycle Services (LCS) Implementation project, use the application hotfix. Installation takes approximately seven hours.

Combined XPO file

1. Extract the combined XPO file from **DMT_V1.0\CombinedXPO**.
2. Import the combined XPO file into AX 2009.
3. Copy the label file from **DMT_V1.0\Label file** to the **Program Files\Microsoft Dynamics AX\50\Application\Appl\<NameOfYourDeployment>** folder.
4. Restart the Application Object Server (AOS) instance.
5. In AX 2009, select **Data migration** > **Setup** > **Compile and synchronize DMT application**.

Note that the combined XPO file is imported into the layer that the user is signed in to.

Application hotfix

1. Go to **DMT_V1.0\ApplicationHotfix\DynamicsAX2009-KB4010403-SP1**, right-click **setup.exe**, and then select **Run**.
2. In AX 2009, in the Application Object Tree (AOT), notice that the **LegalEntityId** field has been added to the **DMTCustomerAddressView** and **DMTVendorAddressView** views.
3. Select **Data migration** > **Setup** > **Compile and synchronize DMT application**.

Parameter setup

Go to the location to where you extracted the zip file, and find **defaultvalue.xlsx**.

> **NOTE**
> The file is saved in .xlsx format. Don't change the extension. When you provide this file as input for the **Default configuration** parameter, select **All Files** so that you can select the .xlsx format. If you don't select this format, errors will occur when you start to generate mappings.

1. In AX 2009, select **Data migration** > **Setup** > **Configure default maps**, and enter the appropriate information in the following fields:
   - **Default configuration** – Enter the path of the Microsoft Excel file.
   - **Export file path** – Enter the server path that can be accessed by the service.
   - **SQL Server user and password** – Enter the SQL authentication credentials for the AX 2009 database.
2. Close the form.
3. Under **Setup**, select **Configure connections**, and enter the appropriate information on the following fields:
   - **DIXF service host** – Enter the host name of the DIXF service installation.

- **Tenant URL** – Enter the URL for the application tenant. If you aren't sure of the tenant, see the web.config file for the Finance and Operations application.

> [!NOTE} In the Azure Portal, when you create a new app in the Azure Active Directory (AAD), you can select from two options. **Web API** and **Native**. In this instance, select **Native** and grant permissions to native AAD app.

Multi-box setup

For a multi-box setup, you must have the following machines:

- Machine A, where the AX 2009 database and DIXF service are installed
- Machine B, where the AX 2009 AOS instance is installed
- Machine C, where the AX 2009 client is installed

In this three-machine setup, machine C is configured to connect to the AOS instance on machine B. Machine B is connected to the database that is configured on machine A.

DIXF service machine prerequisites (machine A)

The DIXF service on machine A has the following prerequisites:

- SQL Server 2008/2012/2014
- The .NET Framework version 4.5
- Access database engines
  - **For SQL Server 2008:** Access engine 32-bit and 64-bit (if Microsoft Excel is 64-bit)
  - **For SQL Server 2012 or later:** Access engine 64-bit
- AX 2009 database (configured on SQL Server)

AOS machine prerequisites (machine B)

The AOS installation on machine B has the following prerequisites:

- AX 2009 AOS Server
- Application files

Client machine prerequisites (machine C)

The client installation on machine C has the following prerequisites:

- AX 2009 client

Shared folder permissions

The path of the default configuration file and the export package file should be shared, and client users and the DIXF service should have read/write access to these files. To grant this access, select **Data migration** > **Setup** > **Configure and generate maps**, and then select **Validate path** to verify that the required access is available.

Set up parameters

1. Select **Data migration** > **Setup** > **Configure connections**.
2. In the **DIXF service host** field, enter the name of the remote machine where the DIXF service is installed. By default, the name is **localhost**.
3. Select **Validate** to validate that the client can access the DIXF service.

Workarounds

If you receive an error message that states, "DIXF service is unavailable," complete the following workaround to enable a service connection for port 7000.

1. Open port 7000, and then, for inbound rules on the DMT service machine, select **Firewall settings**, and then select **Run** > **wf.msc**.
2. Select **Inbound Rules** > **New rule**, and then, on the **Rule Type** tab, select **Port**, and then select **Next**.
3. In the **Specific local ports** field, enter **7000**, and then select **Next**.
4. Select **Allow the connection**, and then select **Next**.
5. Select all three check boxes to apply all the rules, and then select **Next**.
6. Enter the name of the rule, and then select **Finish**.
7. Repeat these steps for outbound rules.

AX 2009 migration – Generate maps

2/18/2021 • 2 minutes to read • Edit Online

Before you can migrate your data from Microsoft Dynamics AX 2009 to Finance and Operations, you must align your source data with your target environment. This topic explains how to generate source-to-target mappings.

Before you can generate maps, you must provide the target URL, tenant URL, and service app ID to validate the connection.

> **NOTE**
> When you create a new app under Microsoft Azure Active Directory (Azure AD) in the Azure portal, you have two options, **Web API** and **Native**. Select **Native**, and grant permissions to the native Azure AD app.

Prerequisites

Before you generate the data maps between the source and target environments, you must install the Data migration tool (DMT). For more information, see AX 2009 migration - Install the Data migration tool.

Generate maps

Follow these steps to generate maps for data migration.

1. In AX 2009, in the navigation pane, go to **Data migration** > **Setup** > **Configure connections**.
2. Review the field information to verify that it's correct, and then click **Validate**.
3. After the validation is completed, close the form.
4. Under **Setup**, click **Configure and generate maps**.
5. Verify that the information in the form is correct, and then click **Validate path**.
6. After validation is completed, click **Generate maps**.

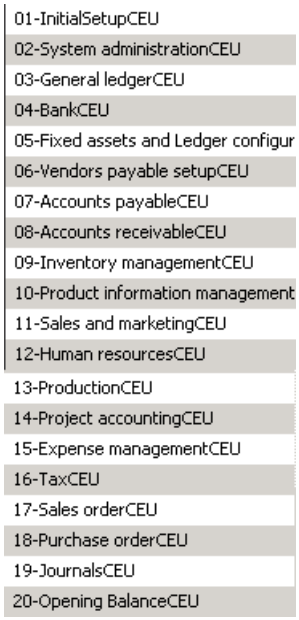
> **NOTE**
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

AX 2009 migration – Create package templates

2/18/2021 • 2 minutes to read • Edit Online

Packages are created by following a predefined sequence. This sequence is based on the dependencies that the data entities have on each another. Because of these dependencies, when you import data entities, you must import the data entities in the defined order. Otherwise, you might encounter issues during import and configuration.

The Data migration tool (DMT) provides twenty predefined templates, as shown in the following illustration.



You can customize an existing template, or you can create your own templates as you require.

Follow these steps to view and select the entity lists that will be used in the templates for migration.

1. In Microsoft Dynamics AX 2009, click **Data migration** > **Common forms** > **Entity list**, and then click **Apply sequence**. Close the message box.
2. Verify that the correct legal entity is selected, and then, in the **Show** field, select to view either all entities or only those entities that should be considered for migration.
3. In the **Template name** field, select a template.
4. In the **Module selected** pane, select the module that contains the data entities to migrate.
5. On the **Entity details** tab, select the **Select for migration** check box for every entity line that you want to migrate.
6. Click **Apply sequence**.
7. To create a customized template, in the Application Object Tree, go to **Resources**, and create a new template in XML format.

AX 2009 migration – Create migration groups

2/18/2021 • 2 minutes to read • Edit Online

When you create a definition for migration, you determine which entities should be packaged and exported together, and then put all the entities together in a migration group. A migration group is a set of entities that must be processed in a sequence, or that can logically be grouped together. The entities in a migration group are exported together, either from the source to staging or directly to a file package. In a migration group, you also associate legal entities. Migration groups must be set up before you begin the export process.

Follow these steps to create a migration group.

1. In Microsoft Dynamics AX 2009, in the navigation pane, click **Data Migration** > **Common forms** > **Create migration group**.
2. In the **Migration group** form, press CTRL+N or click **New** to create a new migration group.
3. Enter a name for the migration group. Then press Tab to move to the **Company** field, and click **Select company**.
4. In the **Select company accounts** form, select one or more companies to add to the migration group, and then click **OK**.
5. In the **Migration group** form, click **Entity**, and select the lines to include in the migration.
6. Fill in any gaps in the field mapping, as required.
7. Click **Apply sequence**, and then close the form.

AX 2009 migration – Export packages

2/18/2021 • 2 minutes to read • Edit Online

You can use the Data Import/Export Framework (DIXF) service in Microsoft Dynamics AX 2009 to retrieve data that must be migrated to Finance and Operations. The export process is completed through a job ID. When you export, you can specify how the export job is defined. You can select the source data to export, the conversion value, and the field mapping. You can also apply a query to each source to limit what is exported.

The export package that the Data migration tool (DMT) generates can consist of one or many data entities. A typical data package consists of a group of entities for a specific task, such as import. For example, the data entities that are required for system setup might be part of one data package. The format of a data package is a compressed file that contains a package manifest, a package header, and any additional files for the data entities that are included.

Before you create a data package, plan out what should be included. In this way, you help guarantee that the correct entities, entity sequence, and fields are included.

Follow these steps to export the data package.

1. In AX 2009, in the navigation pane, click **Data migration** > **Common** > **Create migration group**.
2. In the **Migration group** form, select the migration group to export, and then click **Export now**.
3. In the **Export data** form, update the export file path as required, and then click **OK**.

AX 2009 migration – Import packages

2/18/2021 • 2 minutes to read • Edit Online

Data can be imported for a group of logically related entities that are sequenced in the correct order. You have three options for importing Microsoft Dynamics AX 2009 data that you want to migrate:

- AX 2009
- Finance and Operations

AX 2009

You can import data for migration directly from the source system. Follow these steps.

1. In AX 2009, in the navigation pane, click **Data migration**.
2. Go to **Common** > **Create migration group**.
3. In the **Migration group** form, select the migration group to export, and then click **Export now**.
4. In the **Export data** form, select the **Import package in target** check box, and then click **OK**.

Finance and Operations

You can import data for migration by using your Finance and Operations environment. Follow these steps.

1. Sign in to your environment by using an Administrator role.
2. On the dashboard, select the **Data Management** workspace.
3. Select **Import**.
4. Enter the name of the package, and then, in the **Source data format** field, select **Package**.
5. Select **Upload**, and then select the appropriate package file from the location for the data that is being imported. All the files from the package are imported.

Code migration and upgrade home page

2/18/2021 • 2 minutes to read • Edit Online

Migrate your code

To migrate your code from Dynamics AX 2012 to Dynamics 365 Finance, Supply Chain Management, or Commerce, use the "Migrate and Create Solutions" methodology in Lifecycle Services.

Key concepts

The following links (also included in the methodology) describe key concepts and steps in the migration process. The links are listed here in the order that we recommend you read them.

- Prepare to migrate code to Finance and Operations
- Model split
- Removed or deprecated features for Finance and Operations
- Deprecated APIs

Additional concepts

- Solve dependencies among models by using delegates during code migration
- How to import a SQL Server Analysis Services Project into the AOT
- Upgrades, updates, and hotfixes resources
- Workflow subsystem updates in Finance and Operations
- Migrate upgraded AX 2012 R3 sales cubes to the entity store

Prepare to migrate code to Finance and Operations

2/18/2021 • 10 minutes to read • Edit Online

This topic describes how the Lifecycle Services code upgrade service and Visual Studio tools help you migrate your code and metadata from Dynamics AX 2012 R3 to Finance and Operations. Most of these steps also apply to code migration between two major versions of Finance and Operations.

Prerequisites

You will need access to a Finance and Operations development environment using Remote Desktop, and be provisioned as an administrator on the instance. We recommend you become familiar with some of the Finance and Operations development, customization, and user interface concepts before you upgrade your code. Here are some references.

- Development tools
- Models and packages
- X++ programming language
- Extensions and Overlayering
- User interface development

Overview of the code migration process

Model split

The Finance and Operations application is split into several packages, or assemblies:

Platform Packages

- Application Platform
- Application Foundation
- Test Essentials

Application Packages

- Application Suite
- Other application packages.

ISV and customer code that is migrated from Dynamics AX 2012 R3 will be re-baselined into the correct package.

Auto-migration using the LCS Code Upgrade service

The LCS code upgrade service takes a Dynamics AX 2012 R3 model store as input and completes the following tasks:

- Converts metadata into the latest format.
- Re-baselines metadata, by moving and merging, into the right model.
- Provides an estimation to understand the effort required to upgrade the solution.
- Runs migration rules that auto-migrate parts of a solution.
- Runs migration rules that inform developers what to manually fix by using TODOs.
- Automatically checks-in the upgraded solution into your Azure DevOps project.

To configure and run the code upgrade service, see Configure the code upgrade service in Lifecycle Services

(LCS).

Manual migration steps

After you upgrade your code using the LCS code upgrade service configure your developer VM and Azure DevOps to connect to the upgraded code branch.

- Configure one-box development environments
- Configure the Azure DevOps mapping during code migration

The code upgrade service will provide with Visual Studio solutions that you can open to compile your code. A **code merge** solution for all elements that contain conflicts and an **upgraded** solutions for all your upgraded elements. Typically, you can compile the application by fixing compilation errors in the order shown below. The order is determined based on the package dependencies graph, start with the lowest package in the graph. To determine package dependencies, see Models and packages. A typical order is Application Platform, Application Foundation, Directory, ...etc., Application Suite. For each of your upgraded models:

- Fix merge conflicts.
- Fix compilation errors related to a model split (references across packages).
  - Typical error messages are:
    - <Element Type> X refers to <Element Type> Y which does not exist.
    - The name <Name> does not denote a class, a table or an extended data type.
  - For example, your overlayering customizations may be referencing elements or code that are higher in the package dependency graph:
    - A method in the Directory model is referencing a table in the Application Suite package.
    - A form in the Directory package is referencing a data source in the Application Suite package.
  - You will have to refactor your code to address these dependencies by moving model elements or business logic to higher level packages.
  - Solve dependencies among models by using delegates during code migration describes how to use delegates to solve some of these issues.
- Fix compilation errors.

After you have resolved all of the compilation errors, all packages will compile. Next, you must complete the following tasks:

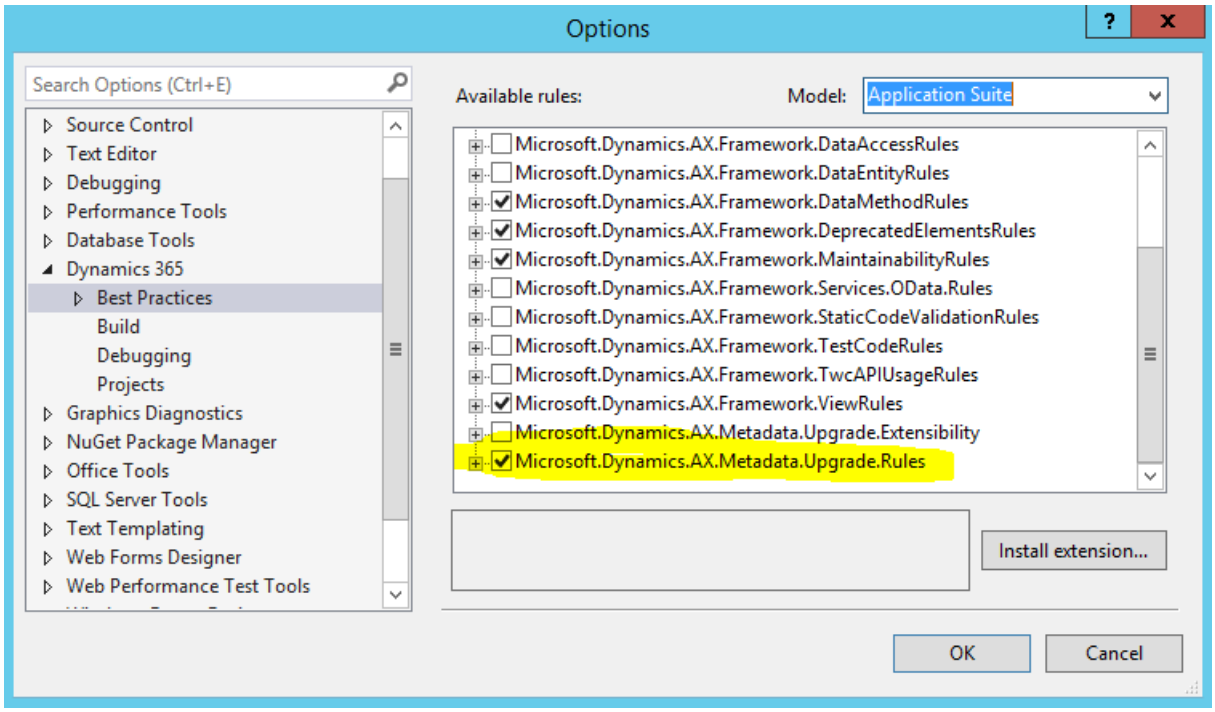
1. Address guided code upgrade TODOs and code upgrade-specific best practice warnings. Some examples and details are in the sections below.
2. Replace deprecated controls, for example, ActiveX or find an alternative.
3. Apply form patterns and sub patterns to all forms.
4. Validate that all scenarios work in multiple browsers with different sizes for custom patterns.
5. Write and run tests.

Best practice setup

In the Best Practice framework, there is a subset of Best Practice warnings that need to be resolved to complete migration. This applies if you are migrating from Dynamics AX 2012 R3 or earlier.

1. In Visual Studio, click **Dynamics 365 > Options > Best Practices**.
2. In the **Model** drop-down menu, select **Application Suite** (Repeat with all models you are working on)

These rules should be set to "ON" while migrating your solution. The setting is driven by an XML file in the AxRuleSet folder. For example, see the Application Suite xml file, BPRules.xml, located under C:\Packages\ApplicationSuite\Foundation\AxRuleSet.



To complete the migration, you need to fix all migration-specific Best Practice rules. The errors will show up in the error list as warnings. In the error list, you will see compiler warnings and best practice errors. Best Practice errors are prefixed with the text **BP**. For example, **BPErrorFormControlPatternUnspecified**.

Debugging

By default, Finance and Operations optimizes the debugging experience for the files that you are working on. As a result, when you step into a file (F11) that is not in your project, the PDBs are not loaded and you can't debug the code. To work around this, change the project debugging setting by clicking **Dynamics 365 **> **Options** > **Debugging**. Verify that the **Load symbols only for items in the solution** check box is not selected. This option is selected by default because it improves the debugger speed significantly. Another debugging setting that you may want to turn off is Intellitrace. Intellitrace collects the complete execution history of an application. It creates a lot of noise in the IDE when debugging. To turn off Intellitrace, click **Options** > **IntelliTrace** > **Enable IntelliTrace**, clear the check box, and then click **OK**. Note that Intellitrace is only available in the Enterprise version of Visual Studio.

Address code migration tasks

When metadata is migrated to Finance and Operations, multiple auto-upgrade scripts are run. In the case where developers need to complete manual migration tasks, TO DOs and Best Practices (BP) have been added.

- TO DOs are prefixed with `/* TODO: (Code Upgrade)`, and need to be fixed as a part of code migration.
- BP migration specific rules also need to be fixed as part of code migration.

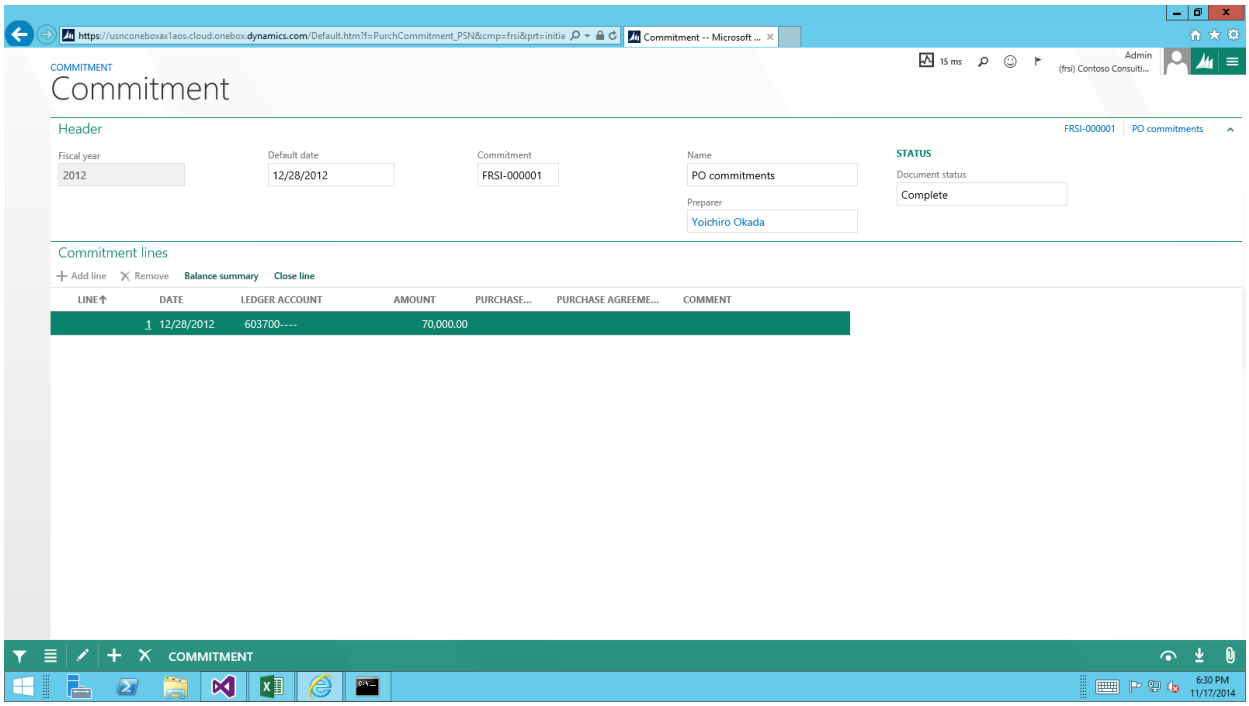
This example below uses the **PurchCommitment_PSN** form to walk you through the migration task of fixing navigation. Specifically, you will see examples of duplicate buttons and Action Pane TODOs.

Setup

1. In Visual Studio, open **Application Explorer**, and search for the form, PurchCommitment_PSN.
2. Click **OK**.
3. Right-click the project and select **Properties**.
4. In the Model property, select **Application Suite**.
5. In the Company property, select FRSI.
6. Note: The form is located in the French demo data company FRSI.

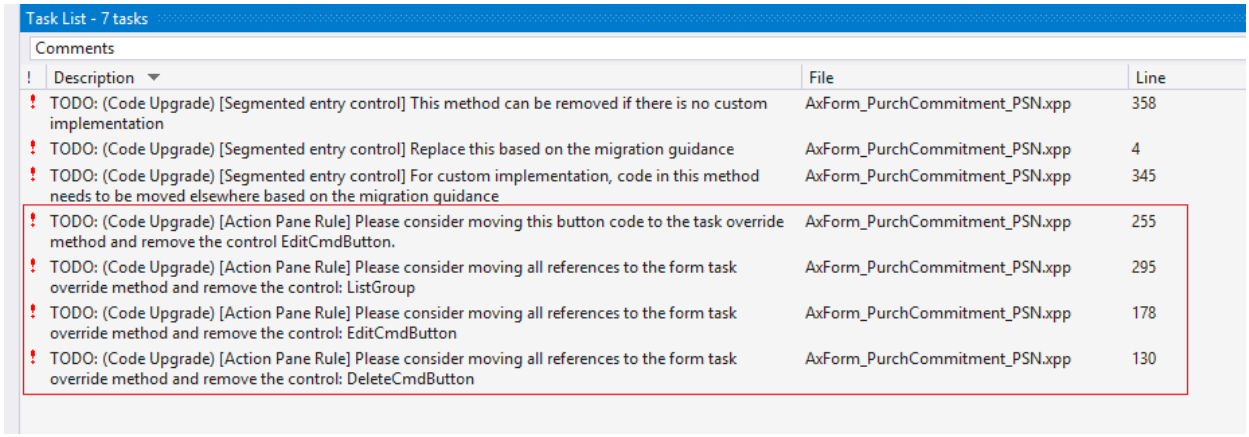
7. Press **Ctrl+F5** to see the form.

While the form looks complete, there are still code migration tasks necessary to be migration-complete.



Navigation migration tasks

1. In Visual Studio, build the project, and then on the toolbar, click **View** > **Task List**.
2. Click the **Comments** drop-down list to view the TO DO: (Code Upgrade) tasks.
3. In the list, find the ActionPane TODOs.



Code upgrade rule - Action Pane

In Finance and Operations, the following core actions are provided as system-defined buttons:

- New
- Delete
- Edit
- Export

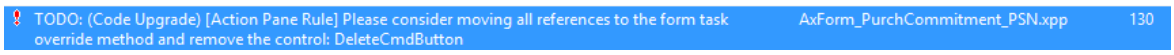
As part of the auto-migration, the Action Pane rule is run to identify redundant buttons. To complete this part of migration, you need to manually:

- Remove or move the code.
- Delete redundant controls in the application code.

NOTE

In the section below, we will provide examples of how to migrate and modify the code on modeled buttons that replicate system-defined buttons. However, in practice, before making changes similar to those made in this article, the code must first be evaluated with respect to the scenario to determine if it is still needed. First, fix the TODO for the DeleteCmdButton, which duplicates the system-defined Delete button.

1. In Visual Studio, find the TODO shown below, and then double-click the TODO.

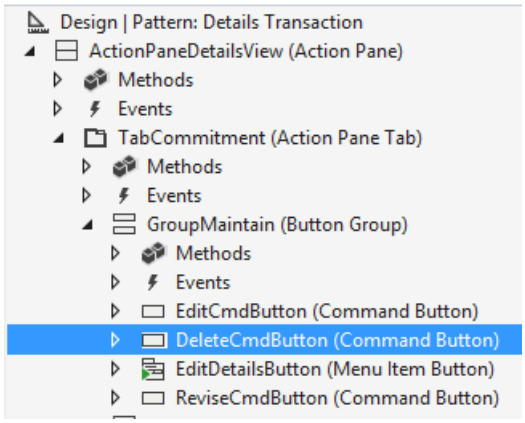


2. Replace the TODO and the line of code as shown below.

   - The state of the system-defined **Delete** button is controlled by the AllowDelete property on the firstmaster datasource. By setting AllowDelete to false, the delete task is kept from executing when the keyboard shortcut is used.

```
// Delete button
/* TODO: (Code Upgrade) [Action Pane Rule] Please consider moving all references to the form
task override method and remove the control: DeleteCmdButton */
deleteCmdButton.enabled(purchCommitmentHeader && purchCommitmentHeader.canDelete());
PurchCommitmentHeader_DS.allowDelete(purchCommitmentHeader &&
purchCommitmentHeader.canDelete());
```

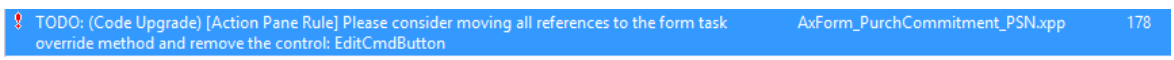
3. In the editor, find and remove DeleteCmdButton from the form design.



4. Press **Ctrl+S** to save the form.

   - Next, we will focus on the EditCmdButton that duplicates the system Edit button, handing the two TODOs associated with this button as well as removing this button.

5. In Visual Studio, find the TODO shown below, and then double-click the TODO.



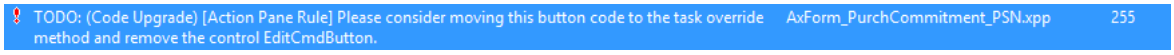
6. Because the visibility of the **Edit** button is controlled by the View/Edit mode of the form, you will need to modify this code so it sets that property. Replace the TODO and the line of code as shown in the following graphic.

```
/* TODO: (Code Upgrade) [Action Pane Rule] Please consider moving all references to the form task
override method and remove the control: EditCmdButton */
editCmdButton.enabled(purchCommitmentHeader && isInDraftOrUnderRevisionStatus &&
!isInWorkFlowReviewState && !isLineReferenced);

if(purchCommitmentHeader && isInDraftOrUnderRevisionStatus && !isInWorkFlowReviewState &&
!isLineReferenced)
{
    element.design().ViewEditMode(ViewEditMode::Auto);
}
else
{
    element.design().ViewEditMode(ViewEditMode::View);

}
```

7. Double-click the other TODO for this button.



8. Inspect the code on the modeled **Edit** button. This logic will need to be moved to the form's task() method.

```
[Control("CommandButton")]
class EditCmdButton
{
    /* TODO: (Code Upgrade) [Action Pane Rule] Please consider moving this button code to the task
override method and remove the control EditCmdButton. */
    void clicked()
    {
        if (purchCommitmentHeader.WorkflowApprovalState ==
            PurchCommitmentWorkflowApprovalState_PSN::Approved)
        {
            if (Box::yesNo(strFmt("@SPS2140", purchCommitmentHeader.CommitmentNumber),
                DialogButton::No) == DialogButton::Yes)
            {
                super();

                PurchCommitmentHeader_PSN::setWorkflowState(purchCommitmentHeader.RecId,
                    PurchCommitmentWorkflowApprovalState_PSN::NotSubmitted);
            }
        }
        else
        {
            super();
        }
    }
}
```

9. On the left side of the Visual Studio designer, right-click **Methods** > **Override**, and select **Task**, to add an override for the form's Task method.

10. Update the task method as shown below so that the code from above is triggered when the system-defined **Edit** button is clicked.

```
///
    ///
    ///
    ///
    ///
    public int task(int _taskId)
    {
        #Task
        int ret;

        switch (_taskId)
        {
            case #taskEditRecord:

                if (purchCommitmentHeader.WorkflowApprovalState ==
PurchCommitmentWorkflowApprovalState_PSN::Approved)
                {
                    if (Box::yesNo(strFmt("@SPS2140", purchCommitmentHeader.CommitmentNumber),
DialogButton::No) == DialogButton::Yes)
                    {
                        ret = super(_taskId);

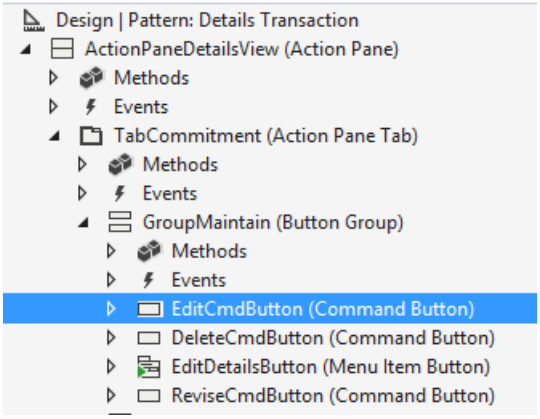
                        PurchCommitmentHeader_PSN::setWorkflowState(purchCommitmentHeader.RecId,
PurchCommitmentWorkflowApprovalState_PSN::NotSubmitted);
                    }
                }
                else
                {
                    ret = super(_taskId);
                }

                break;

            default:
                ret = super(_taskId);
                break;
        }

        return ret;
    }
```

11. In the Editor, find and remove the **EditCmdButton** from the form design.



12. Press **Ctrl+S** to save the form.

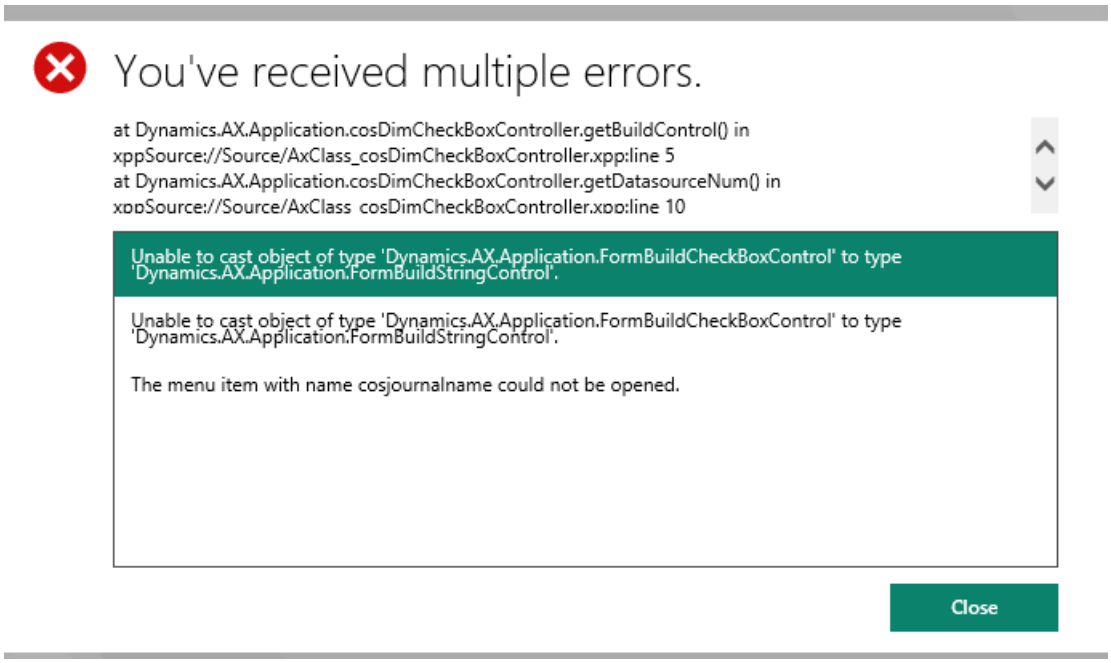
13. Press **Ctrl+F5** to view the form. Notice the **Delete** and **Edit** buttons in the **Commitment** tab have been removed.

Resolve casting exceptions

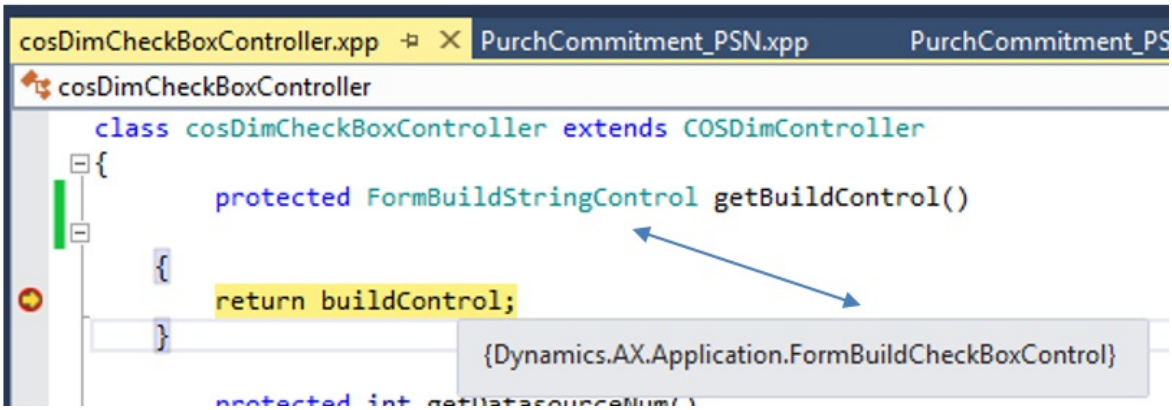
In Finance and Operations, X++ is completely intermediate-language (IL) based and therefore has a stricter runtime type behavior than the interpreted Dynamics AX2 012. This stricter runtime type behavior can generate exceptions in migrated Dynamics AX 2012 R3 metadata. It is likely you will encounter these exceptions during your migration. The casting exceptions can be raised in different runtime scenarios, such as down-casting, casting runtime to design time objects, and side-casting. In the section below, we will walk through an example where a form, CosJournalName, is generating controls at runtime, and has a type mismatch which causes a .NET exception because it is strongly typed.

Example: Side-casting exception

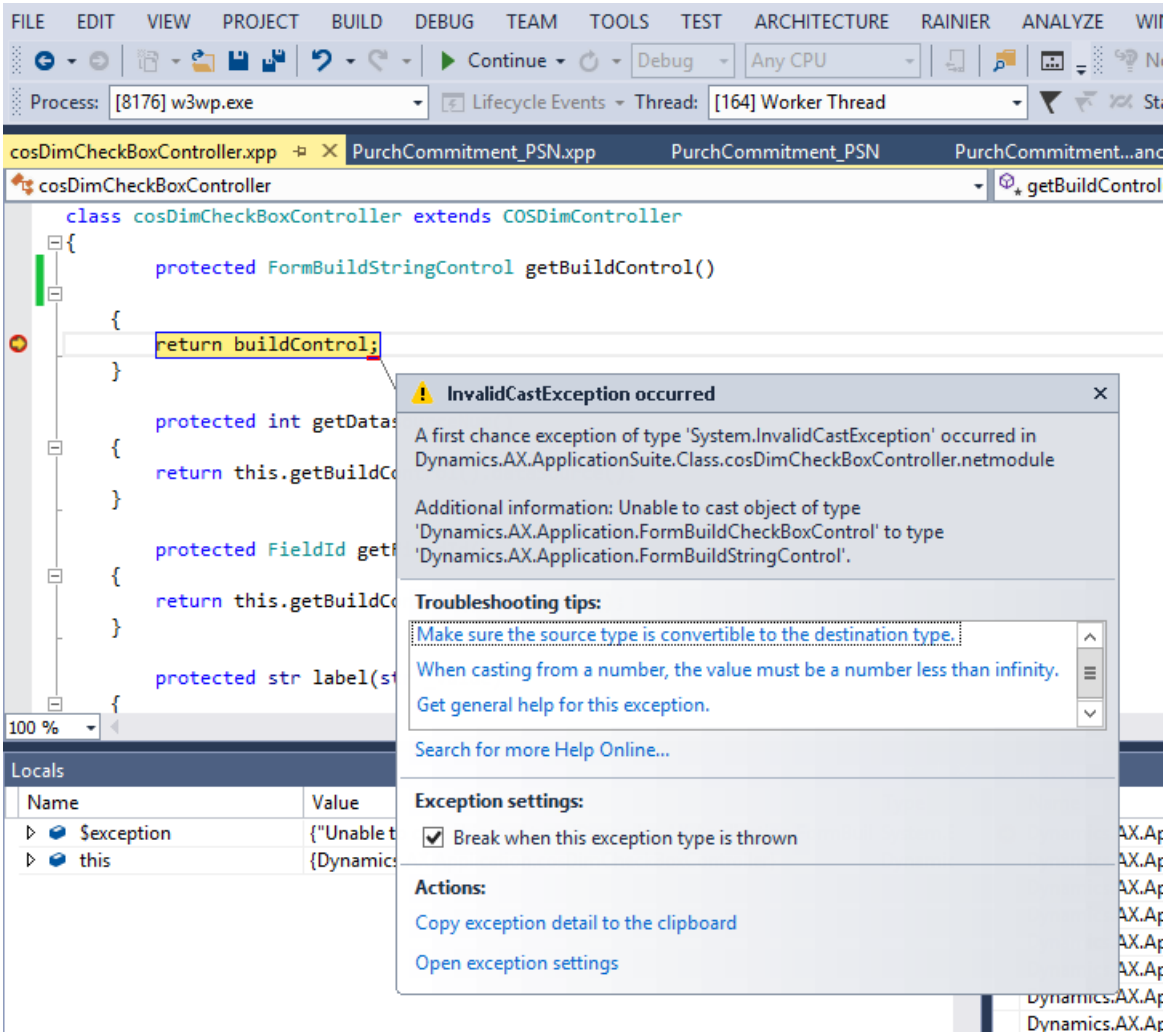
1. In Visual Studio, select and right-click **Project Properties**, and verify that USMF is the default company.
2. Add the display menu item CosJournalName to your project, and set the menu item as your Startup object.
3. Add the CosJournalName form to your project.
4. Add the cosDimCheckBoxController class to your project.
5. Rebuild your project.
6. Press **Ctrl+F5** to run the form.
7. Note that you will get an exception, similar to the following, when running the form.



8. Right-click the class, cosDimCheckBoxController, and then select **View Code**.
9. Set a breakpoint on the cosDimCheckBoxController::getBuildControl().
10. Press **F5**.
    - The breakpoint will be hit. This is where the casting error occurs. The reason for the casting error is because we are trying to return a control of type: FormBuildCheckboxControl and the object is expecting FormBuildStringControl.
11. Hover over the buildcontrol to see the type and notice the differences.



12. Press **F10** to hit the exception.

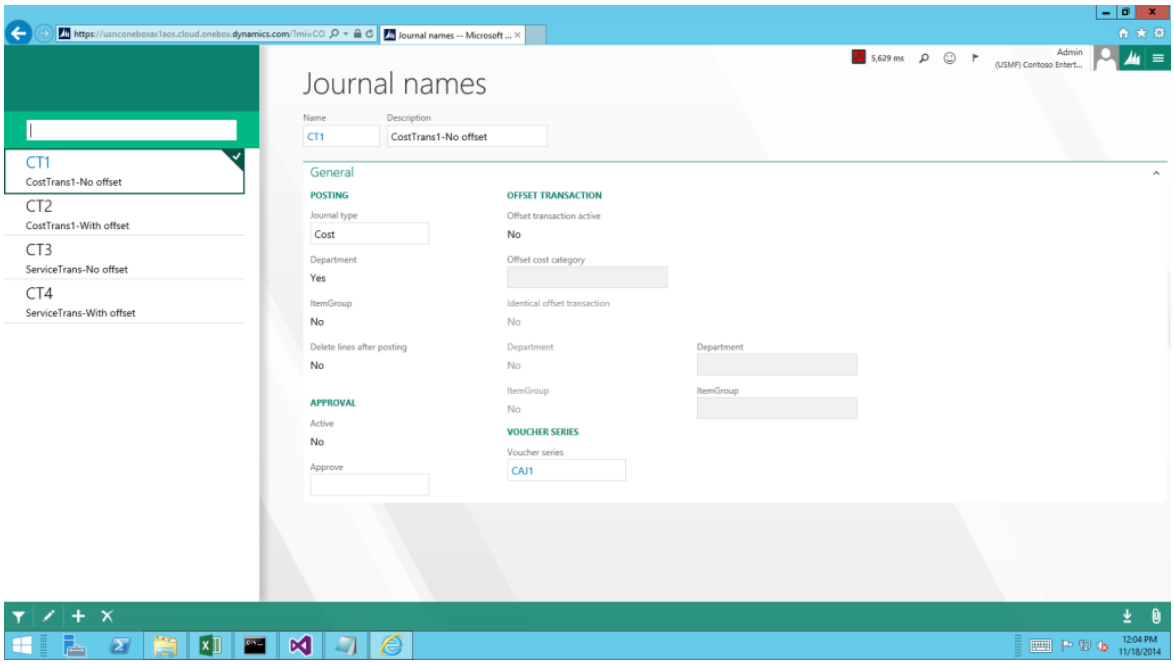


13. Stop debugging.

14. To fix the exception, change the method declaration from FormBuildStringControl to FormBuildCheckBoxControl.

```
protected FormBuildStringControl getBuildControl()
protected FormBuildCheckBoxControl getBuildControl()
```

15. Rebuild the project, and press **Ctrl+F5**. The form should open successfully because the casting error is resolved.



Migrating context menus and mouse double-click code

Refer to these topics to migrate code Dynamics AX 2012 that deals with context menus and mouse double-click actions.

- Code migration - Context menu code
- Code migration - Mouse double-click logic

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Configure the Azure DevOps mapping during code migration

2/18/2021 • 2 minutes to read • Edit Online

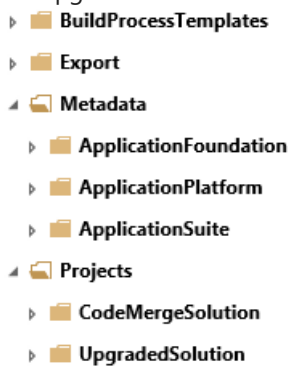
This tutorial shows how to map your development box to the Azure DevOps project after the Lifecycle Services (LCS) code upgrade service has completed.

The LCS code upgrade service automatically checks your upgraded code into Azure DevOps. You will then need to map your development box to the upgrade folder/branch in your Azure DevOps project (The name of the upgrade folder/branch depends on the version you migrated to). Within your upgraded folder, you will find three folders:

- Export
- Metadata
- Projects

Key concepts

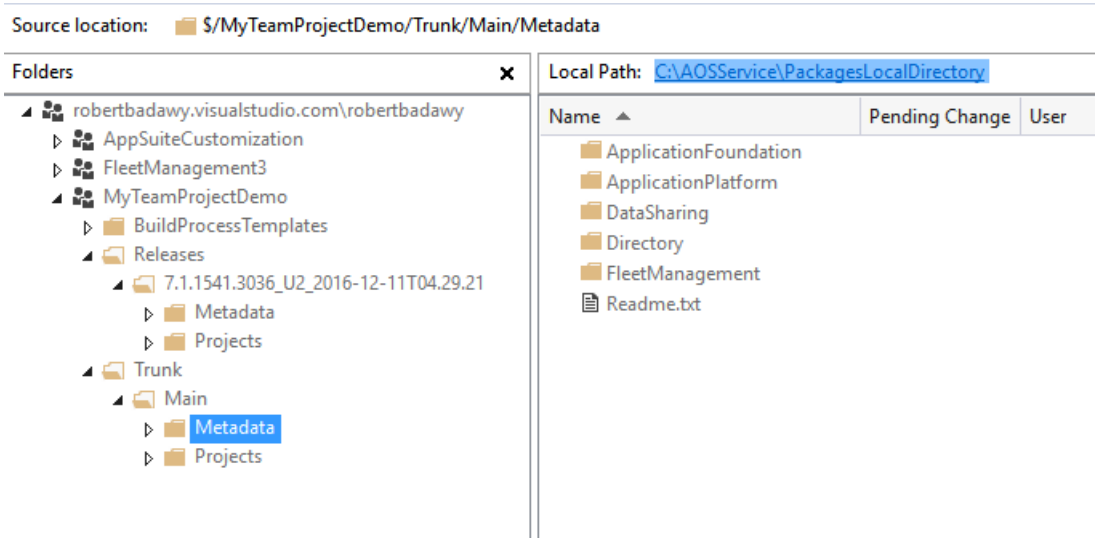
- **Export** is the project that contains the XML files after exporting from Microsoft Dynamics AX 2012. This project is your metadata in XML format before it is upgraded. This project is only relevant if you are upgrading from Dynamics AX 2012.
- **Metadata** is your upgraded code (metadata XML file).
- **Projects** are two solutions that you can use during upgrade. One solution, CodeMergeSolution, is the solution that contains projects with the elements that have conflicts and need to be resolved. The other solution, UpgradedSolution, contains a collection of projects, one for each upgraded model. For example, in Azure DevOps, you should see something like the following structure.



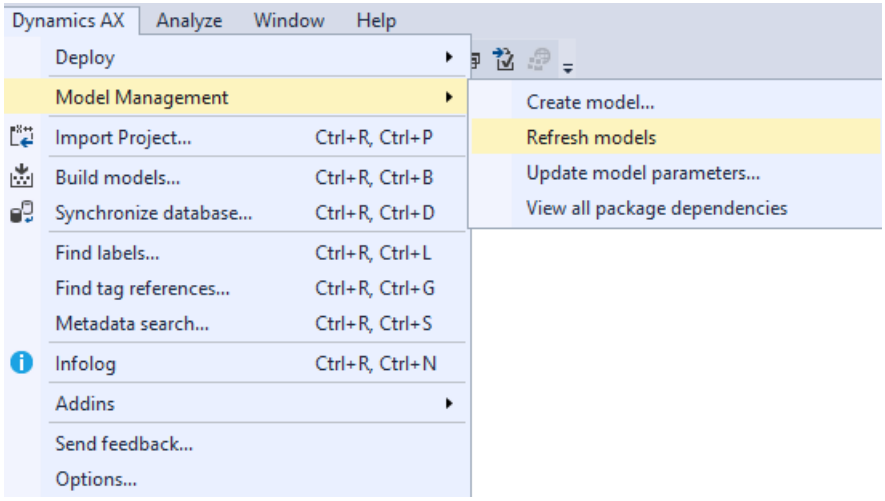
Map Azure DevOps to your development box

1. In Visual Studio, connect to your account by going to **Team explorer** > **Select Team Projects** > **Servers** > **Add.**
2. Enter the URL to your team project. Select **Close**.
3. Make sure the Azure DevOps account shows up. On the right, choose the project that you want to work on. Select **Connect**.
4. Now you need to map your workspace to the Azure DevOps folders. Go to the **Source Code Explorer** and do this mapping:
   a. Projects >
      - For **Visual Studio 2015** : C:\Users\<username>\Documents\Visual Studio 2015\Projects
      - For **Visual Studio 2017** or newer : C:\Users\<username>\source\repos

b. Metadata > C:\AOSService\PackagesLocalDirectory
   - On cloud VMs, this folder is located on the I:\, J:\ or K:\ drive
   - On earlier versions, this folder is C:\packages
   - **Important**:
     - If you are migrating from Dynamics AX 2012 R3 or earlier, you will be mapping to the metadata folder under the **Main** branch.
     - If you are migrating between two product versions, you will be mapping to the metadata folder under one of the **Releases** branch.



After you have mapped these folders, you can synchronize the code to your local box. Right-click **Metadata** and select **Get latest**. Similarly synchronize the Projects folder. After synchronizing the metadata folder, refresh your models in Visual Studio from **Finance and Operations** > **Model Management** > **Refresh Models**.



You are now ready to open your projects, resolve conflicts, build, test, and complete your code migration.

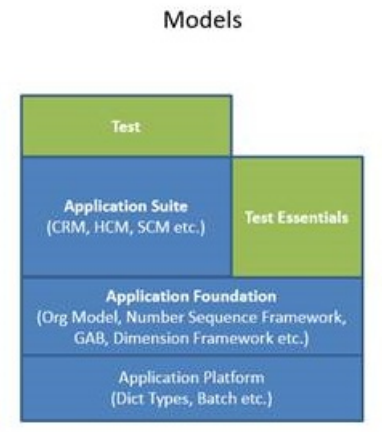
Model split

2/18/2021 • 3 minutes to read • Edit Online

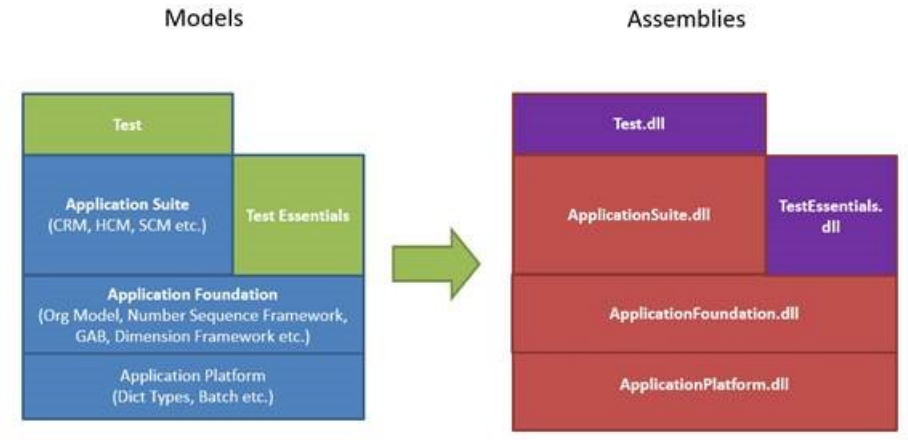
This topic explains the split of the stack into three main models - the Application Platform, the Application Foundation, and the Application Suite.

Overview

Developing modular code is the driving force behind the model split. Splitting the stack into multiple models provides many benefits, including faster compile time and a greater distinction between partner's IP in production. There are three main models: the **Application Platform**, the **Application Foundation**, and the **Application Suite**.



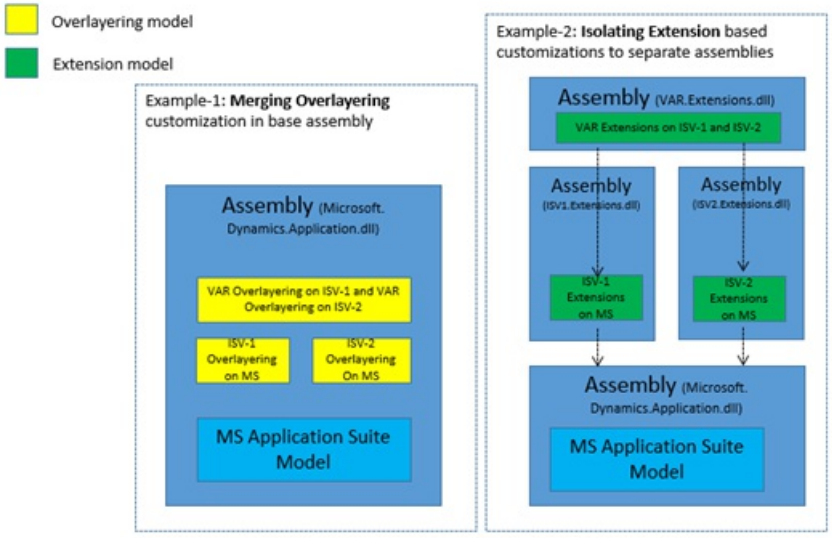
The **Application Platform** is the lowest model and contains the lowest level elements that interface with the kernel. The **Application Object Server** (**AOS**) can be started with only the **Application Platform**. The **Application Foundation** sits atop the **Application Platform** and contains framework functionalities that are shared by all applications. Finally, the **Application Suite** sits atop the **Application Foundation** and contains application specific elements. The Model Breakdown table in the appendix provides examples of components in each of these models. Each model is compiled into its own assembly with dependencies on lower layer model assemblies. The **Application Platform** does not depend on any other models. This implies a direct mapping of the model to an assembly.



Developing in the modular stack allows changes to be made in the **Application Suite** and compiled without touching the rest of the stack. Only models with new changes need to be compiled, greatly reducing compile time. More information can be found in the "Model Breakdown" section at the end of this article.

Customizing models

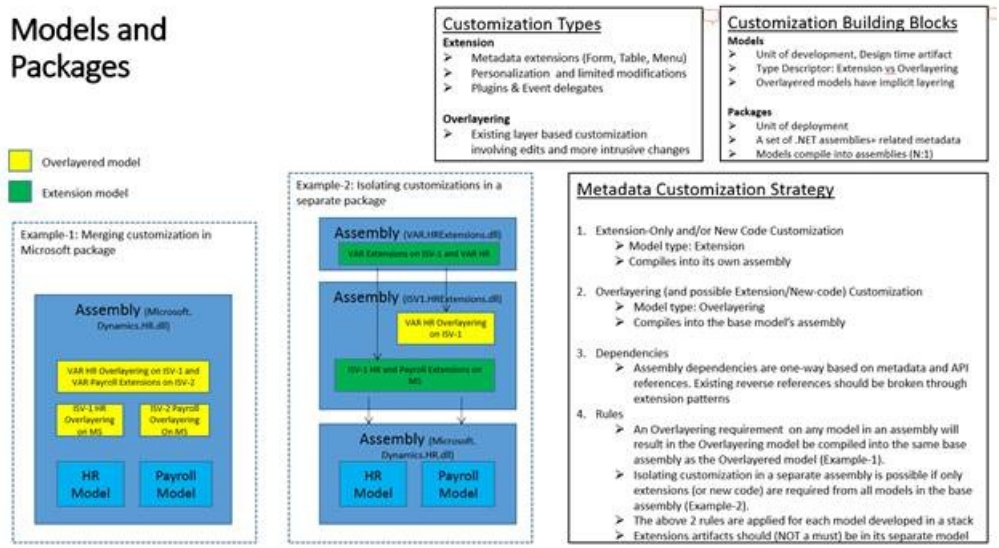
There are two methods for customizing: overlayering and extensions. Overlayering allows for changes to be made at multiple layers that alter, or overlay, elements in models at lower levels. Extensions allow for new elements to be added or code to be attached to element events or plug-in points. The type of customization used impacts how a model will be compiled and ultimately packaged. One or more models are compiled into an assembly. An assembly, its non-code metadata, and its compiled artifacts, form a package. A package is an independent deployable unit. A model that contains only extension customizations can be compiled into its own assembly and be deployed in its own package. A model that contains any overlayering must be compiled into the assembly based on the overlayed model.



Using extensions has several advantages, including:

- For application lifecycle management purposes, you need to manage only your extension artifacts.
- Building a customization does not require you to recompile the entire application.
- In the cloud, Microsoft can install, patch, upgrade, and change internal APIs without affecting your customizations.
- You can service your solutions independently without concerns about other customizations.

There are currently support code extensions, table extensions, form extensions, menu extensions, and enum extensions. The Extensions section in Customize through extension and overlayering and Customize model elements through extension provide a more detailed explanation on how to use extensions. Extensions should be used on supported elements wherever possible and are best applied when no change to existing Microsoft code is needed. A change to mask a method's functionality requires overlayering to change the code itself. Overlayering should be in areas not covered by extensions and when the customization alters the base functionality. The illustration below summarizes differences between the two customization strategies.



Model breakdown

Model	Concept
Application Platform	The Application Platform interfaces to kernel functionalities that are application logic agnostic. The AOS can be started with just this model. • AIF base objects • Batch • Form base objects • RunbaseSysOperations* base objects • DictXX objects • Appl, Info, Global, ClassFactory • Data access objects • Helper Classes • ENUM/EDT/Macros/ConfigKey/LicenseCode

Application Foundation	• Application foundation features:
	  ◦ Dimension framework
	  ◦ Global Address Book
	  ◦ Number Sequence
	  ◦ OrgModel
	• Feature areas:
	  ◦ Business Intelligence
	  ◦ Reports
	  ◦ Upgrade framework
	  ◦ Security
	  ◦ E-Signature
	  ◦ Data Import/Export
	  ◦ Workflow
	• SYS objects:
	  ◦ Best Practices
	  ◦ CheckList
	  ◦ Policy
	  ◦ RecordTemplate
	• Miscellaneous:
	  ◦ Currency
	  ◦ Unit of Measure

Application Suite	• Supply Chain Management • Human Capital Management • Professional Services, etc.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

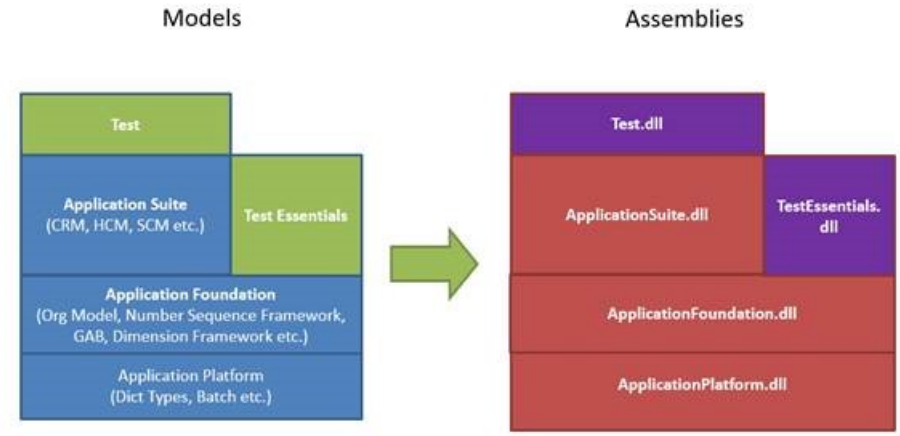
Solve dependencies among models by using delegates during code migration

2/18/2021 • 6 minutes to read • Edit Online

This topic explains how delegate methods serve as a means for defining a contract between the delegate instance and the delegate handler.

Overview

Finance and Operations is split into several models, with each model in separate package. The principal 3 models are Application Platform, Application Foundation, and Application Suite. With the model split, a hierarchy has been created where a higher model can take dependencies and access elements in the models below, but not in models above. For example, in this setup, Application Suite has full access to its elements, Application Foundation's elements, and Application Platform's elements. Application Foundation can access its own elements and those of Application Platform. Finally, Application Platform can only access its own elements. To learn about models and packages, see Models and packages.



While the model split provides many benefits, it creates a problem when trying to access elements defined in higher models. Delegates are the recommended method for accessing elements in higher models from a lower model. Delegates are very similar to events in that when a delegate instance is invoked, a handler with compatible signature code is executed. This permits higher layer code, the handler, to be called by lower layer code, the delegate instance.

Create delegates and handlers

A delegate declaration must have three things:

- The delegate keyword
- Type void
- Empty method

Delegate methods serve as a means for defining a contract between the delegate instance and the delegate handler. A delegate takes no action itself. This is enforced by having a void type and having no code in the method.

```
delegate void applyDiscountDelegate(real _receiptTotal, EventHandlerResult _result)
{
}
```

Adding the **SubscribesTo** keyword to a method creates a static delegate handler. **SubscribesTo** requires the class name of the delegate, and the string name of the delegate method.

In order for a delegate to be properly handled, the delegate method declaration, the delegate instance, and the delegate handler must have the *same* method signature. For example, the delegate instance below takes two inputs, a real number and an EventHandlerResult, matching the delegate declaration and handler signatures above.

```
[SubscribesTo(classStr(SimpleTax), delegateStr(SimpleTax, applyDiscountDelegate))]
public static void applyDiscountDelegateHandler(real _receiptTotal, EventHandlerResult _result)
{
    real discountedTotal = _receiptTotal * (1-DiscountRate);
    _result.result(discountedTotal);

}
```

Due to the fact that delegates do not have a return value, an EventHandlerResult is passed as a parameter to provide access to the needed result value after the delegate has returned. This topic focuses on static delegate handlers using the SubscribesTo. The delegate functionality from Dynamics AX 2012 remains. How to use X++ Delegates in Dynamics AX 2012 is a great blog post on MSDN by Microsoft developer Marcos Calderon on delegate concepts in Dynamics AX 2012. These concepts still apply.

Example scenarios

Overlaying an existing delegate

In many cases where delegates are needed, the code that was formerly overlayed has already been moved to a delegate handler by Microsoft. In these instances, Microsoft created delegates that can be leveraged and the code can be overlayed in a similar manner in the delegate handler. In this scenario, an Independent Software Vendor (ISV) is migrating code from Dynamics AX 2012 R3 where they have overlayed the showSalesTax() method in the LogisticsEntityPostalAddressFormHandler class. After migration, the CodeUpgrade project will contain the LogisticsEntityPostalAddressFormHandler with the *Your Solution*, *Microsoft AX 2012,* and *Microsoft AX* sections to resolve for the showSalesTax() method. The commented Your Solution section shows that the showSalesTax() method was overlayed by adding an additional table to approve showing sales tax from. This overlay is shown between the <isv> tags circled in red below.

```
////Your Solution
///
/*
public boolean showSalesTax()
{
    boolean showTaxField;

    switch (this.getCallerRecord().TableId)
    {
        case tableNum(CustTable) ,
        tableNum(VendTable) ,
        tableNum(smmBusRelTable) ,
        tableNum(CompanyInfo) ,
        tableNum(InventSite) ,
        tableNum(InventLocation),
        //<isv>
        tableNum(MYISVTable),
        //</isv>
        tableNum(HcmWorker):
            showTaxField = true;
            break;

        default:
            showTaxField = false;
    }
    return showTaxField;

}
*/
```

When comparing this overlay with the code from Dynamics AX 2012, this is a simple change. The overlay has added an additional table to the switch statement.

```
////Microsoft AX 2012
/*
public boolean showSalesTax()
{
    boolean showTaxField;

    switch (this.getCallerRecord().TableId)
    {
        case tableNum(CustTable) ,
            tableNum(VendTable) ,
        tableNum(smmBusRelTable) ,
        tableNum(CompanyInfo) ,
        tableNum(InventSite) ,
        tableNum(InventLocation),
        tableNum(HcmWorker):
            showTaxField = true;
            break;

        default:
            showTaxField = false;
    }
    return showTaxField;

}
*/
```

However, the section for Finance and Operations does not appear to resemble either of the Dynamics AX 2012 code snippets.

```
////Microsoft AX 7
/*
public boolean showSalesTax()
{
    boolean showTaxField = false;

    EventHandlerResult result = new EventHandlerResult();

    this.showSalesTax_delegate(this.getCallerRecord().TableId, result);

    if (result.result() != null)
    {
        showTaxField = result.result();
    }

    return showTaxField;
}
*/
```

Upon deeper inspection, the code is calling a delegate method, showSalesTax_delegate().

```
this.showSalesTax_delegate(this.getCallerRecord().TableId, result);
```

The use of a delegate implies that code has been moved to another location. The showSalesTax_delegate() has been declared in the Application Foundation and handled in the Application Suite. To view the code that has been moved, find the delegate handler. The **Finding Delegates and Handlers** section contains methods to locate delegates and handlers. After finding the delegate handler method in the Application Suite, we see the code that has been moved from the showSalesTax() method. The same overlayered changes applied in Dynamics AX 2012 can be applied in the delegate handler.

```
[SubscribesTo(classStr(LogisticsEntityPostalAddressFormHandler),
delegatestr(LogisticsEntityPostalAddressFormHandler, showSalesTax_delegate))]
public static void onShowSalesTax_delegate(TableId _callerTableId, EventHandlerResult _res)
{
    switch (_callerTableId)
    {
        case tableNum(CustTable):
        case tableNum(VendTable):
        case tableNum(smmBusRelTable):
        case tableNum(CompanyInfo):
        case tableNum(InventSite):
        case tableNum(InventLocation):
        case tableNum(HcmWorker):
            _res.result(true);
            break;

        default :
            _res.result(false);
    }
}
```

After adding the new table to the switch statement in the delegate handler, the code will function as it did in Dynamics AX 2012.

```
[SubscribesTo(classStr(LogisticsEntityPostalAddressFormHandler),
delegatestr(LogisticsEntityPostalAddressFormHandler, showSalesTax_delegate))]
public static void onShowSalesTax_delegate(TableId _callerTableId, EventHandlerResult _res)
{
    switch (_callerTableId)
    {
        case tableNum(CustTable):
        case tableNum(VendTable):
        case tableNum(smmBusRelTable):
        case tableNum(CompanyInfo):
        case tableNum(InventSite):
        case tableNum(InventLocation):
        //<isv>
        case tableNum(MYISVTable):
        //</isv>
        case tableNum(HcmWorker):
            _res.result(true);
            break;

        default :
            _res.result(false);
    }
}
```

Adding a new delegate

In this scenario, we will modify an existing tax calculation method that resides in the Application Foundation to account for discounts created in the Application Suite. The following class in the Foundation layer calculates the tax based on the gross total.

```
public class SimpleTax
{

    public real ReceiptTotal;

    public real TaxRate;

    public real calculateTotalTax()
    {
        //   calculates tax on gross total.
        real totalTax;

        totalTax = this.ReceiptTotal * this.TaxRate;

        return totalTax;

    }
```

In the Application Suite, we have introduced the notion of discounts by adding a ProductDiscount class that contains the current discount.

```
public class ProductDiscount
{

    public static real DiscountRate;

}
```

The TaxCalculator class, in the lower Foundation layer, does not have access to the DiscountRate in the Suite layer and must use a delegate to update receipt total to use in the tax calculation. In the SimpleTax class, we create a delegate method, applyDiscountDelegate, with the state information that is needed by the handler in the signature. A delegate method is always empty because its only purpose is to define the contract between the delegate instance and the handler.

```
delegate void applyDiscountDelegate(real _receiptTotal, EventHandlerResult _result)
{
}
```

> **NOTE**
>
> The signature for the delegate declaration, the delegate instance, and the delegate handler must match. We now have to create an instance of the delegate at the point in the code where we would like the delegate handler to be run. The changes in between the <isv> tags represent the added code.

```
public real calculateTotalTax()
{
    //   calculates tax on gross total.
    real totalTax;

    //<isv>
    EventHandlerResult result;
    this.applyDiscountDelegate(this.ReceiptTotal, result);
    this.ReceiptTotal = result.result();
    //</isv>

    totalTax = this.ReceiptTotal * this.TaxRate;


    return totalTax;

}
```

With the delegate in place, we now add a handler method in the Application Suite layer that has access to the discount information.

```
[SubscribesTo(classStr(SimpleTax), delegateStr(SimpleTax, applyDiscountDelegate))]
public static void applyDiscountDelegateHandler(real _receiptTotal, EventHandlerResult _result)
{
    real discountedTotal = _receiptTotal * (1-DiscountRate);
    _result.result(discountedTotal);

}
```

Using the SubscribesTo keyword, we tie the applyDiscountDelegateHandler method as a handler to the applyDiscountDelegate delegate.

> **NOTE**
>
> There can be more than one handler per delegate. There is **not** a defined order in the processing of handler methods. If order is important, delegate handler pairs should be chained together. With the final classes below, when the calculateTotalTax() method is run, the applyDiscountDelegate is fired and handled, updating the receiptTotal to provide an accurate tax calculation.

Full Code

SimpleTax class in the Application Foundation Layer

```
public class SimpleTax
{

    public real ReceiptTotal;

    public real TaxRate;

    public real calculateTotalTax()
    {
        //   calculates tax on gross total.
        real totalTax;

        //<isv>
        EventHandlerResult result;
        this.applyDiscountDelegate(this.ReceiptTotal, result);
        this.ReceiptTotal = result.result();
        //</isv>

        totalTax = this.ReceiptTotal * this.TaxRate;


        return totalTax;

    }

    //<isv>
    delegate void applyDiscountDelegate(real _receiptTotal, EventHandlerResult _result)
    {
    }
    //</isv>
}
```

ProductDiscount class in the Application Suite layer

```
//<isv>
public class ProductDiscount
{

    public static real DiscountRate;


    [SubscribesTo(classStr(SimpleTax), delegateStr(SimpleTax, applyDiscountDelegate))]
    public static void applyDiscountDelegateHandler(real _receiptTotal, EventHandlerResult _result)
    {
        real discountedTotal = _receiptTotal * (1-DiscountRate);
        _result.result(discountedTotal);

    }

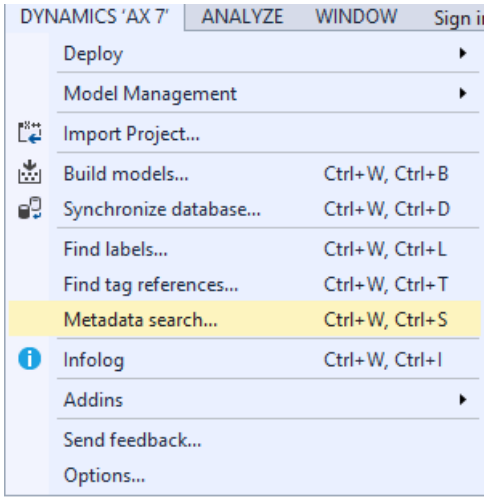
}
//</isv>
```

Find delegates and handlers

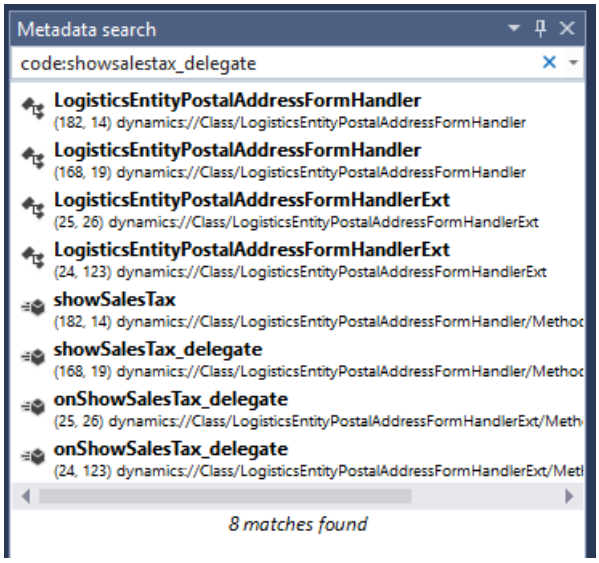
There are three key ways to find delegates and handlers

- Metadata search
- Class references
- SubscribesTo references

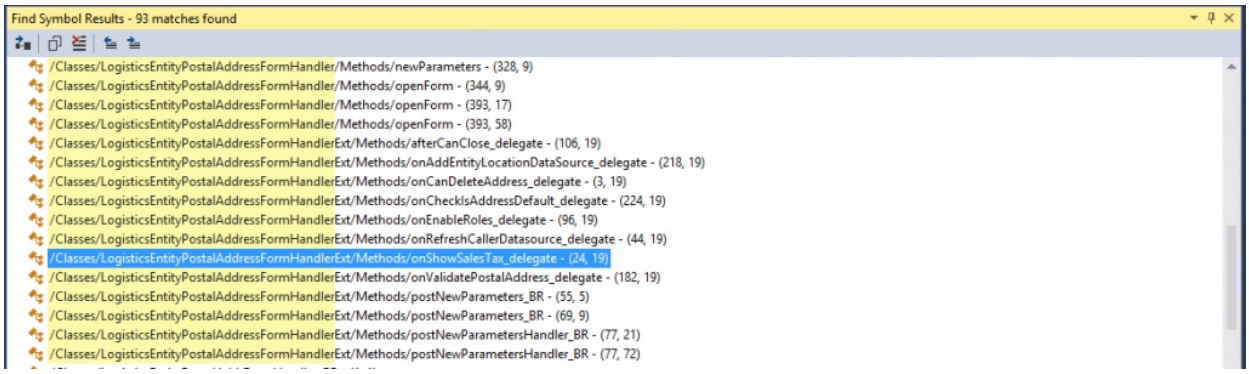
The Metadata search tool, described on the Metadata search in Visual Studio page, is the best way to find either delegates or their handlers. In Visual Studio, go to **Dynamics 365 > Metadata Search** to open the metadata search tool.



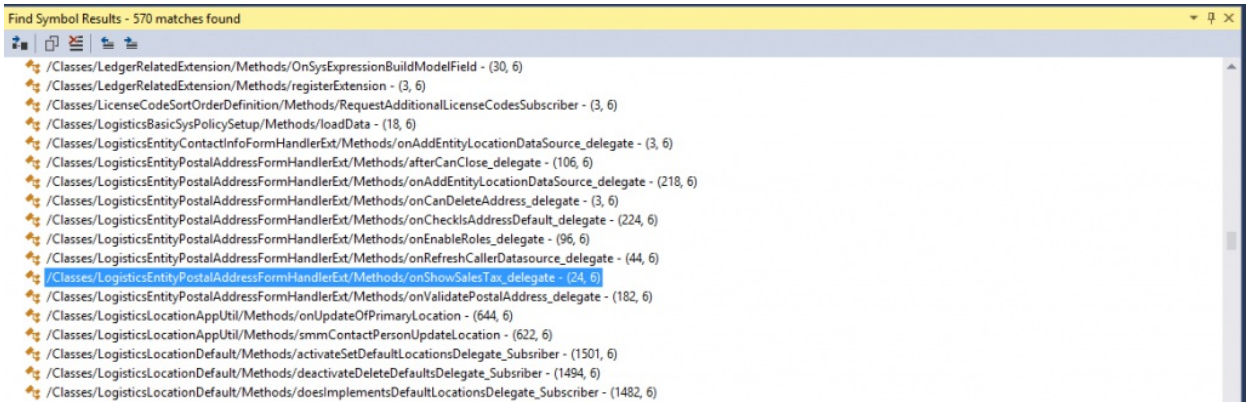
In the search field, type "code:<delegate name>" which will restrict the search to code and find any use of the delegate name, returning both the delegate and handler. Metadata search will search the entire code base and may take some time to complete, but will return any use of the search term in code.



Methods two and three can be used in parallel to the metadata search. The class where a delegate is defined can also serve as a means to narrow down the search for a delegate or handler. The SubscribesTo keyword requires the class name where the delegate was defined. Visual Studio's find references (right-click the class name > find references) will return a list of files that reference the class. This list will include both the class definition where the delegate is declared and the handler referencing the class. Finding class references is not a perfect method and will require some manual searching through class references. However, it produces a smaller subset of files and can be faster than a metadata search.



Similar to finding class references, finding all references can be done on the SubscribesTo keyword. The resulting list will include all static delegate handlers. Manually going through this list provides another means for finding static delegate handlers. This will not return dynamically declared delegate handlers that do not use the SubscribesTo keyword.



Process for moving to the latest update of Finance and Operations

2/18/2021 • 2 minutes to read • Edit Online

This topic explains the process of updating or upgrading to the latest release of Finance and Operations. It describes the overall process and supported scenarios, but it doesn't provide detailed instructions for every step of the process.

For information about the contents of each release of Finance and Operations, see What's new or changed in Finance and Operations home page.

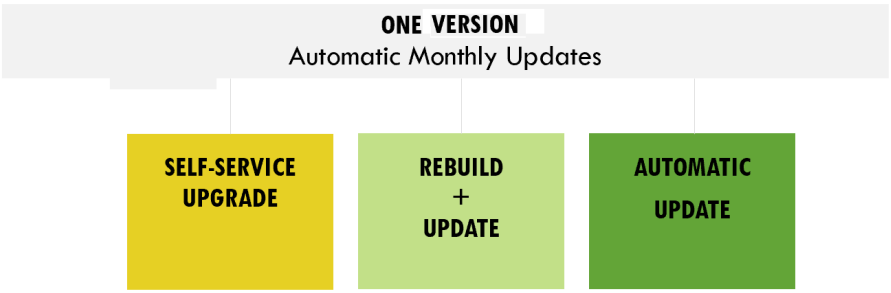
For information about One Version service updates, see the One Version service updates overview.

> **NOTE**
>
> For those looking to upgrade to Finance and Operations from Microsoft Dynamics AX 2012, please see Upgrade from AX 2012 to Finance and Operations.

Definitions

- **Upgrade** – The process of moving from one official release of Finance and Operations to the next release, for source environments prior to version 8.0. Some examples are the move from 7.1 to 7.3, or from 7.3 to 10.0.1. The process involves setup of a free sandbox environment, code upgrade, and data upgrade.
- **Update** – The process of applying a binary package to an environment to move it from one official release of Finance and Operations to the next release, for source environments starting with version 8.0. This process has lower downtime requirements and doesn't involve data upgrade.

Paths to One Version



There are three primary paths to get to the latest version of Finance and Operations. Each path is referenced below with a link to detailed steps.

Self-service upgrade

Applicable starting version: Microsoft Dynamics 365 for Finance and Operations 7.0 (RTW), 7.1 (1611), 7.2 (July 2017), 7.3.
Scope: Complex
This path involves code refactoring to Extensions, and Data Upgrade in a DevTest, Sandbox, and eventually a Production environment.

Self-service upgrade to the latest version.

Rebuild and update

Applicable starting version: Microsoft Dynamics 365 for Finance and Operations 8.0
Scope: Moderate
This path involves removing Microsoft X++ hotfixes, and creating a merged update package.

Update environments from version 8.0 to 10.0.X.

Automatic update

Applicable starting version: Finance and Operations 8.1.0+
Scope: Simple
This path involves configuring your project for continuous updates.

Configure service updates through Lifecycle Services (LCS).

Self-service upgrade to the latest version

2/18/2021 • 13 minutes to read • Edit Online

IMPORTANT

This topic applies to the following starting versions:

- Microsoft Dynamics 365 for Operations version 1611 (November 2016) (also known as version 7.1)
- Microsoft Dynamics 365 for Finance and Operations, Enterprise edition (July 2017) (also known as version 7.2)
- Microsoft Dynamics 365 for Finance and Operations, Enterprise edition 7.3

In this tutorial, you will learn how to perform these tasks:

- Understand which version to select.
- Refactor your customizations as extensions.
- Run the data upgrade in a development environment.
- Do a self-service upgrade in a sandbox user acceptance testing (UAT) environment.
- Do a self-service upgrade in a production environment.

Understand which version to select for upgrade

To align the self-service upgrade process to support continuous updates, each new release will cause the oldest release version to be discontinued.

For example, you have application version 7.3 with Platform update (PU) 23. Currently, the supported upgrade versions are 8.1.3 with PU 23, 10.0.0 with PU 24, and 10.0.1 with PU 25. When the next release, 10.0.2 with PU 26, is made generally available, it will be added to the available upgrade options, and 8.1.3 with PU 23 will be removed.

Because of this continuous process of adding a new version and removing the oldest version, we recommend that customers upgrade to the latest version that is available. In that way, you have two months in which you can upgrade your sandbox environment and then later upgrade your production environment to the same version.

If you choose to upgrade your sandbox environment to version 8.1.3 with PU 23 and Microsoft then releases version 10.0.2 with PU 26, so that version 8.1.3 with PU 23 is removed as an upgrade option, **you will be blocked** from upgrading your production environment. In this case, you must start over in the sandbox environment and upgrade to a newer supported version.

Targeted release schedule

NOTE

The dates in this table are subject to change. Upgrades are available a minimum of two weeks and a maximum of six weeks after the date of general availability (GA) for new customers. In addition, **support for upgrades that have a target version of application 7.3 ends in March 2020**.

SELECTABLE VERSIONS	GA OF THE LATEST VERSION FOR NEW CUSTOMERS	GA OF THE LATEST VERSION FOR UPGRADE
7.3 with PU 23 – PU 25 8.1.3 with PU 23 – 10.0.1 with PU 25	Week of April 8, 2019	Week of April 29, 2019
7.3 with PU 24 – PU 26 10.0.0 with PU 24 – 10.0.2 with PU 26	Week of May 13, 2019	Week of May 27, 2019
7.3 with PU 25 – PU 27 10.0.1 with PU 25 – 10.0.3 with PU 27	Week of June 10, 2019	Week of June 24, 2019
7.3 with PU 26 – PU 28 10.0.2 with PU 26 – 10.0.4 with PU 28	Week of July 8, 2019	Week of July 29, 2019
7.3 with PU 27 – PU 29 10.0.3 with PU 27 – 10.0.5 with PU 29	Week of September 17, 2019	Week of September 30, 2019
7.3 with PU 28 – PU 30 10.0.4 with PU 28 – 10.0.6 with PU 30	Week of October 11, 2019	Week of October 28, 2019
7.3 with PU 29 – PU 31 10.0.5 with PU 29 – 10.0.7 with PU 31	Week of November 29, 2019	Week of December 30, 2019
7.3 with PU 30 – PU 32 10.0.6 with PU 30 – 10.0.8 with PU 32	Week of January 17, 2020	Week of February 17, 2020
7.3 with PU 31 – PU 33 10.0.7 with PU 31 – 10.0.9 with PU 33	Week of March 3, 2020	Week of March 30, 2020
10.0.8 with PU 32 – 10.0.10 with PU 34	Week of April 8, 2020	Week of April 27, 2020

Refactor your customizations as extensions

To prepare for upgrade, you must refactor any customizations that were overlays as extensions. We recommend that you deploy a new development environment on the latest version, create a new branch in version control, and follow the guidance in Migrate from overlayering to extensions.

If you have no overlays and are already using extension for 100 percent of your customizations, we still recommend that you create a new branch for the upgrade effort in version control. If any Microsoft X++ hotfixes are installed, you must delete them from version control, because they aren't applicable to the latest version.

Run the data upgrade in a development environment

Run the data upgrade process on a copy of your source database. If your environment is already live in production, the source database is a copy of the production database. Otherwise, it's your most current database that is running the old version.

Run this process in the development environment that is running the release that you're upgrading to. This step is a validation process that is done by a developer. It helps the developer verify that the data upgrade can be successfully completed by using the specific set of customizations in the environment, without requiring any manual intervention.

To make a copy of your production database, follow the steps in Export a copy of the standard user acceptance

testing (UAT) database.

To run the data upgrade process, follow the steps in Upgrade data in development or demo environments.

> **IMPORTANT**
>
> - Data upgrade in a development environment is a required step. It helps reduce the risk of extended downtime and upgrade errors later, when you upgrade sandbox UAT and production environments.
> - Several application hotfixes might be required before you can upgrade data. Before you redeploy your existing development environment, verify whether these hotfixes are required. Install the required hotfixes, and check them in to Microsoft Azure DevOps. This step can be completed only in the old version of your development environment. For a list of the hotfixes that are required in various situations, see Upgrade data in development or demo environments.

Upgrade your Tier 2+ Standard Acceptance Test sandbox environment

When you've completed the code upgrade and have been able to do an end-to-end data upgrade in your development environment without having to manipulate data in Microsoft SQL Server, you can begin the process in your sandbox environment.

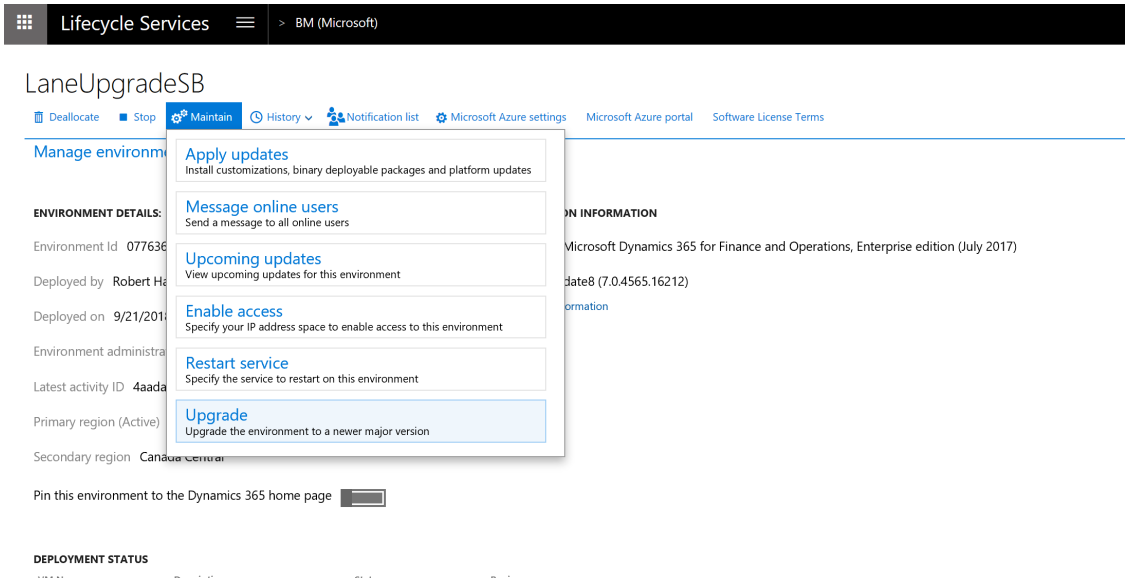
Prerequisite

Before you begin your upgrade, we highly recommend that you make sure that your sandbox environment has the latest production data. If the data set is up to date, you can have more confidence that the upgrade will work in the production environment. To complete this step, use the Refresh for training purposes tutorial.

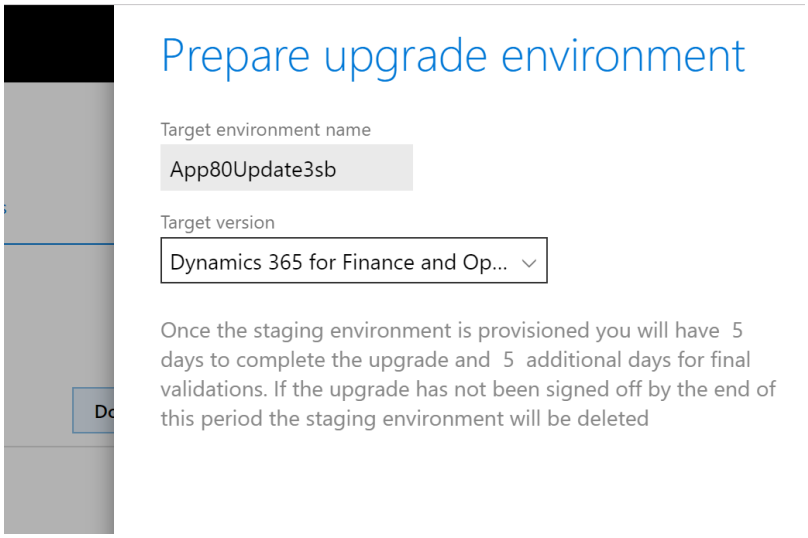
> **IMPORTANT**
>
> Changing Integrated Software Vendor (ISV) solutions, including changing the ISV license code/metadata, during upgrade is strictly not supported. If you are installing a new ISV solution or removing an existing ISV solution, you should do this before or after your upgrade. It cannot be performed during self-service upgrade.

Begin the upgrade

In your sandbox environment, on the **Maintain** menu, select **Upgrade**.



A dialog box appears, where you can select the latest combination of an application version and a platform update.

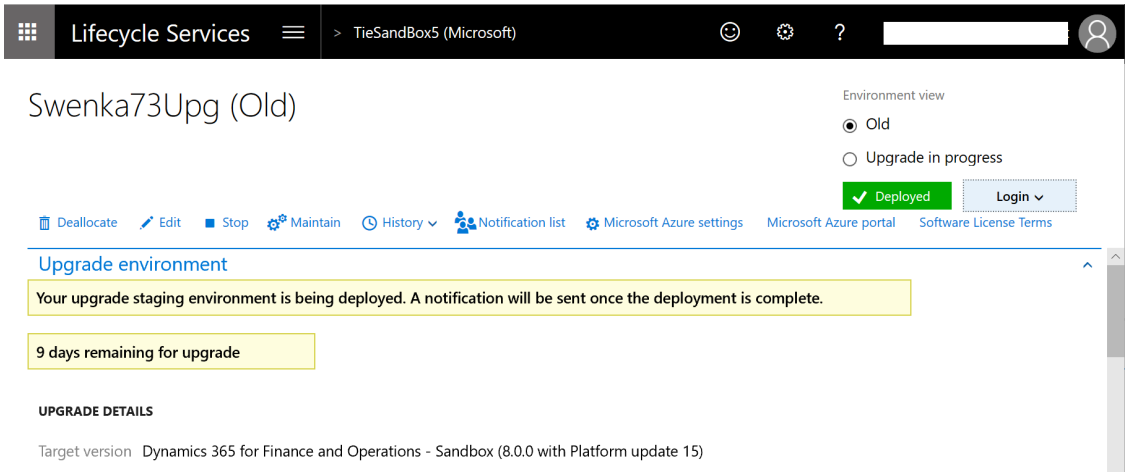


IMPORTANT

If you receive an error that states that preparation failed, see the Known issues section later in this topic.

Preparation

The environment details page is refreshed, and options for two sandbox environments now appear in the upper-right corner. By selecting the options, you can switch between your old sandbox environment and your new upgrade-in-progress sandbox environment.



The preparation stage can take eight hours or longer, because it resembles a full environment deployment. The upgrade-in-progress environment is connected to an empty Azure SQL database to speed up deployment, and it runs on the newer version that you selected to deploy.

During this time, your original sandbox environment is left untouched. There is no downtime impact at this stage.

IMPORTANT

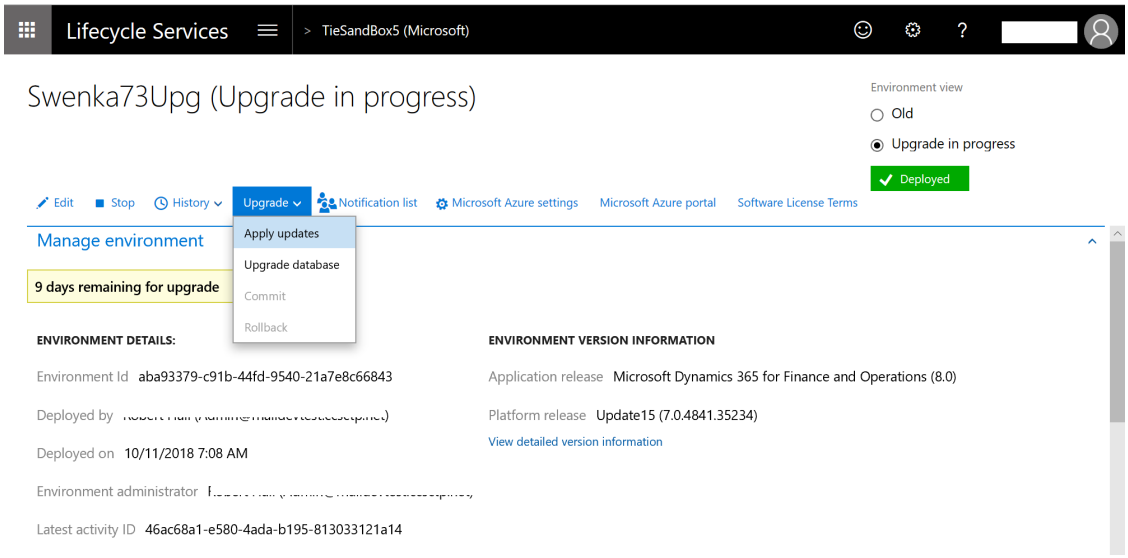
If you receive an error that states that staging deployment failed, the Microsoft Dynamics Service Engineering (DSE) team will be notified and will proactively resolve the issue for you. This issue can occur if Azure doesn't have the required resources available in your region. Microsoft DSE will work with the Azure engineers to allocate more resources. When staging deployment is successfully completed, you will receive an email.

Package application

After staging deployment is completed, go back to the environment details page, and switch to the **Upgrade in progress** view. In this view, you will now see an **Upgrade** menu.

The **Upgrade** menu includes an **Apply updates** option. You can select this option to apply your software deployable packages to the new environment. These packages include any binary packages, whether they are from an independent software vendor (ISV) solution, your own customization packages, or platform binary update packages.

We highly recommend that you apply the latest platform update as your first step. If you're upgrading to version 8.1, we recommend that you get the latest binary update package, such as 8.1.3. This package will also include the latest platform update. In this way, you help guarantee that you have the latest hotfixes that are available and help reduce errors later in the process.

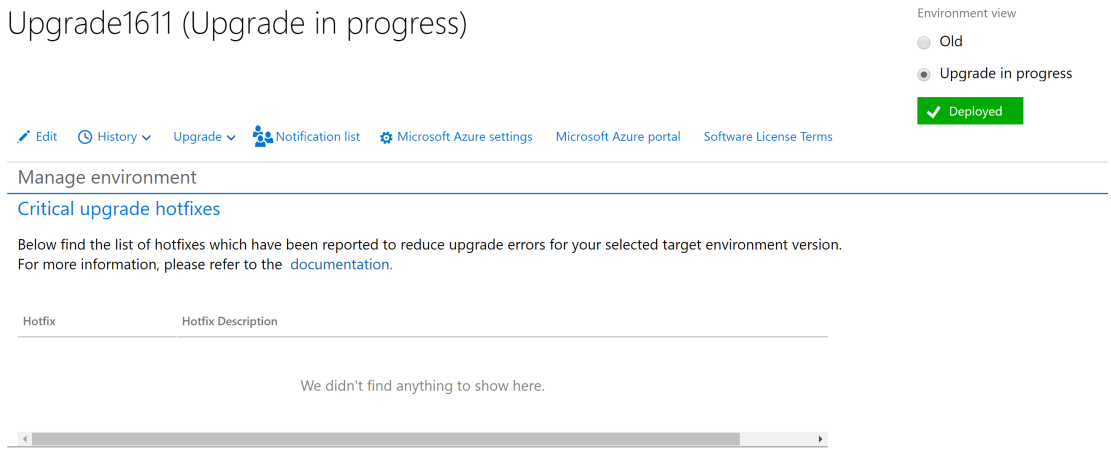


When you apply a new package to the environment, the process is the same as the process for regular environment servicing. When package application is completed, you must use the **Sign Off** button for that package before you can move on or apply another package.

If package deployment fails, you can use the **Rollback** button to reverse it. Note that this button is **not** the same as the **Rollback** option on the **Upgrade** menu.

Critical hotfixes

As use of the self-service upgrade process has increased, Microsoft has found that several hotfixes are critical to success for various target versions. For example, if you're upgrading to version 7.3, a list of Microsoft Knowledge Base (KB) articles that have consistently resolved issues with data upgrade, Retail components, or performance will appear.



The goal is that this list should be empty before you begin the **Data Upgrade** step of the process. The hotfixes in these KB articles must be installed in your upgrade-in-progress environment.

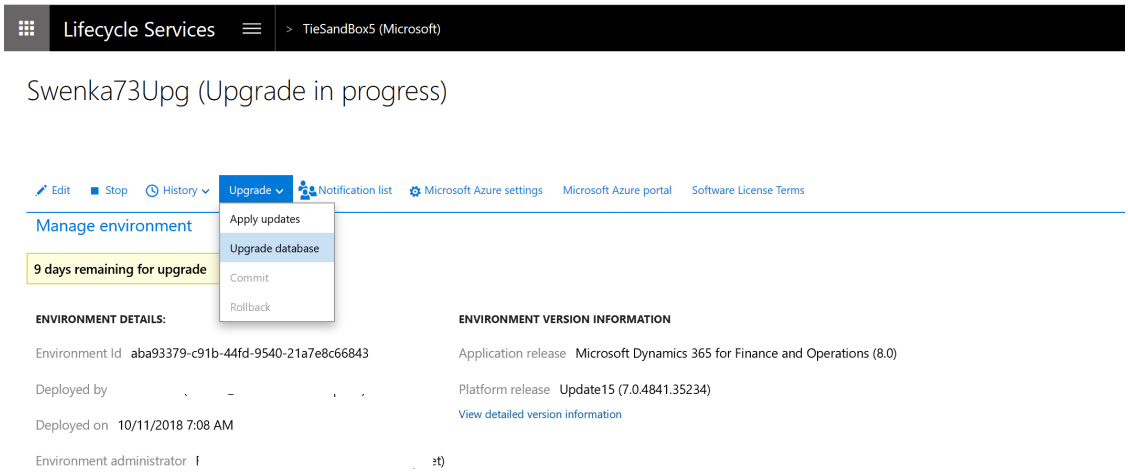
Data upgrade and environment swap

After all packages are applied to your upgrade-in-progress sandbox environment, and you've signed off on

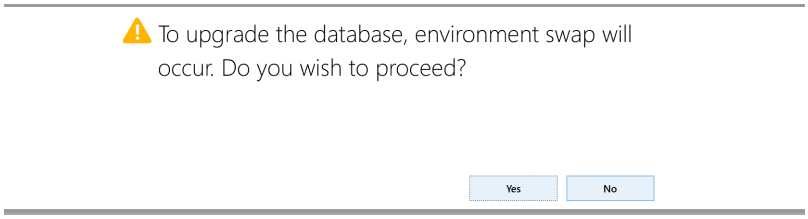
them, you can begin the data upgrade.

> **IMPORTANT**
> This stage begins the downtime for your original sandbox environment.

On the **Upgrade** menu, select **Data upgrade**.



Your original sandbox environment is turned off, and the database connection is swapped so that your new environment is connected to the original database. This process can take up to one hour.



Next, the data upgrade package for your target version is automatically applied. The time that is required to apply the data upgrade package varies, depending on the size of your database.

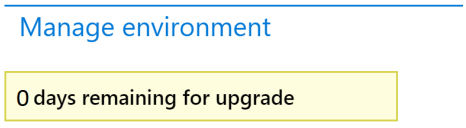
If the data upgrade fails, you must select **Rollback** on the **Upgrade** menu to restore your database to the point that it was at before the data upgrade began. Before you do a rollback, we highly recommend that you download the logs to determine the root cause of the failure. In this way, you can help guarantee that your next data upgrade execution will go more smoothly.

Upgrade days remaining

Because the self-service upgrade process provides a parallel environment at no additional cost to you, there is a time limit on how long this environment can be used. Currently, this time limit is set to 10 calendar days and begins when you select **Upgrade** on the **Maintain** menu to start the process.

What happens when the time limit expires?

There are three possible outcomes when the timer reaches 0 (zero):



- If you haven't yet started the **Data Upgrade** step, the new environment is queued for deletion. In this scenario, the upgrade-in-progress environment was provisioned, and customizations and packages were optionally applied. However, no data was upgraded, and the original environment never incurred downtime.
- If you ran the **Data Upgrade** step but then later performed a rollback, the new environment is queued for deletion. In this scenario, the old environment is the primary environment, because the data upgrade was

rolled back.

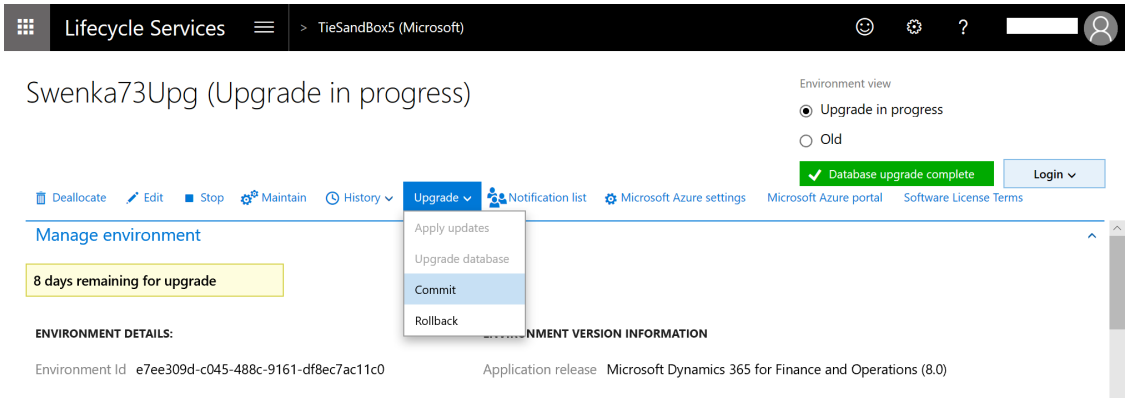
- If you've run the **Data Upgrade** step but haven't yet committed the upgrade, no actions are performed, and no environments are deleted. You can remain in this state until you commit or do a rollback. If you decide to do a rollback, and the timer is at 0 (zero), the new environment will be deleted.

> **IMPORTANT**
>
> **Rollback is only available, at maximum, for 30 calendar days. This is due to the nature of point-in-time restore.** If you try to perform a rollback after 30 days have passed, you will be forced to commit the upgrade, delete the environment, and redeploy on the previous version.

The original environment is queued for deletion only after you commit the upgrade as a success.

Commit or roll back

After the data upgrade package is applied, you can review the environment, and your users can perform business validation activities. If this validation is successful, you can mark the whole upgrade as a success by selecting **Commit** on the **Upgrade** menu. You must commit the upgrade before you can move on to your production environment. After you commit the upgrade, the original environment is queued for deletion.



If the business validation fails, you can select **Rollback** on the **Upgrade** menu. This option will do a point-in-time restore of the database, swap the database connection back to your original sandbox environment, and bring your original sandbox environment back online. The sandbox environment will then be back in its previous state. Be aware, as stated above, that rollback is only possible for up to 30 calendar days.

Post-upgrade actions

After you've signed off on your upgrade, you must update aggregate measurements. Aggregate measurements must be updated after every major upgrade. To update them, go to **System Administration** > **Setup** > **Entity Store**, and then select **Refresh**.

> **NOTE**
>
> You can schedule this update to run by using batch processing.

Upgrade production

After you've committed the upgrade in the sandbox UAT environment, you've finished the upgrade process in the sandbox environment. You can now begin the same process in your production environment. The steps that you follow are the same.

If you encounter an issue that causes excessive downtime during your production upgrade, use the Report production outage process to alert Microsoft and get help.

Upgrade additional environments

You can upgrade additional sandbox environments in the same way. You also can deallocate and delete your

other sandbox environments, and then redeploy on the newer version. By using the Refresh database self-service action, you can copy in the upgraded database from another sandbox or production environment.

Known issues

Prepare operation could not start. Microsoft support has been notified. If the issue persists, please contact support with this ID.

This known issue involves environment certificates on the Microsoft Dynamics Lifecycle Services (LCS) back end. If it affects you, submit a support ticket, and include the activity ID from the error message. Microsoft will work to resolve the issue. Microsoft is compiling a list of affected environments and intends to proactively fix this issue in the future.

I want to cancel the upgrade and try again later.

To cancel an upgrade, you can select **Cancel Upgrade** on the **Maintain** menu. The **Maintain** menu is available in the **Old** view (for the original sandbox environment), not in the **Upgrade in progress** view (for the new sandbox environment).

Upgrade failed at step X: DVT script for service model: MRProcessService.

This DVT error is intermittent and can be resolved by using the **Resume** button for your data upgrade package. When you select **Resume**, the process resumes at the same step. Microsoft is trying to reliably reproduce this issue and intends to produce a fix in the future.

Application configuration sync failed. Call to TTSCOMMIT without first calling TTSBEGIN.

This TTSCOMMIT error is intermittent and can be resolved by using the **Resume** button for your data upgrade package. When you select **Resume**, the process resumes at the same step. (This issue is fixed in PU 21.)

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Update environments from version 8.0 to 10.0.X

2/18/2021 • 7 minutes to read • Edit Online

This topic explains the steps required to update existing Finance and Operations 8.0 environments to 10.0.X application releases.

Background

Traditionally, moving to a newer application version has involved a rigorous upgrade that includes deployment of additional virtual machines, code upgrade, data upgrade, and scheduling several days in advance with the Microsoft Dynamics Service Engineering (DSE) team. You will notice that we are making the uptake of the latest version simpler, and this will continue to improve over time.

> **NOTE**
>
> We are supporting an update experience as compared to a full upgrade. This is possible because there are **no Data Upgrade or Code Upgrade** steps between the 8.0 and 10.0.X application schema. The target environments will be updated just like you would apply a Platform update.

The high-level process to update from version 8.0 to 10.0.X includes the following:

1. Deploy 10.0.X developer and build environments.
2. Branch in version control and remove any application hotfixes.
3. Recompile custom extensions and/or ISV solutions.
4. Produce a single software deployable package.
5. Merge a deployable package with the 10.0.X binary update package.
6. Deploy to target environments for validation.
7. Deploy to Production.

Deploy 10.0.X developer and build environments

Using Lifecycle Services, deploy at least one developer environment and a single, new build environment on application 10.0.X release.

On average this takes 3-4 hours and can be done simultaneously. For the build environment, **Create a new agent pool** and assign it to this environment on the **Advanced options** screen.

In Azure DevOps, visit your existing Build Definition and ensure that it is not using your new agent pool for 10.0.X. This will keep your new build agent from trying to compile older application code.

Apply the latest binary update to your build and development servers

In Lifecycle Services, go to the **build server** that you deployed in step 1. Using the **Update** tiles at the bottom of the **Environment details** page, grab the latest updates available and store it in your project's **Asset Library** using the **Save package** button. For example, this could be saving the 10.0 Platform update 24 package to the **Asset Library**.

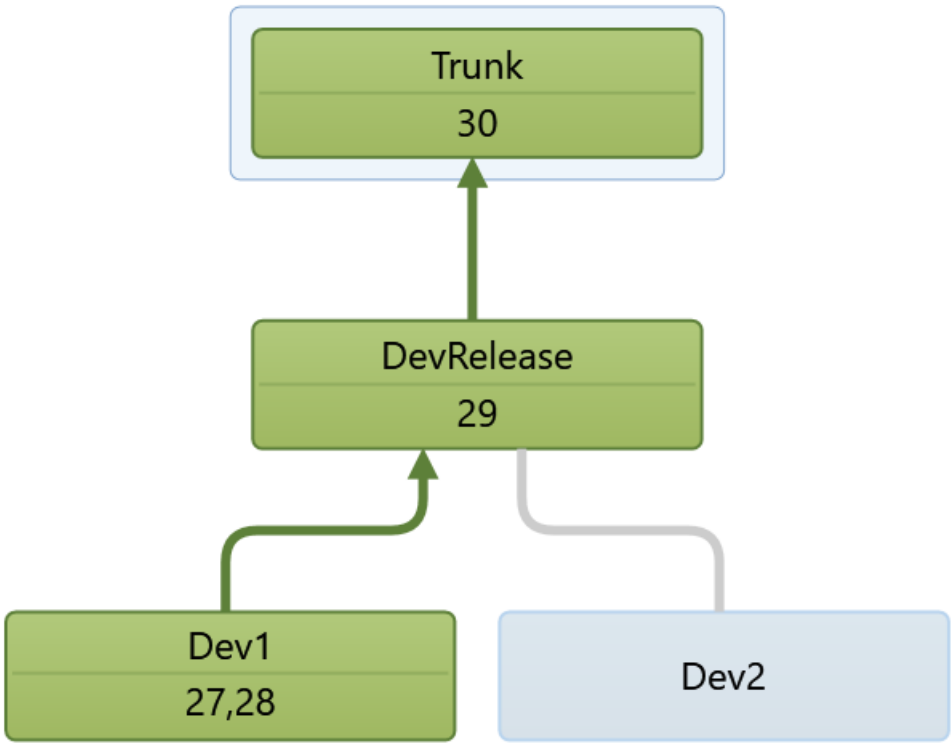
Apply this same package back to the build server where you saved the package, as well as any of the new 10.0.X developer environments you have deployed.

NOTE

We recommend taking the latest updates such as 10.0 Platform update 24 which will be available from the **Update** tiles when you deploy your 10.0 Platform update 24 build server. If your tiles show a count of "0" on the build server, then you can pull the related package from the shared asset library for your version. If your tiles show a count greater than 0, this is the better package to pull and use.

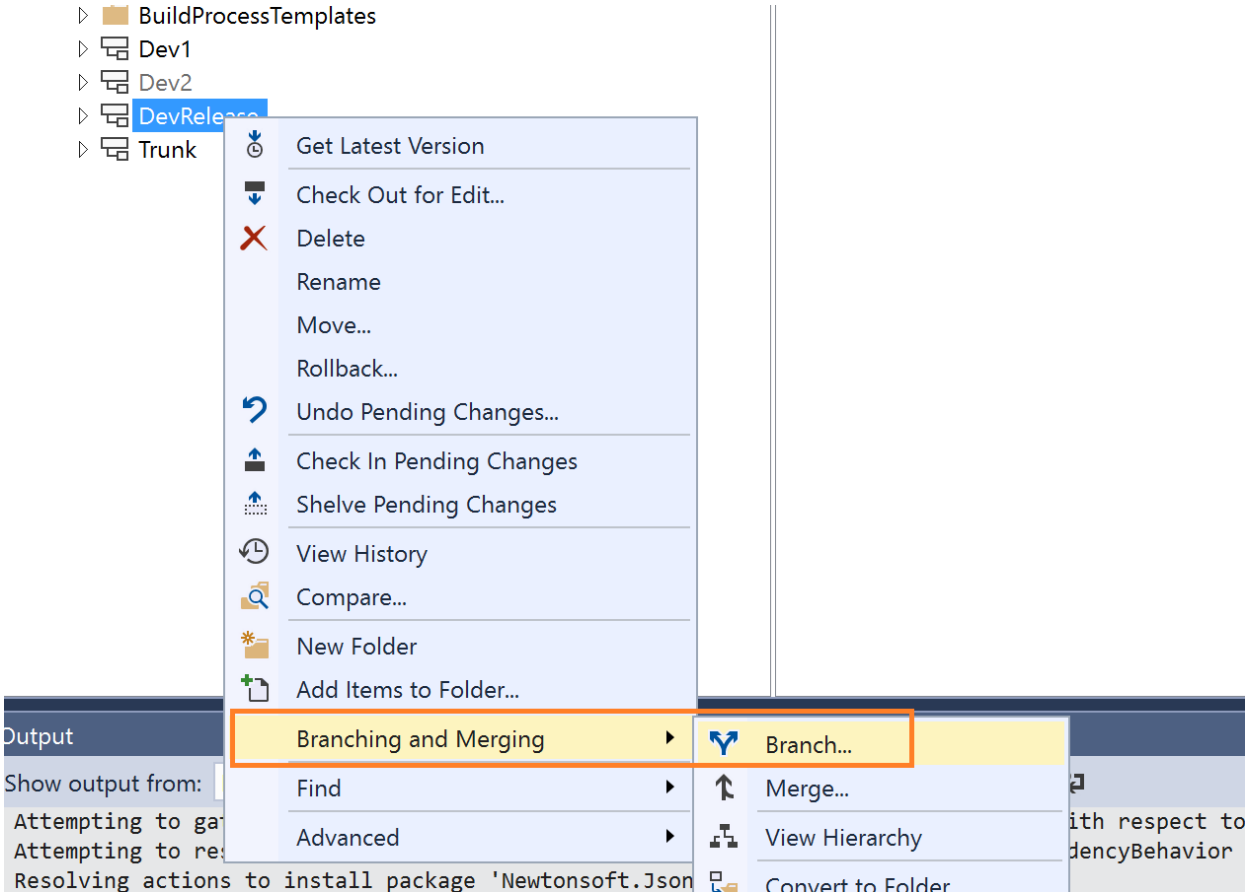
Begin branch work for version control and remove any application hotfixes

While the new environments are deploying, begin the branching work for your update. Use the following branch structure in version control as an example. Branching design varies for each customer, so be careful to adjust your steps accordingly based on how your branches are set up.



Prepare using Visual Studio

On any other development machine (other than the new ones being deployed), open Visual Studio and visit the Source Control Explorer. You will create a new branch that will be isolated for the 10.0.X update.



Next, delete any Microsoft package folders in this branch. You can have packages, such as ApplicationSuite, checked in from applying hotfixes on 8.0 which need to be removed. When only your custom packages or ISVs remain, check these changes in to the branch.

> **IMPORTANT**
> It is critical that this is done before you map version control workspaces on your new development environments. This is to avoid the deletion of the Microsoft hotfixes to cascade to your working environment and delete untouched 10.0.X application code.

Recompile custom extensions and/or ISV solutions

Now you are ready to map this branch to a new development environment and compile your extensions and ISV solutions if they have provided you with source code. If your ISVs have only provided binary packages, you can check them in to source control, and the build environment will merge the binaries with your extension package to produce a single software deployable package. Additional information on this process can be found at Deployable packages from third parties. This will help later when you merge your package with the 10.0.X binary update.

> **NOTE**
> This **step is necessary** as the 10.0.X application code is not backward compatible with 8.0 from a binary level. In future application releases this step will be optional.

Produce a single software deployable package

After you have compiled in a developer environment and there are no errors to resolve, start a build in Azure DevOps using your new 10.0.X build environment agent that was setup earlier. When this is complete, a deployable package artifact will be attached to your build results. Download this package and upload it to the

Lifecycle Services Asset Library. This single package should have all of your extensions and ISV solutions.

Merge the deployable package with the 10.0.X binary update package

In your project's **Asset Library**, locate both your new 10.0.X software deployable package (your customization package that includes your ISVs) and the 10.0.X PU2X binary update package that was saved in Step 1 at the beginning of the topic. Highlight both packages and select **Merge**. This will combine the files into a merged update package. You can now apply this package to your various test environments.

> **NOTE**
> You can't move this merged package between different Lifecycle Services projects. The merge references other packages in your Asset library, and those packages won't be found in a different project.

Deploy to target environments for validation

Using the merged update package, deploy this to your various test environments. For more on how to do this, see Apply updates to cloud environments. This merged update package can be deployed to your Tier1/OneBox environments as well as Tier-2 sandboxes. At a minimum, you must deploy this to the sandbox Tier-2 environment that comes with your subscription. After you have finished with validation, mark the merged update package as a Release Candidate.

Deploy to Production

After you have marked the Release Candidate in your Asset Library, you can schedule the deployment to your Production environment. This will follow the same process for applying other software deployable packages.

Known issues

GlobalUpdate script for service model: AOSService with error 'The specified module 'C:\Program Files\Microsoft Security Client\MpProvider'

This error is transient and can be ignored. To bypass, hit the **Resume** button from Lifecycle Services.

Deploying the 10.0.X binary update to Developer environments causes ApplicationSuite compilation errors

The package can be applied to your 8.0 environments and it will update your source code. Compiling of your extension packages should not be impacted. If you had overlayering and have removed objects from the ApplicationSuite package, and try to recompile it, you may run in to errors. Until this is resolved, please redeploy your developer environments on 10.0.X and sync in your source code from version control.

Cannot find 10.0.X binary update package on the All Binary Updates tile on the My environment details page

It was originally communicated that the package would be found on the **All Binary Updates** tile. To prevent customers who want to simply get the latest binaries for release 8.0 from accidentally updating to release 10.0.X, we have moved the binary package to the Shared Asset Library. This topic has been updated to reflect this change.

Deployment of my environment fails with error on duplicate objects

By default, in Visual Studio when an object is extended, it is created with a name of Object.*Extension1*. This name could clash if Microsoft introduces new extensions of the same object. If this occurs, your deployment will fail with an error similar to the following:

```
Exception calling "CreateRuntimeProvider" with "1" argument(s): "Runtime metadata is invalid because the
same metadata artifact has been defined in multiple assemblies. \nFirst 10 conflicting names:
SystemAdministration.Extension1. \nSee metadata events for complete list."
```

To prevent this from occurring, ensure that you compile your extensions on an 10.0.X developer machine. To resolve this issue, rename any of your extension objects with a vanity extension naming convention, such as SystemAdministration.*Customer*.

Deployment on my environment fails with error on DVTs or ETWs

There is a known issue where IIS/Application Pools are not fully restarted when the DVT or ETW step runs. The failure occurs because the DVTs are trying to connect to your environment's URL. To resolve this issue, click **Resume** on your deployment in LCS to retry the step. We are working to add a timer and automatic retry to resolve this issue.

Software lifecycle policy and cloud releases

2/18/2021 • 9 minutes to read • Edit Online

This topic outlines the lifecycle and support policies for the Finance and Operations online service.

Modern Lifecycle Policy

The Finance and Operations online service is covered by the Modern Lifecycle Policy. The Modern Lifecycle Policy covers products and services that are serviced and supported continuously. For more information about this policy, see Modern Lifecycle Policy. Licensed customers must stay current with updates to the Finance and Operations online service in accordance with the following servicing and system requirements:

- Customers purchasing subscriptions of Finance and Operations and operating on the following application versions will experience continuous updates of the Platform and Financial Reporting. Microsoft will continually update these components with the option to postpone up to 3 consecutive service updates.
    - Dynamics 365 for Operations version 1611 (November 2016)
    - Dynamics 365 for Finance and Operations, Enterprise edition (July 2017)
    - Dynamics 365 for Finance and Operations, Enterprise edition 7.3
    - Dynamics 365 for Finance and Operations, version 8.0 (April 2018)
- Platform versions maintain backward compatibility with the application versions that are supported at the time of the platform release within the application support lifecycle. For more information about platform versions, see Cloud platform monthly updates FAQ.
- Critical fixes and non-critical updates are handled in the following way:
    - **Critical fixes** – Critical fixes include security fixes and any fixes that are required to adhere to the availability service level agreement (SLA) that the service supports. Critical fixes will be made available in the latest platform update version and in the latest service update for customers operating on version 8.1. In addition, to help protect the customer and the online service, Microsoft might apply critical fixes directly to a customer's environment. If a critical fix must be applied, Microsoft will notify the customer about the required downtime window (if there will be any downtime) and apply the fix to the applicable environment. The critical fix will update the system to the latest update version.
    - **Non-critical updates** – Customers operating on the following application releases must update to the most current Finance and Operations platform and financial reporter version to deploy non-critical updates.
        - Dynamics 365 for Operations version 1611 (November 2016)
        - Dynamics 365 for Finance and Operations, Enterprise edition (July 2017)
        - Dynamics 365 for Finance and Operations, Enterprise edition 7.3
        - Dynamics 365 for Finance and Operations, version 8.0 (April 2018)

      Customers operating on release 8.1 must update to the most current service update to deploy non-critical updates.

NOTE

Application and platform releases expire at the end of the month of their software lifecycle.

Microsoft will not provide any fixes to issues on versions that have reached end of service. Microsoft will also not investigate or troubleshoot any issue that you may encounter on an older version. If you encounter an issue on a version that has reached end of service, you will be required to update to the latest update and report the issue if it persists.

All environments will continue to be operated by Microsoft. All automatic processes around your environments, such as monitoring or self-healing, will also continue as is for supported versions.

Dates and versions for application and platform releases

Table 1: Continuous update releases

For information about the new features included in each release, click the links in the **Version** column.

RELEASE	MAJOR RELEASE OR SERVICE UPDATE	VERSION	BUILD NUMBER	AVAILABILITY	END OF SERVICE
Dynamics 365 for Finance and Operations	Major release	10.0	10.0.8	April 2019	Not applicable (continuously updated)*
Dynamics 365 for Finance and Operations	Major release	8.1	8.1.136	October 2018	Not applicable (continuously updated)*

* Indicates a major release is required to be updated through service updates. Service updates are cumulative in nature and may include updates for some or all of the following components: Platform, Application, Financial Reporting, Retail, and operating system updates. You will be required to have an update that's no older than 3 service updates. The 8.1.x version series will be replaced by version 10.0, which is targeted for release in April 2019. For more information, see One Version service updates FAQ.

Table 2: Application releases

For information about the new features included in each release, select the links in the **Version** column.

RELEASE	MAJOR OR MINOR RELEASE	VERSION	BUILD NUMBER	AVAILABILITY	END OF SERVICE
Dynamics 365 for Finance and Operations	Major release	8.0	8.0.30	April 2018	April 30 2019
Dynamics 365 for Finance and Operations, Enterprise edition	Major release	7.3	7.3.11971.56116	December 2017	April 30 2019*
Dynamics 365 for Finance and Operations, Enterprise edition	Major release	July 2017	7.2.11792.56024	June 2017	April 30 2019

RELEASE	MAJOR OR MINOR RELEASE	VERSION	BUILD NUMBER	AVAILABILITY	END OF SERVICE
Dynamics 365 for Operations	Major release	1611	7.1.1541.3036	November 2016	April 30 2019
Dynamics AX	Minor release	7.0.1	7.0.1265.23014	May 2016	June 2017
Dynamics AX	Major release	7.0	7.0.1265.3015	February 2016	June 2017

* All customers must be on the latest version of Finance and Operations by April 2019. However, we are making an exception for customers who have unfulfilled extension requests that have been submitted to Microsoft. Those customers who submitted extensibility requests by January 1, 2019, will be supported on version 7.3 until their extensibility requests are fulfilled. Customers are expected to upgrade to the latest version within 90 days of the extensibility request being fulfilled. For more information, see One Version service updates FAQ.

Table 3: Platform releases

For information about the new features included in each release, select the links in the **Release** column.

RELEASE	BUILD NUMBER	AVAILABILITY	EXPIRATION DATE
Platform update 31	7.0.5457	January 2020	N/A (Continuously updated)
Platform update 30	7.0.5407	November 2019	N/A (Continuously updated)
Platform update 29	7.0.5372	October 2019	N/A (Continuously updated)
Platform update 28	7.0.5314	July 2019	N/A (Continuously updated)
Platform update 27	7.0.5286	June 2019	N/A (Continuously updated)
Platform update 26	7.0.5257	May 2019	N/A (Continuously updated)
Platform update 25	7.0.5222	April 2019	N/A (Continuously updated)
Platform update 24	7.0.5179	March 2019	N/A (Continuously updated)
Platform update 23	7.0.5126	January 2019	N/A (Continuously updated)
Platform update 22	7.0.5095	December 2018	N/A (Continuously updated / Retired)
Platform update 21	7.0.5073	October 2018	N/A (Continuously updated / Retired)
Platform update 20**	7.0.5030	October 2018	N/A (Continuously updated / Retired)
Platform update 15*	7.0.4841	March 2018	N/A (Continuously updated / Retired)
Platform update 12	7.0.4709	November 2017	November 2018
Platform update 11	7.0.4679.35176	October 2017	October 2018

RELEASE	BUILD NUMBER	AVAILABILITY	EXPIRATION DATE
Platform update 10	7.0.4641.16233	August 2017	August 2018
Platform update 9	7.0.4612.35162	July 2017	July 2018
Platform update 8	7.0.4565.16212	June 2017	June 2018
Platform update 7	7.0.4542.16189	May 2017	May 2018
Platform update 6	7.0.4509.16180	April 2017	April 2018
Platform update 5	7.0.4475.16165	March 2017	March 2018
Platform update 4	7.0.4425.16161	February 2017	February 2018
Platform update 3	7.0.4307.16141	November 2016	November 2017
Platform update 2	7.0.4230.16130	August 2016	August 2017
Platform update 1	7.0.4127.16103	May 2016	May 2017
Platform 7.0	7.0.4030.16079	February 2016	January 2017

** Platform updates 16, 17, 18, and 19 have not been made generally available.

* Platform updates 13 and 14 have not been made generally available.

Table 4: Application updates

The application updates listed below consist of a small subset of application enhancements released on top of Finance and Operations versions 8.0, 7.3, and 7.2 (July 2017). These updates do not affect the support lifecycle of the release--support is in-line with the policies for each release.

Note that application updates are not cumulative. The individual packages only contain the enhancements that were included in that specific release. However, if there is a dependency between two packages, then both packages will be included.

For information about the new features included in each update, click the links in the **Version** column.

RELEASE	VERSION	BUILD NUMBER	AVAILABILITY
Dynamics 365 for Finance and Operations	8.1.3: KB 4470000 Microsoft Dynamics 365 for Finance and Operations version 8.1.3 with Platform update 23*	8.1.227	January 2019
Dynamics 365 for Finance and Operations	8.1.2: KB 4470000 Microsoft Dynamics 365 for Finance and Operations version 8.1.2 with Platform update 22*	8.1.195	December 2018

RELEASE	VERSION	BUILD NUMBER	AVAILABILITY
Dynamics 365 for Finance and Operations	8.1.1: KB 4470000 Microsoft Dynamics 365 for Finance and Operations version 8.1.1 with Platform update 21*	8.1.170	October 2018
Dynamics 365 for Finance and Operations	8.0.4: KB 4458992 Microsoft Dynamics 365 for Finance and Operations - Version 8.0.4 (Binary part)*, KB 4458993 Microsoft Dynamics 365 for Finance and Operations - Version 8.0.4 (X++ part)*	8.0.35.15532	August 2018
Dynamics 365 for Finance and Operations	8.0.3: KB 4346176 Microsoft Dynamics 365 for Finance and Operations - Version 8.0.3 (Binary part)*, KB 4346172 Microsoft Dynamics 365 for Finance and Operations - Version 8.0.3 (X++ part)*	8.0.35.15342	July 2018
Dynamics 365 for Finance and Operations	8.0.2: KB 4340414 Microsoft Dynamics 365 for Finance and Operations - Version 8.0.2 (Binary part)*, KB 4340413 Microsoft Dynamics 365 for Finance and Operations - Version 8.0.2 (X++ part)*	8.0.35.15211	July 2018
Dynamics 365 for Finance and Operations	8.0.1: KB 4295107 Microsoft Dynamics 365 for Finance and Operations - Version 8.0.1 (Binary part)*, KB 4294515 Microsoft Dynamics 365 for Finance and Operations - Version 8.0.1 (X++ part)*	8.0.30.15107	June 2018
Dynamics 365 for Finance and Operations, Enterprise edition	7.3.2: KB 4093261 Microsoft Dynamics 365 for Finance and Operations - Version 7.3.2 (Binary part)*, KB 4093262 Microsoft Dynamics 365 for Finance and Operations - Version 7.3.2 (X++ part)*	7.3.11971.62687	March 2018
Dynamics 365 for Finance and Operations, Enterprise edition	7.3.1: KB 4093139 Microsoft Dynamics 365 for Finance and Operations - Version 7.3.1 (Binary part)*, KB 4091727 Microsoft Dynamics 365 for Finance and Operations - Version 7.3.1 (X++ part)*	7.3.11971.62430	March 2018

RELEASE	VERSION	BUILD NUMBER	AVAILABILITY
Dynamics 365 for Finance and Operations, Enterprise edition	Application update 5: KB 4053277 Application Update 5 for Microsoft Dynamics 365 for Finance and Operations (Binary part)*, KB 4053278 Application Update 5 for Microsoft Dynamics 365 for Finance and Operations (X++ part)*	7.2.11792.62725	November 2017
Dynamics 365 for Finance and Operations, Enterprise edition	Application update 4: KB 4047325 Application Update 4 for Dynamics 365 for Finance and Operations (Binary part)*, KB 4047321 Application Update 4 for Dynamics 365 for Finance and Operations (X++ part)*	7.2.11792.62509	October 2017
Dynamics 365 for Finance and Operations, Enterprise edition	Application update 3: KB 4043284 Application Update 3 for Dynamics 365 for Finance and Operations (Binary part)*, KB 4043285 Application Update 3 for Dynamics 365 for Finance and Operations (X++ part)*	7.2.11792.62370	September 2017
Dynamics 365 for Finance and Operations, Enterprise edition	Application update 2: KB 4039142 Application Update 2 for Dynamics 365 for Finance and Operations (Binary part)*, KB 4039487 Application Update 2 for Dynamics 365 for Finance and Operations (X++ part)*	7.2.11792.62192	September 2017
Dynamics 365 for Finance and Operations, Enterprise edition	Application update 1: KB 4035749 Application Update 1 for Dynamics 365 for Finance and Operations (Binary part)*, KB 4035751 Application Update 1 for Dynamics 365 for Finance and Operations (X++ part)*	7.2.11792.62089	July 2017

* The link points to a Knowledge Base (KB) article. You must sign in to Lifecycle Services (LCS) to view the KB article.

Support matrix

Platform updates are compatible with all application versions that are supported at the time of release.

Table 5: Downloadable virtual hard drive (VHD) releases

Use of the VHDs is subject to the Software license terms.

RELEASE	VHD NAME	VHD EXPIRATION DATE
Platform update 12 / Application release 7.2	FinandOps7.2PlatUpdate12.vhd	May 24, 2018
Platform update 12 / Application release 7.3	FinandOps7.3PlatUpdate12.vhd	June 05, 2018
Platform update 15 / Application release 7.3	FinandOps7.3withPlatUpdate15	December 08, 2018

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Apply the latest platform update to environments

2/18/2021 • 7 minutes to read • Edit Online

This topic explains how to apply the latest platform release to your Finance and Operations environment.

Overview

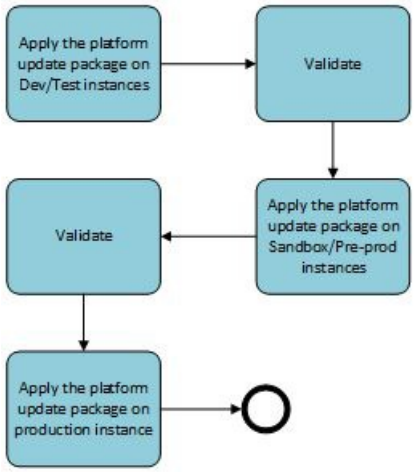
In Finance and Operations, the platform consists of the following components:

- Binaries such as Application Object Server (AOS), the data management framework, the reporting and business intelligence (BI) framework, development tools, and analytics services.
- The following Application Object Tree (AOT) packages:
  - Application Platform
  - Application Foundation
  - Test Essentials

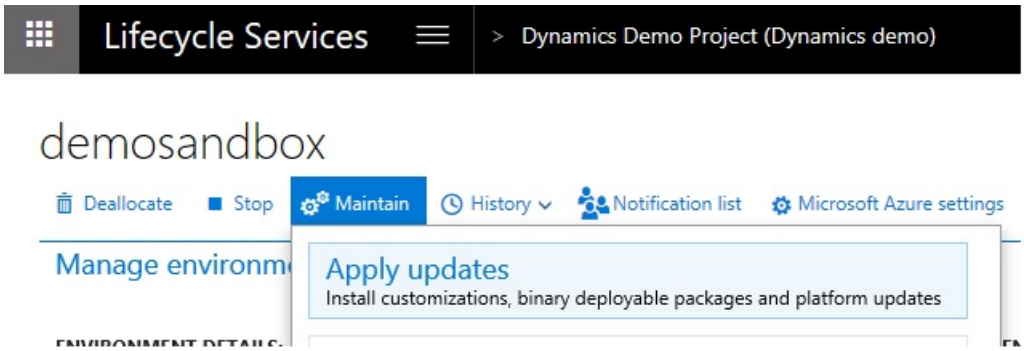
> **IMPORTANT**
>
> To move to the latest platform, your Finance and Operations implementation **cannot** have any customizations (overlayering) of any of the AOT packages that belong to the platform. This restriction was introduced in Platform update 3, so that seamless continuous updates can be made to the platform.

Overall flow

The following illustration shows the overall process for upgrading the platform to the latest update.



If you are already running on Platform update 4 or later, updating to the latest release is a simple servicing operation. After the platform update package is in your LCS asset library, follow the flow to apply an update from the LCS environment page. Select **Apply updates** under **Maintain**, then select the platform update package.



Learn how to **get the latest platform package and apply it to an environment deployed through LCS** in the next section.

Apply the latest platform update package

There are two ways to get the latest platform update package in LCS from your environment page.

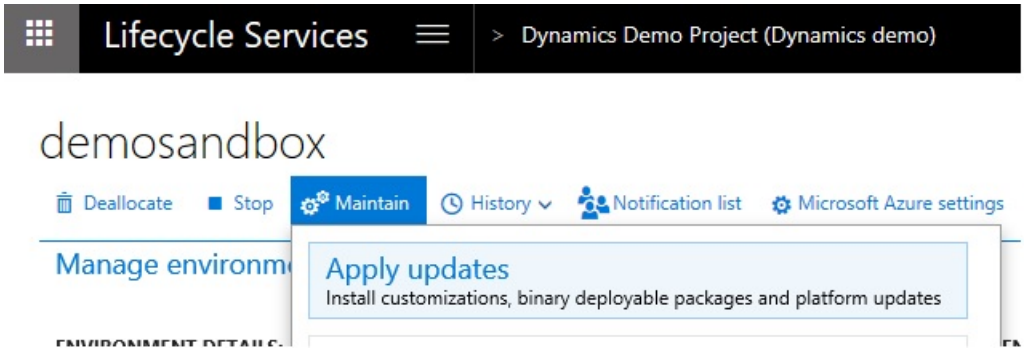
- Click the **Platform binary updates** tile
- Click the **All Binary Updates** tile to see a list of combined package of application and platform binary updates. (As of Platform update 4, binary updates from LCS include an upgrade to the latest platform).

> **NOTE**
> Tiles on an environment's page in LCS show only the updates that are applicable to your environment based on the current version and state of the environment.

Get the latest platform update package by clicking on one of the two tiles as mentioned above. After reviewing the fixes included in the platform, click **Save Package** to save the package to the project asset library.

From a process perspective, deploying a platform upgrade package resembles a binary hotfix deployable package.

- To apply a platform update package to your cloud development, build, demo, tier-2 sandbox, or production environment, update directly from LCS.



For more details, follow the instructions for applying a binary hotfix in Apply updates to cloud environments.

NOTE

Migrate files for Document management: After upgrading to Platform update 6 or later, an administrator needs to click the **Migrate Files** button on the **Document management parameters** page to finish the upgrade process. This will migrate any attachments stored in the database to blob storage. The migration will run as a batch process and could take a long time, depending on the number and size of the files being moved from the database into Azure blob storage. The attachments will continue to be available to users while the migration process is running, so there should be no noticeable effects from the migration. To check if the batch process is still running, look for the **Migrate files stored in the database to blob storage** process on the **Batch jobs** page.

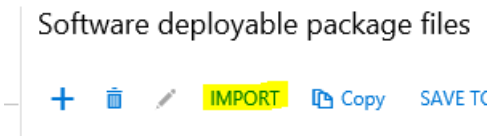
Apply a platform update to environments that are not connected to LCS

This section describes how to apply a platform update package to a *local development environment* (one that that is not connected to LCS).

How to get the platform update package

Platform update packages are released by Microsoft and can be imported from the Shared asset library in Microsoft Dynamics Lifecycle Services (LCS). The package name is prefixed with **Dynamics 365 Unified Operations Platform Update**. Use these steps to import the platform update package:

1. Go to your LCS project's Asset library.
2. On the **Software deployable package** tab, click **Import** to create a reference to the platform update package.



3. Select the desired platform update package.

NOTE

The package in the Shared Asset library may not correspond to the latest build (with hotfixes) of the desired platform release. To guarrantee the latest build, use the LCS environment page as described earlier in this article.

Apply the platform update package to your development environment

NOTE

These instructions apply only to environments that cannot be updated directly from LCS.

Install the deployable package

1. Download the platform update package (AXPlatformUpdate.zip) to your virtual machine (VM).
2. Unzip the contents to a local directory.
3. Depending on the type of environment that you're upgrading, open the PlatformUpdatePackages.Config file under \AOSService\Scripts, and change the **MetaPackage** value.
   - If you're upgrading a development or demo environment that contains source code, change the **MetaPackage** value to **dynamicsax-meta-platform-development**.
   - If you're upgrading a runtime environment, such as a tier-2 sandbox environment or another

environment that doesn't contain source code, the default value, **dynamicsax-meta-platform-runtime**, is correct.

> **NOTE**
> Step 3 is not applicable when upgrading to Platform update 4 or later.

4. Follow the instructions for installing a deployable package. See Install deployable packages from the command line.

5. If you're working in a development environment, rebuild your application's code.

Example

```
AXUpdateInstaller.exe generate -runbookid="OneBoxDev" -topologyfile="DefaultTopologyData.xml" -
servicemodelfile="DefaultServiceModelData.xml" -runbookfile="OneBoxDev-runbook.xml"

    AXUpdateInstaller.exe import -runbookfile=OneBoxDev-runbook.xml

    AXUpdateInstaller.exe execute -runbookid=OneBoxDev
```

Install the Visual Studio development tools (Platform update 3 or earlier)

> **NOTE**
> Skip this section if you are updating to Platform update 4 or later, development tools are automatically installed as part of installing the deployable package.

Update the Visual Studio development tools as described in Update the Visual Studio development tools.

Regenerate form adaptor models

Form adaptor models are required for test automation. Regenerate the platform form adaptor models, based on the newly updated platform models. Use the xppfagen.exe tool to generate the form adaptor models. This tool is located in the package's bin folder (typically, j:\AosService\PackagesLocalDirectory\bin). Here is a list of the platform form adaptor models:

- ApplicationPlatformFormAdaptor
- ApplicationFoundationFormAdaptor
- DirectoryFormAdaptor

The following examples show how to generate the form adaptor models.

```
xppfagen.exe -metadata=j:\AosService\PackagesLocalDirectory -model="ApplicationPlatformFormAdaptor" -
xmllog="c:\temp\log1.xml"

xppfagen.exe -metadata=j:\AosService\PackagesLocalDirectory -model="ApplicationFoundationFormAdaptor" -
xmllog="c:\temp\log2.xml"

xppfagen.exe -metadata=j:\AosService\PackagesLocalDirectory -model="DirectoryFormAdaptor" -
xmllog="c:\temp\log3.xml"
```

Install the Data Management service (Platform update 3 or earlier)

> **NOTE**
>
> Skip this section if you are updating to Platform update 4 or newer, the data management service is automatically installed as part of installing the deployable package.

After the deployable package is installed, follow these instructions to install the new Data Management service. Open a **Command Prompt** window as an administrator, and run the following commands from the .\DIXFService\Scripts folder.

```
msiExec.exe /uninstall {5C74B12A-8583-4B4F-B5F5-8E526507A3E0} /passive /qn /quiet
```

If you're connected to Microsoft SQL Server Integration Services 2016 (13.0), run the following command.

```
msiexec /i "DIXF_Service_x64.msi" ISSQLSERVERVERSION="Bin\2012" SERVICEACCOUNT="NT AUTHORITY\NetworkService"
/qb /lv DIXF_log.txt
```

If you're connected to an earlier release of Microsoft SQL Server Integration Services, run the following command.

```
msiexec /i "DIXF_Service_x64.msi" ISSQLSERVERVERSION="Bin" SERVICEACCOUNT="NT AUTHORITY\NetworkService" /qb
/lv DIXF_log.txt
```

Apply the platform update package on a build environment (Platform update 6 or earlier)

> **NOTE**
>
> Skip this section if you are updating to Platform update 7 or newer. This was a prerequesite step for build environments.

If the build machine has been used for one or more builds, you should restore the metadata packages folder from the metadata backup folder before you upgrade the VM to a newer platform update. You should then delete the metadata backup. These steps help ensure that the platform update will be applied on a clean environment. The next build process will then detect that no metadata backup exists and will automatically create a new one. This new metadata backup will include the updated platform. To determine whether a complete metadata backup exists, look for a BackupComplete.txt file in I:\DynamicsBackup\Packages (or C:\DynamicsBackup\Packages on a downloadable virtual hard disk [VHD]). If this file is present, a metadata backup exists, and the file will contain a timestamp that indicates when it was created. To restore the deployment's metadata packages folder from the metadata backup, open an elevated Windows PowerShell **Command Prompt** window, and run the following command. This command will run the same script that is used in the first step of the build process.

```
if (Test-Path -Path "I:\DynamicsBackup\Packages\BackupComplete.txt") { C:\DynamicsSDK\PrepareForBuild.ps1 }
```

If a complete metadata backup doesn't exist, the command will create a new backup. This command will also stop the Finance and Operations deployment services and Internet Information Services (IIS) before it restores the files from the metadata backup to the deployment's metadata packages folder. You should see output that resembles the following example.

```
6:17:52 PM: Preparing build environment...* <em>6:17:53 PM: Updating Dynamics SDK registry key with
specified values...</em> <em>6:17:53 PM: Updating Dynamics SDK registry key with values from AOS web
config...</em> <em>6:17:53 PM: Stopping Finance and Operations deployment...</em> <em>6:18:06 PM: **A backup
already exists at: I:\\DynamicsBackup\\Packages. No new backup will be created</em><em>.</em> <em>6:18:06
PM: **Restoring metadata packages from backup...</em>** <em>6:22:56 PM: **Metadata packages successfully
restored from backup</em><em>.</em> <em>6:22:57 PM: Preparing build environment complete.</em> <em>6:22:57
PM: Script completed with exit code: 0</em>
```

After the metadata backup has been restored, delete (or rename) the metadata backup folder (DynamicsBackup\Packages), so that it will no longer be found by the build process.

Apply the platform update package

After you've prepared your build environment for this update, apply the platform update package by using the same method that you use on other environments.

Additional resources

Process for moving to the latest update of Finance and Operations

Upgrade data in development or demo environments

2/18/2021 • 17 minutes to read • Edit Online

This topic explains how to upgrade an older database to the latest Finance and Operations application release.

The topic provides instructions for upgrading your Finance and Operations database in a Tier 1 environment to the latest update. A Tier 1 environment is also known as a development, one-box, or demo environment.

In Tier 2 or higher environments, including Production, you will run through the self-service upgrade steps as outlined in Self-service upgrade to the latest version.

IMPORTANT

- You do **not** have to upgrade your database if you're updating to the latest **platform** of Finance and Operations. Platform updates are backward-compatible. This topic applies only to the process of upgrading between releases of Finance and Operations applications, such as an upgrade from Microsoft Dynamics 365 for Operations version 1611 (November 2016) to Finance and Operations 8.0.
- This process doesn't apply to the upgrade of document attachments that are stored in Microsoft Azure blob storage.
- All upgraded custom code has to be applied on the environment before running the data upgrade process.
- If you are on version 8.0 or later, there is no longer a data upgrade between application versions.

Before you begin

1. Back up your current database.
2. You must have a functional environment that is already successfully running the update.
3. In the **source** environment, you must install one of the following hotfixes, depending on the version that you're upgrading from. These hotfixes correct an issue in the SysSetupLog logic, so that the upgrade process can detect the version that you're upgrading from:

   - **If you're upgrading from the November 2016 release (also known as 1611 or 7.1, build 7.1.1541.3036):** KB 4023686, "'Could not find source system version information' error when you upgrade to the latest Application Release."
   - **If you're upgrading from the July 2017 release (also known as 7.2, build 7.2.11792.56024):** No hotfix is required for this version.
   - After you install application hotfixes required in this step, run a full database synchronization. This step is especially important for golden database environments. A full database synchronization fills the SysSetupLog table, which is used when the database is upgraded. Don't run the database synchronization from Microsoft Visual Studio for this step, because the SysSetup interface won't be triggered. To trigger the SysSetup interface, run the following command from an Administrator **Command Prompt** window.

```
cd J:\AosService\WebRoot\bin>

Microsoft.Dynamics.AX.Deployment.Setup.exe -bindir "J:\AosService\PackagesLocalDirectory" -
metadatadir        J:\AosService\PackagesLocalDirectory -sqluser axdeployuser -sqlserver
localhost -sqldatabase axdb -setupmode sync -syncmode fullall -isazuresql false -sqlpwd \
<password for axdeployuser\>
```

4. If you're upgrading from Microsoft Dynamics AX 2012, install the following application X++ hotfixes in the destination environment before you run the data upgrade:
   - KB 4033183 - Dynamics AX 2012 R2 or Dynamics AX 2012 R3 Pre-CU8 non-retail upgrade fails with Object not found for dbo.RETAILTILLLAYOUTZONE.
   - KB 4040692 - Dynamics AX 2012 R3 to Microsoft Dynamics 365 for Operations 7.2 upgrade fails on RetailSalesLine duplicate index on SalesLineIdx.
   - KB 4035490 - Performance issue with GeneralJournalAccountEntry MainAccount field upgrade script.
5. If you're upgrading to Dynamics 365 Finance version 10.0.9 or 10.0.10, install the quality updates in the destination environment before you run the data upgrade.
6. If you're upgrading a database that began as a standard demo data database, you must also run the following script. This step is required because the demo data contains bad records for some kernel X++ classes.

   ```
   delete from classidtable where id >= 0xf000 and id <= 0xffff
   ```

7. Make sure that all Commerce Data Exchange (CDX) jobs have been successfully run, and that there is no unsynchronized transactional data in the cloud version of the channel database.

Select the correct data upgrade deployable package

To obtain the latest data upgrade deployable package for a target environment that is running the latest update, download it from the Microsoft Dynamics Lifecycle Services (LCS) Shared asset library.

1. Sign in to LCS.
2. Select the **Shared asset** library tile.
3. In the Shared asset library, under **Select asset type**, select **Software deployable package**.
4. In the list of deployable package files, find the data upgrade package that corresponds to your upgrade.
   - If you're upgrading from AX 2012, the package name starts with **AX2012DataUpgrade**. Select the package that corresponds to the release you are upgrading to. For example, **AX2012DataUpgrade-10-0**.
   - If you're upgrading from a previous release to the latest 10.0.X release, the package name is **DataUpgrade-10-0**.
   - If you're upgrading from a previous release to a preview release, the package name contains PREVIEW. For example, **DataUpgrade-10-0-2-PREVIEW**.
5. Select the package that corresponds to the release that you are upgrading to.

Upgrade the database

1. Extract the data upgrade deployable package to **C:\Temp** or a location of your choice.

> **NOTE**
>
> Skip this step if this is a development environment that is connected to LCS and you are planning to execute the data upgrade process directly from LCS.

2. Import or restore a backup of the source database (the database that you will be upgrading) to the demo or development environment that is already running the latest update that you want to upgrade to. Leave the existing database in place, and name your new database **imported_new**.

> **NOTE**
>
> If you are validating the data upgrade of your production database running on the earlier release: To copy a database from a production environment back to a demo or development environment, follow the steps in Export a copy of the standard user acceptance testing (UAT) database.
>
> For better upload/download speed between Azure virtual machines (VMs), we recommend that you use AzCopy. For information about how to download AzCopy, and how to use it to copy to or from an Azure blob store, see Transfer data with the AzCopy Command-Line Utility.

3. Rename the original database by adding the suffix **_orig**. Rename the newly restored database so that it has the same name as the original database. In this way, the two databases switch places.

```
ALTER DATABASE <original Dynamics 365 database> MODIFY NAME = <original Dynamics 365 database>_ORIG
ALTER DATABASE imported_new MODIFY NAME = <original Dynamics 365 database>
```

4. Create a backup of the source database, in case you have to revert to it. This step is important because the following steps will modify the source database.

5. Execute the data upgrade package from the **C:\Temp\DataUpgrade** folder (the location that you extracted the deployable package to earlier). Executing a data upgrade package is similar to installing any software deployable package. For detailed instructions, see Install deployable packages from the command line. Start at the section titled **Generate a runbook from the topology** then execute the steps in the section **Install a deployable package**.

> **NOTE**
>
> If you are upgrading a database on a development environment, you can instead execute the data upgrade package directly from the LCS environment page, using the **Maintain > Apply Updates** servicing functionality. This does not require the user to be a local Administrator on the development VM. This is available as of the February release of LCS.
>
> This will upgrade your Finance and Operations database, channel database, and reset the Financial reporting database.

Re-enable SQL change tracking

Run the following SQL against the upgraded database to make sure that change tracking is enabled at the database level. You must specify the name of your database in the **alter database** command.

```
ALTER DATABASE [<your AX database name>] SET CHANGE_TRACKING = ON (CHANGE_RETENTION = 6 DAYS, AUTO_CLEANUP =
ON)
```

Refresh the data entities list

If you have upgraded to Platform update 14 or later, then you will need refresh the data entity list in the Data management workspace (**Data management** > **Framework parameters** > **Entity settings** > **Refresh entity list**) to ensure that the entity list is rebuilt on the latest platform and that the required metadata is available for data management operations.

Troubleshoot upgrade script errors

This section provides information that can help you troubleshoot various issues.

Rerun the runbook after a data upgrade script failure

A data upgrade deployable package enables the runbook to be rerun in a more granular manner than a typical deployable package. The data upgrade scripts begin to be run at Step 5 of the runbook. If you experience a failure during Step 5, view the output in the command window to learn which substep you reached. For example, if you reached substep 5.3, use the following command to rerun from that substep.

```
AXUpdateInstaller.exe execute -runbookid=upgrade -rerunstep=5.3
```

When you're debugging, you don't have to rerun the whole data upgrade piece and database synchronization. You can keep rerunning just the script that fails.

View more details about a script error

Upgrade scripts run in X++ by using a batch process that the runbook installer starts. In Application Explorer in Visual Studio, some classes that you can view are prefixed with **ReleaseUpdate**. If an upgrade script fails during the runbook process, you can learn more about the reason for the error by opening Microsoft SQL Server Management Studio and running the following code to query ReleaseUpdateScriptsErrorLog.

```
select \* from RELEASEUPDATESCRIPTSERRORLOG
```

You can add this code to a new runnable class in Visual Studio, and directly observe, debug, and rework its behavior.

Skip failed scripts

> **IMPORTANT**
> This process is intended to be used only in a development scenario.

You can skip all scripts that have failed a specific number of times, and move to the next viable scripts. This functionality helps with the troubleshooting process. By design, the process is very manual, so that you're less likely to unintentionally skip scripts.

In the ReleaseUpdateConfiguration table, there is a new field that is named **ScriptRetryCount**. The value in this field controls how many times the runbook process will rerun scripts before it ignores them. When the runbook is run, the system updates the **ReleaseUpdateScriptsErrorLog.ErrorCount** field every time that a specific script fails. A new row is created for each script.

In the DataUpgrade package folder, under ..\AosServices\Scripts\, there is a script that is named IgnoreBlockingScripts.ps1. Run this script from an Administrator Windows PowerShell window to skip all scripts where **ScriptRetryCount**=**ErrorCount**. Then rerun the runbook step that failed, so that scripts will be ignored. The **ReleaseUpdateScriptsErrorLog.Ignored** field will also be set for each script that is skipped. Therefore, you can easily identify skipped scripts later.

Monitor the duration of scripts that are run

Every script that is successfully run records the number of minutes that it took in the

ReleaseUpdateScriptsLog.DurationMins column. Therefore, you can easily identify the longest-running scripts when you're trying to tune the performance of the data upgrade process. Note that the duration is the amount of time that each script takes to run. However, because multiple scripts run in parallel, the sum of values in the **DurationMins** column will exceed the overall duration of the upgrade process.

Known issues

A duplicate key was found for the object that is named dbo.RESOURCESETUP

When you upgrade a database, you might receive the following error message during the database synchronization phase of the runbook process:

> Database execution failed: The CREATE UNIQUE INDEX statement terminated because a duplicate key was found for the object name 'dbo.RESOURCESETUP' and the index name 'I_6716AK'

This issue is a known issue that will be resolved in a future hotfix. The workaround is to delete the duplicate rows from the table by running the following SQL script against the database from Management Studio.

```
delete RS from ResourceSetup as RS
join ResResourceIdentifier as RRI on RRI.RecId = RS.Resource_
join WrkCtrTable as WCT on WCT.RecId = RRI.RefRecId
join DirPartyTable as DPT on DPT.DataArea = WCT.DataAreaId
where DPT.DataArea != '' and RS.LegalEntity != DPT.RecId
```

A record can't be selected in Dimension hierarchy nodes (CAMDataDimensionHierarchyNode)

When you upgrade a database, you might receive the following error message during the database synchronization phase of the runbook process:

> Cannot select a record in Dimension hierarchy nodes (CAMDataDimensionHierarchyNode). Dimension hierarchy: 0. The SQL database has issued an error. Object Server DynamicsAXBatchManagement: [Microsoft][ODBC Driver 13 for SQL Server][SQL Server]Invalid column name 'RELATIONTYPE'.

This issue is a known issue that will be resolved in a future release. The workaround is to create a missing field in several tables by running the following SQL script against the database from Management Studio.

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
DROP PROCEDURE IF EXISTS [DBO].PATCHRELATIONTYPE
GO

CREATE PROCEDURE [DBO].PATCHRELATIONTYPE
        @TABLENAME SYSNAME
    AS
    BEGIN
        DECLARE @TABLEID  INT ;
        DECLARE @FIELDID  INT ;
        DECLARE @FIELDNAME  SYSNAME;
        DECLARE @SQLTATEMENT NVARCHAR(1024);
        DECLARE @TOTALRECORDS INT;
        DECLARE @ParmDefinition   NVARCHAR(150);

        SET NOCOUNT ON;

        SELECT @FIELDNAME = 'RELATIONTYPE';
        SELECT @FIELDID = 61453; --Hardcoded in code DBFIELD_DISCRIMINATOR
        IF OBJECT_ID(@TABLENAME, 'U') IS NULL
        BEGIN
```

```
            PRINT @TABLENAME + ' table does not exists. Please provide a base table';
            RETURN;
        END

        IF EXISTS(SELECT 1 FROM SYS.COLUMNS
            WHERE NAME = @FIELDNAME
            AND OBJECT_ID = OBJECT_ID(@TABLENAME))
        BEGIN
            PRINT @TABLENAME + ' table contains RelationType. No patching needed.';
            RETURN;
        END
        PRINT @TABLENAME + ' table does not contain RelationType.';
        SELECT @TABLEID = ID FROM TABLEIDTABLE WHERE NAME = @TABLENAME;
        IF @@ROWCOUNT <> 1
        BEGIN
            PRINT @TABLENAME + ' was not present in TABLEIDTABLE. Cannot proceed!!';
            RETURN;
        END

        -- Ensure that instancerelationtype is added. We don't want to randomly add relationtype to any
table.
        IF NOT EXISTS (SELECT \* FROM SQLDICTIONARY WHERE TABLEID = @TABLEID AND NAME =
'InstanceRelationType')
        BEGIN
            PRINT @TABLENAME + ' table does not exist. Please provide a base table';
            RETURN;
        END

        -- Ensure SQLDictionary is populated
        IF NOT EXISTS (SELECT \* FROM SQLDICTIONARY WHERE TABLEID = @TABLEID AND FIELDID = @FIELDID)
        BEGIN
            INSERT INTO SQLDICTIONARY (TABLEID, FIELDID, ARRAY, NAME, SQLNAME, FIELDTYPE, STRSIZE, SHADOW,
RIGHTJUSTIFY, NULLABLE, FLAGS, RECVERSION)
            VALUES (@TABLEID,@FIELDID,1, @FIELDNAME, @FIELDNAME, 49, 0, 0, 0, 0, 0, 1);
            PRINT 'Inserted into SqlDictionary';
        END

        -- Actual alter statement
        SELECT @SQLTATEMENT = 'ALTER TABLE ' + @TABLENAME +  ' ADD ' + @FIELDNAME + ' BIGINT NOT NULL
DEFAULT 0';
        PRINT 'Issuing ' + @SQLTATEMENT;
        EXEC (@SQLTATEMENT);

        SELECT @SQLTATEMENT = 'UPDATE ' + @TABLENAME +  ' SET RELATIONTYPE = 0';
        PRINT 'Issuing ' + @SQLTATEMENT;
        EXEC (@SQLTATEMENT);
    END;
GO
exec PatchRelationType  'CAMDATADIMENSIONHIERARCHYNODE'
exec PatchRelationType  'CAMDataJournal'
exec PatchRelationType  'CAMDataCostAccountingLedgerSourceEntryProvider'
exec PatchRelationType  'CAMDataImportedDimensionMember'
```

An index can't be created on InventDistinctProduct

When you upgrade a database, you might receive the following error message during the database synchronization phase of the runbook process:

> Cannot create index on InventDistinctProduct a duplicate key exists on column Product.

This issue is a known issue that will be resolved in a future release. The workaround is to delete all records in the InventDistinctProduct table and then resume the runbook from the current step. The records in the InventDistinctProduct table are disposable. They will be regenerated the first time that Finance and Operations is started, when an item is created, or when MRP is run. To delete all records in InventDistinctProduct, run the following query against the current database from Management Studio.

```
truncate table InventDistinctProduct
```

To resume the runbook from the current step, run the following command.

```
axupdateinstaller execute -runbookid=<your runbook name> -rerunstep=<the last step number>
```

For example, you can use this command.

```
axupdateinstaller execute -runbookid=dataupgrade -rerunstep=5.4
```

An exchange rate can't be found when demo data is upgraded

When you upgrade a demo database, you might receive the following error message when you deploy the data upgrade package:

> An exchange rate cannot be found for exchange rate type Default between currencies INR and BRL on exchange date 12/1/2014.

Because you're upgrading demo data, look in the TrvUnreconciledExpenseTransaction table, which is where the expense line is. Change the currency to **USD**. (Because the data is demo data, you don't have to be careful to preserve this expense line.)

```
update TrvUnreconciledExpenseTransaction
set transactioncurrencycode = 'USD'
where transactioncurrencycode = 'INR'
```

Alternatively, go to the original environment that the data came from (such as the old version), and add the missing exchange rate at **General Ledger** > **Currencies** > **Currency exchange rates**. You must add records for Indian rupee (INR) and Brazilian real (BRL) that cover 2014. Then bring that database into your new environment, and start the upgrade against that database.

The interpreter evaluation stack has grown during a call to the kernel

If you've enabled database logging on a kernel table such as UserInfom, you might receive the following error message:

> Executing step: 5.1
> prereq for data upgrade
> prereq for data upgrade
> Unhandled exception More Information: The interpreter evaluation stack has grown during a call to the kernel method xRecord::Delete (), height before call: 0, height after call: 3. Unhandled exception More Information: KernelInstance: Kernel is accessing deleted memory
> The step failed

To resolve this issue, review the database log setup at **System administration** > **Setup** > **Database log setup**. Remove records for kernel tables as you require.

The batch process fails to start

The batch process can fail if the environment was left in maintenance mode after the configuration keys were changed. To resolve this issue, turn maintenance mode off, and then resume the runbook process.

The system fails to locate or generate a user GUID

You might receive the following error message:

> ...Unhandled exception More Information: System failed to locate or generate a user GUID...

To resolve this issue, rerun the runbook step that failed. You will then be able to continue.

Pre-sync or post-sync errors on ReleaseUpdateDB71_LedgerPeriodClose

You might receive one of the following error messages on the **preSyncLedgerPeriodCloseTemplateTask**, **updateMenuItemTypeForCurrencyReval**, or **updateLedgerPeriodCloseTemplateTask** method in the **ReleaseUpdateDB71_LedgerPeriodClose** class:

> Cannot execute the required database operation. The SQL database has issued an error. Object Server DynamicsAXBatchManagement: [Microsoft][SQL Server Native Client 11.0][SQL Server]Invalid column name 'TEMPLATE'. INSERT INTO LedgerPeriodCloseTemplateTaskTmp (TEMPLATE, AREA, NAME, MENUITEM, MENUITEMTYPE, TARGETDAYSFROMPROJECTCOMPLETE, DUETIME, LEGALENTITYSELECTION, RECVERSION, PARTITION, RECID, CLOSINGROLE, LINENUM) SELECT T1.TEMPLATE, T1.AREA, T1.NAME, T1.MENUITEM, T1.MENUITEMTYPE, T1.TARGETDAYSFROMPROJECTCOMPLETE, T1.DUETIME, T1.LEGALENTITYSELECTION, T1.RECVERSION, T1.PARTITION, T1.RECID, T1.CLOSINGROLE, 0 FROM LedgerPeriodCloseTemplateTask T1 session 1013 (Admin) Microsoft.Dynamics.Ax.Xpp.ErrorException: Cannot execute the required database operation. The SQL database has issued an error.
>
> Cannot execute the required database operation. The SQL database has issued an error. Object Server DynamicsAXBatchManagement: [Microsoft][SQL Server Native Client 11.0][SQL Server]Invalid column name 'MENUITEMTYPE'. UPDATE LedgerPeriodCloseTemplateTaskTmp SET MENUITEMTYPE = 0 WHERE MENUITEMTYPE = 2 AND MENUITEM = 'LedgerExchAdj' AND PARTITION = 5637144576 session 1013 (Admin) Microsoft.Dynamics.Ax.Xpp.ErrorException: Cannot execute the required database operation. The SQL database has issued an error.
>
> Cannot execute the required database operation. The SQL database has issued an error. Object Server DynamicsAXBatchManagement: [Microsoft][SQL Server Native Client 11.0][SQL Server]Invalid column name 'TEMPLATE'. INSERT INTO LedgerPeriodCloseTemplateTask (TEMPLATE, AREA, NAME, MENUITEM, MENUITEMTYPE, TARGETDAYSFROMPROJECTCOMPLETE, DUETIME, LEGALENTITYSELECTION, RECVERSION, PARTITION, RECID, CLOSINGROLE, LINENUM) SELECT T1.TEMPLATE, T1.AREA, T1.NAME, T1.MENUITEM, T1.MENUITEMTYPE, T1.TARGETDAYSFROMPROJECTCOMPLETE, T1.DUETIME, T1.LEGALENTITYSELECTION, T1.RECVERSION, T1.PARTITION, T1.RECID, T1.CLOSINGROLE, T1.LINENUM FROM LedgerPeriodCloseTemplateTaskTmp T1 session 1013 (Admin)

To resolve this issue, use Management Studio to manually drop the LedgerPeriodCloseTemplateTaskTmp table from the database. Then rerun the runbook step. This issue will be fixed in a future hotfix.

Table Sync Failed for Table: WarrantyGroupConfigurationItem

If you're upgrading to Dynamics 365 Finance version 10.0.9 or 10.0.10, you might receive the following error message during data upgrade:

> Table Sync Failed for Table: WarrantyGroupConfigurationItem

To resolve the issue, roll back the database upgrade, install the quality updates in the destination environment, and then rerun the data upgrade.

KB number 3170386

If KB number 3170386 isn't installed, you will receive the following error message:

> GlobalUpdate script for service model: AOSService on machine .... Etc .... UpgradeServiceHelper::WaitForDataUpgradeToComplete(Object[]... The step failed

This error is caused by a failure in the pre-sync or the post-sync substep of the data upgrade. Follow these steps to determine which substep failed and the details of the failure.

> **NOTE**
> You can't rerun the failed runbook step until the pre-sync or post-sync substep has been manually completed, and the AutoDataUpgrade.config file has been updated to skip the substeps that have already been run.

1. In File Explorer, in the **DataUpgradeAosServiceScripts** folder, sort by descending order of the date when files were last modified, and then look at the file at the top of the list to determine which substep failed.
   - If the top file is named **dbUpgrade*PreSync*Monitor.error.log**, the pre-sync substep failed.
   - If the top file is named **dbUpgrade*PostSync*Monitor.error.log**, the post-sync substep failed.
2. In Management Studio, run the following **SELECT** statement.

   ```
   SELECT * FROM RELEASEUPDATELOG
   ```

   The second-to-last record in the result set will have the errors and call stacks for all the failures.

DMF errors

If you receive either of the following Data Migration Framework (DMF) errors, download hotfix KB number 3170386, and then restart the upgrade process.

DMF pre-sync error

> Batch error: initial.DAT.ReleaseUpdateDB70_DMF.updateIntegrationActivityExecutionMessageIdPreSync (Batch:AOS-F01B9F0CCC8, 9, Info, Error,):[[1][3,Cannot execute the required database operation. The SQL database has issued an error.][3,Object Server DynamicsAXBatchManagement:][3,[Microsoft][SQL Server Native Client 11.0][SQL Server]Incorrect syntax near 'GO'.][3,

DMF post-sync error

> Batch error: initial.DAT.ReleaseUpdateDB70_DMF.updateIntegrationActivityExecutionMessageIdPostSync (Batch:AOS-F01B9F0CCC8, 9, Info, Error,):[[1][3,Cannot execute the required database operation. The SQL database has issued an error.][3,Object Server DynamicsAXBatchManagement:][3,[Microsoft][SQL Server Native Client 11.0][SQL Server]Incorrect syntax near 'GO'.][3,

Encrypted fields in demo data

After upgrade, values in encrypted fields in the database will be unreadable. However, new values that are entered in these fields after upgrade will be readable. This behavior occurs because of a technical limitation that is related to the certificate that is used for data encryption. The following table shows the fields that are affected.

TABLE.FIELD	DATA EXISTS IN DEMO DATA
CreditCardAccountSetup.SecureMerchantProperties	Yes
ExchangeRateProviderConfigurationDetails.Value	Yes
RetailChannelPaymentConnectorLine.SecureMerchantProperties	Yes

TABLE.FIELD	DATA EXISTS IN DEMO DATA
RetailConnDatabaseProfile.ConnectionString	Yes
RetailHardwareProfile.SecureMerchantProperties	Yes
RetailHardwareProfileMerchantInfoEntity.SecureMerchantProperties	Yes
FiscalEstablishment_BR.ConsumerEFDocCsc	No
FiscalEstablishmentEntity.CSC	No
FiscalEstablishmentStaging.CSC	No
HcmPersonIdentificationNumber.PersonIdentificationNumber	No
HcmWorkerActionHire.PersonIdentificationNumber	No
SysEmailSMPTPassword.Password	No
SysOAuthUserTokens.EncryptedAccessToken	No
SysOAuthUserTokens.EncryptedRefreshToken	No

Additional resources

Process for moving to the latest update of Finance and Operations

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Update the Visual Studio development tools

2/18/2021 • 2 minutes to read • Edit Online

This topic explains how to update the development tools.

Use this tutorial to update your Visual Studio development tools with a new version. It explains how to uninstall your existing Visual Studio development tools and install the new extension. The new extension is in the form of an installable VSIX file. This file is a part of the binary hotfix available on the Dynamics Lifecycle Services (LCS) site. The VSIX file is located in the **DevToolsService\Scripts** folder of the binary hotfix package.

> **NOTE**
> You do not need to follow the instructions in this article if you are upgrading your Finance and Operations platform to Platform update 4 or newer. It is an automatic step that is part of the platform upgrade process.

Uninstall the existing Visual Studio extension

In order to install a new version of the development tools, you'll need to uninstall the existing version first. Verify the version of the development tools that you have installed. If you don't have it installed, you can skip this section.

Verify your current version of the Visual Studio extension

1. Open the Visual Studio **Help > About Microsoft Visual Studio** dialog and find **Finance and Operations Developer Tools**.
2. Select it and click **OK**.

Uninstall the extension

1. Open the Visual Studio **Tools > Extensions and Updates** dialog.
2. Select **Finance and Operations Visual Studio Tools** and click **Uninstall**.
3. When the extension is uninstalled, exit Visual Studio.

Install a new version of the extension

1. Make sure Visual Studio is not running.
2. Double-click (or right-click and **Open**) the VSIX file of the new version.
3. Follow the installation instructions.
4. When installation is complete, you can start Visual Studio and start developing your application.

> **NOTE**
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Plan and prepare for compiling code against the latest update

2/18/2021 • 2 minutes to read • Edit Online

With the rollout of the One Version servicing plan, Microsoft is committed to backward compatibility from a binary and functional perspective. For detailed information about One Version, see One Version service updates FAQ. Even with backward compatibility as a priority, there are situations where development activities may result in required code changes. Some of those situations are described below.

- When Microsoft makes an enumeration extensible, it is considered a binary compatible change. The compiler checks for unsafe extensible enumeration operations that depend on the integer value of a non-extensible enumeration. Any partner code that contains unsafe extensible enumeration operations will have compiler errors when re-compiled and will need to be modified. For more information, see Add values to enums through extension.
- To avoid possible unresolved references when compiling, a partner model should reference the top-level modules and sub-modules. If this is not done, a Microsoft change that adds new resources in an unreferenced sub-module may cause an unresolved reference. To resolve the compilation error, add the sub-module as a reference.
- Some methods will be attributed as obsolete to signal that they will be fully deprecated in the future. Any compiler warning that is generated due to the calling or wrapping of an obsolete method should be investigated to ensure that the expected code path still exists. In some cases, Microsoft code will directly call the new method in place of the obsolete method. When this happens, the code built around the obsolete method will not execute when expected.

Apply updates to on-premises deployments

2/18/2021 • 8 minutes to read • Edit Online

This topic explains how to apply supported updates to Dynamics 365 Finance + Operations (on-premises). All updates to on-premises environments are done through Microsoft Dynamics Lifecycle Services (LCS).

Search for and download updates

For more information about how to find the updates that you can apply to your on-premises environment, see Issue search in Lifecycle Services (LCS). For information about how to download updates from the tiles in the **Updates** section of the **Environment details** page in LCS, see Download updates from Lifecycle Services (LCS).

> **NOTE**
>
> When you are updating an on-premises environment, always select updates from the update tiles on the **Environment** details page. If you select updates from another location, the updates might not work.

Update an on-premises deployment

You can apply updates to an on-premises environment either during deployment or after the deployment is completed.

While an on-premises environment is being deployed, you can select to deploy a custom package in the **Advanced** settings. For more information about how to apply customizations or application X++ updates, see Develop and deploy custom models to on-premises environments.

To apply updates to an on-premises environment after it has been deployed, in LCS, on the **Environment details** page for the environment, under **Maintain**, select **Apply updates**.

> **NOTE**
>
> You can apply updates after deployment only on environments that have Platform update 12 for Finance and Operations or later. The environment must also have the latest version of the local agent available in LCS. For more information, see Update the local agent. If you're on a platform version that is older than Platform update 12, you can reconfigure an environment that is already deployed to update the customizations or update to the latest platform release. For more information about how to redeploy an environment, seeRedeploy on-premises environments.

Apply application or binary updates through LCS

The following steps can be used to apply X++, All Binary, or Platform binary updates.

> **IMPORTANT**
>
> The application of updates requires downtime for your environment. Therefore, no business transactions can be performed in the environment during the update. When you complete the following steps, verify that the system isn't being used, and that an official downtime notice has been communicated to all system users.

> **IMPORTANT**
> To move to the latest platform, always select the platform update from the **Platform Binary Updates** tile on the **Environment** details page. If you select updates from another location, the updates might not work.

Prerequisites

- Before you begin, complete a full backup of the Management Reporter (MR), Microsoft Dynamics AX, and Microsoft SQL Server Reporting Services (SSRS databases). Although the code is restored through LCS, the database must be manually restored to help guarantee that there is no data loss.
- Update your environment to the latest build of Platform update 12.
- Update the local agent to the latest version. For more information, see Update the local agent.
- Depending on the type of update, complete the following steps to generate a deployable package:
  - **Platform binary updates** – Download or save the update directly to the Asset library in LCS by following the steps in Download updates from Lifecycle Services (LCS).
  - **Application binary updates** – Download or save the update directly to the Asset library in LCS by following the steps in Download updates from Lifecycle Services (LCS).
  - **Application X++ updates** – Download the required hotfix to your development environment, and then follow the steps in Create deployable packages of models.
  - **Customizations** – Follow the steps in Develop and deploy custom models to on-premises environments.

Update a sandbox environment

1. In the LCS Asset library, upload the deployable package that was generated in the "Prerequisites" section of this topic to the**Software deployable packages**tab.
2. In LCS, open the on-premises implementation project, and then open the **Environment details** page of the environment to update.
3. Under **Maintain**, select **Apply updates**. A slider shows the updates that were uploaded to the Asset library. Note that only packages that are marked as **Valid** in the Asset library appear.

If you are on local agent version 2.1.0 and higher, complete the following steps.

1. Select the update, and then click **Prepare**. Clicking on **Prepare** will prepare your on-premises environment for servicing.

   > **NOTE**
   > During preparation, the environment state will be **Deployed** but the Deployment status field will show the progress of Preparation. Steps such formatting the package and downloading the package are executed during preparation. The environment is not directly touched during preparation and hence there is no downtime during the preparation phase. Users can continue to use the system during preparation.

2. After the preparation is complete, you will see **Abort** and **Update Environment** buttons. To start applying the update, click **Update Environment**. If preparation fails, see the "Resolve a failed update application" section later in this topic.
3. In the confirmation message, select **Yes**. The servicing operation has started on this environment. This is the start of the downtime on your environment.
4. The environment state is changed from **Deployed** to **Deploying**.
5. After the update is completed, the environment state is changed back to **Deployed**. If application of the

update fails, the environment state is changed to **Failed**. For information about what to do if package application fails, see the "Resolve a failed update application" section later in this topic.

6. Open the **History** and **Environment details** pages to view the operations that were performed on the environment. You can also view a record of major actions that were performed on the environment, such as deployments, servicing, and rollbacks.

If you are on local agent version lower than 2.1.0, complete the following steps.

1. Select the update, and then click **Apply**.
2. In the confirmation message, select **Yes**. The servicing operation has started on this environment. This is the start of the downtime on your environment.
3. Environment state changes from **Deployed** to **Preparing**.

> **NOTE**
> During preparation, steps such formatting the package and downloading the package are executed during preparation. The environment is not directly touched during preparation and hence there is no downtime during the preparation phase. Users can continue to use the system during preparation. However, we recommend that the downtime starts when the environment enters the Preparing state.

4. After preparation is complete, the environment state is changed from **Preparing** to **Deploying**.
5. After the update is completed, the environment state is changed back to **Deployed**. If application of the update fails, the environment state is changed to **Failed**. For information about what to do if package application fails, see the "Resolve a failed update application" section later in this topic.
6. Open the **History** and **Environment details** pages to view the operations that were performed on the environment. You can also view a record of major actions that were performed on the environment, such as deployments, servicing, and rollbacks.

Update a production environment

Before you update a production environment, you must successfully complete the package application update on a sandbox environment.

1. In the project for the sandbox environment that you applied the package to, open the Asset library, and then, on the **Software deployable packages** tab, select the package, and mark it as a **Release candidate**.
2. On the **Environment details** page, under **Maintain**, select **Apply updates**. In the dialog box, only packages that are marked as a **Release candidate** are shown.
3. Select the Release candidate package to be applied to the Production environment.
4. The rest of the Update flow is the same as that of a sandbox environment. Your update experience will differ based on the version of the local agent running on your environment. We recommend that you always run with the latest version.

Resolve a failed update application

When preparation fails, the environment state is **Deployed**. When the application of an update fails, the environment state is **Failed**. The first step is to determine why there is a failure. The location of the logs varies, depending on the stage where the failure occurred:

- **Preparation stage:** If the operation fails during the **Preparation** stage, the logs are uploaded to LCS. In the log files, select **Download logs** to download the log files. If the package has any merge issues, the error is included in the log file.
- **Deploying stage:** If the operation fails during the **Deploying** stage, the logs are located in the on-premises

environment. You must sign in to the environment, and then access the logs and event viewer.

For more information about how to use the troubleshooting logs, see Troubleshoot on-premises deployments.

After you review the logs and determine the cause of the failure, complete one of the following operations to restore the environment to a healthy state. No actions can be performed on an environment that is in a **Failed** state. The environment must first be restored to a healthy state.

- **Retry failed operation** – If update application fails, select **Retry** to recover from the failed operation.
- **Abort failed operation** – Because there is no change made to the on-premises environment, if the preparation fails, you have the option to cancel the operation. Select **Abort** to cancel the preparation.
- **Roll back the update** – To roll back the update that failed, select **Rollback**. Before you start the rollback, you must restore the database to the last known good state. When you select **Rollback**, the environment is restored to the last known good state. The environment state is then changed to **Preparation**, then to **Deploying**, and then to either **Deployed** or **Failed**.

  > **NOTE**
  > The **Rollback** button doesn't roll back the database. You're responsible for restoring the database to the last known backup that was made before update application. This step is critical to help guarantee that there is no data loss.

- **Refresh the state** – If update application fails during the **Preparation** stage, the failure is on the LCS side, and update application hasn't yet started. Therefore, the on-premises environment is in a good state. To restore the LCS environment state to **Deployed**, on the project dashboard page, select **Refresh**.
- **Delete and redeploy an environment** – If the retry and rollback options don't work, you must delete and redeploy the environment. To delete the environment, on the project dashboard page, select **Delete**. You then see the option to configure the environment.

  > **IMPORTANT**
  > This option should **not** be used on a production environment. However, it can be used on a sandbox deployment to restore the environment to a healthy state.

  > Because this option requires that you do a fresh deployment of the environment, you lose any updates that were previously applied. Any customizations and binary updates must be reapplied to the environment.

> **NOTE**
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Redeploy on-premises environments

2/18/2021 • 3 minutes to read • Edit Online

At some point, you might have to redeploy your on-premises environment. This could be to apply a new platform update or because of changes or issues in your implementation. Before you delete the environment you are currently working with, you should save your configuration setting information to use when you redeploy. This topic describes how to save configuration settings and how to redeploy your environment.

Save your configuration

Before you delete the environment you plan to update, use the following steps to save your configuration.

1. In LCS, navigate to **Project Settings** > **On-prem Connectors**.
2. Select the connector to your environment, and then click **Edit**.
3. On the **Edit connector** tab, navigate to **Configure Agent** > **Enter Configuration**.
4. Copy the value of the Download Fileshare location in the **Configuration Settings** section. You will need this later.
5. Log in to the on-premises environment file share machine and copy the **\agent\wp\StandaloneSetup\config.json**. You can use the configuration settings in this json file to redeploy your environment.

Configuration settings

The following tables provide information about configuration settings. Use the **Configuration setting** value from the .json file that you saved in the previous procedure.

Active Directory Federation Services settings

FIELD	CONFIGURATION SETTING
The email address of the user who will be the initial administrator (such as, `adminuser@yourdomain.com`)	components.(AOS).parameters.provisioning.adminPrincipleName.value
ADFS OpenID metadata endpoint for the Dynamics 365 Application group. (such as, https://[federation-service-name]/adfs/.well-known/openid-configuration)	components.(AOS).parameters.activeDirectory.adfsMetadata.value
ADFS OpenID Connect client ID for the AOS application group	components.(AOS).parameters.activeDirectory.adfsClientId.value
ADFS OpenID Connect client ID for the Financial Reporting application group	components.(FinancialReporting).parameters.aad.nativeClientAuthentication.clientId.value

SQL database configuration

FIELD	CONFIGURATION SETTING
SQL SERVER	components.(AOS).parameters.database.dbServer.value
AX DATABASE	components.(AOS).parameters.database.dbName.value

FIELD	CONFIGURATION SETTING
FINANCIAL REPORTING DATABASE	components.(FinancialReporting).parameters.mrdb.dbName.value

File share settings

FIELD	CONFIGURATION SETTING
The file share path for the Microsoft Dynamics 365 instance. This share is used as the document store for files uploaded by users.	components.(AOS).parameters.storage.fileSharePath.value
The File share certificate thumbprint for the Microsoft Dynamics 365 instance.	components.(AOS).parameters.storage.sharedAccessThumbprint.value

> **NOTE**
>
> When you copy the file path configuration value from .json file to LCS UI, make sure to remove the extra backslashes. For example, configuration value **\\DC1\D365FFOStorage** from the .json file should be **\DC1\D365FFOStorage** in the LCS UI.

SSRS configuration settings

FIELD	CONFIGURATION SETTING
The IP Address of the SSRS instance.	components.(AOS).parameters.biReporting.persistentVirtualMachineIPAddressSSRS.value
The thumbprint used by the SSRS application to communicate with AX Service.	components.(ReportingServices).parameters.reportingClientCertificateThumbprint.value

Configure service settings

FIELD	CONFIGURATION SETTING
DYNAMICS 365 DNS INFORMATION - The DNS host name of the Microsoft Dynamics 365 instance, such as ax.d365ffo.onprem.contoso.com.	components.(AOS).parameters.infrastructure.hostName
AOS SERVICE PRINCIPAL USER SETTINGS - The domain user account to run the AX service, such as yourdomain\axserviceuser.	components.(AOS).parameters.infrastructure.principalUserAccountName *
MR SERVICE PRINCIPAL USER SETTINGS - The group managed service account (gMSA) to run the MR application service, such as yourdomain\Svc-FRAS$.	components.(FinancialReporting).parameters.ApplicationServicePrincipalUser.accountName.value *
The group managed service account (gMSA) to run the MR process service, such as yourdomain\Svc-FRPS$.	components.(FinancialReporting).parameters.ProcessServicePrincipalUser.accountName.value *

FIELD	CONFIGURATION SETTING
The group managed service account (gMSA) to run the MR click-once service, such as yourdomain\Svc-FRCO$.	components.(FinancialReporting).parameters.ClickOnceServicePrincipalUser.accountName.value *

> **NOTE**
> Remove the extra backslash from the Principal username cofiguration value in the .json file before entering in the LCS UI. For example, contoso\\AXServiceUser should be entered as contoso\AXServiceUser in LCS.

Application certificate settings

FIELD	CONFIGURATION SETTING
The thumbprint of the Data Encryption certificate.	components.(AOS).parameters.database.dataEncryptionCertificateThumbprint.value
The thumbprint of the Data Signing certificate.	components.(AOS).parameters.database.dataSigningCertificateThumbprint.value
The thumbprint of the Session Authentication certificate.	components.(FinancialReporting).parameters.sessionAuthenticationCertificateThumbprint.value
The thumbprint of the SSL certificate used for WCF/SOAP support.	components.(AOS).parameters.infrastructure.sslCertificateThumbprint.value
The thumbprint used by the Management Reporter to communicate with AX service.	components.(FinancialReporting).parameters.tokenSpec.certThumbprint.value

Redeploy your environment

The following instructions provide information about how to update or redeploy your environment with a new platform or topology.

1. In LCS, navigate to the **Environments** blade in your on-premises project.
2. Click **Delete** to delete your environment.

   > **NOTE**
   > Deleting the environment will not delete the database, infrastructure or Local agent. Only the Service Fabric applications are deleted.

3. Wait for a few minutes and verify that the deployment is deleted. To confirm the deployment is deleted, log in to the on-premises environment and navigate to the Service Fabric Explorer.

   The following applications should be deleted:

   - AXBootstapperAppType

- AXSFType
- FinancialReportingType
- RTGatewayAppType
- ReportingService

  The following on-premises service fabric agent applications should not be deleted:

- LocalAgentType
- MonitoringAgentAppType

4. After all of the applications in step 3 are deleted, go back to LCS and click **Configure**.
5. Select the new topology for your platform.
6. Enter the environment name. You can use the same name or enter a new one.
7. Click **Advanced Settings**. You can now use the relevant configurations from the .json file that you saved to configure your environment.

In-place upgrade process for on-premises environments

2/18/2021 • 12 minutes to read • Edit Online

This topic provides the detailed process for upgrading on-premises environments of Finance and Operations from version 7.x to 10.0.x.

> **NOTE**
> Please perform the upgrade with your sandbox environment before upgrading your production environment.

On-premises upgrade from version 7.x to 10.0.x

> **NOTE**
> Be aware that this upgrade process takes time to complete and Finance and Operations will be inaccessible for the entire duration of the data upgrade.

To upgrade from version 7.x to 10.0.x, there are two possible paths that are currently supported.

An overview of each path is given below:

- **Upgrading from within VHD** - This path involves copying your database into the virtual hard disk (VHD) and executing the upgrade inside it. Overall, this is the simpler method.
- **Upgrading with VHD pointing to your database** - This path involves pointing the VHD upgrade process to your database. The upgrade process is still executed from within the VHD.

> **NOTE**
> The VHD does not need external network access in order to carry out the upgrade process.

Prerequisites

1. In Lifecycle Services (LCS), go to the Shared Assets Library (right side of the screen).
2. Under **Select asset type**, choose **Downloadable VHD**, and download all parts of the VHD package that closely matches the version you will be upgrading to in your on-premises environment. The image requires a high amount of disk space, so be sure to download and extract on a drive with adequate free space.
3. The files that you downloaded are a self-extracting zip file. Extract the VHD to a location with a good amount of free space.
4. Using Hyper-V, launch a virtual machine (VM) and attach the VHD. (Note that the machine must be Generation 1.)
5. Connect to the VM. You can find the credentials in Running the Virtual Machine (VM) locally.
6. Depending on your planned on-premises target version of 10.0.x and the VHD image you downloaded, you may need to download and apply the required Application and Platform Update from the Shared

Asset Library under **Select asset type** and **Software deployable package**. For more information, see Install deployable packages from the command line.

7. If you have any extensions or customizations install them into the VHD now, otherwise the upgrade process will remove any data related to customizations. Check with your independent software vendor (ISV) or value-added reseller (VAR) if you need to prepare your environment before the upgrade.

Upgrading from within VHD

1. Shut-down on-premises AOS, BI, and MR servers or stop the Service Fabric Host Service in each of the nodes and set to disabled.
2. Back up your database from your on-premises environment (typically AXDB). For more information, see Create a Full Database Backup.
3. In the VHD, go to C:\AOSService\PackagesLocalDirectory\Bin\CustomDeployablePackage and copy the MinorVersionDataUpgrade zip file.
4. Paste the file wherever you want and unzip it. For example: c:\D365FFOUpgrade\
5. Open a Command Prompt as Administrator and change the directory to the unzipped folder in step 4.
6. Restore the backup that you created into the OneBox VM. For more information, see Restore a Database Backup Using SSMS.
7. Optional: If the name of your restored database is not AXDB, using PowerShell with administrator privileges, execute:

```
.\Configure-On-Premises-Upgrade.ps1 -DatabaseName '<DB-name>'
```

> **NOTE**
> Substitute with the appropriate value in your case (for example, AXDB). If you would like to edit more values, refer to the appendix of this topic.

   The script will run a database connection test to check that the information you provide is valid.

8. Using the Command Prompt from step 5, execute the following commands:

   a.
   ```
   AxUpdateInstaller.exe generate -runbookid=upgrade -runbookfile=upgrade.xml -
   topologyfile=defaulttopologydata.xml -servicemodelfile=defaultservicemodeldata.xml
   ```

   b. `AxUpdateInstaller.exe import -runbookfile=upgrade.xml`

   c. `AxUpdateInstaller.exe execute -runbookid=upgrade`

   During the execution of cleanup for data upgrade you may encounter an error:

   ```
   Stack trace: Call to TTSCOMMIT without first calling TTSBEGIN.\' on category \'Error\'.
   ```

   To resolve this, re-run the step with this command:

   ```
   AxUpdateInstaller.exe execute -runbookid=upgrade -rerunstep=\<failed-step\>
   ```

9. When the upgrade process has finished successfully, back up the newly upgraded database. If you have customizations from ISVs or VARs, check if you have to run some post data upgrade scripts.
10. Restore the database into your environment with a different name from the production one (for example, AXDBupgraded).

11. Start on-premises AOS, BI, and MR servers, or start the services from the Service Fabric portal.

12. In LCS, open the project, and then, in the **Environments** section, delete the deployment. The applications should start to disappear from Service Fabric Explorer in the environment. This process may take longer depending on the number of nodes that you have. Check the service fabric explorer to verify that all applications have been deleted before deploying a new environment. Note that LCS might indicate that the environment is deleted before the actual process is finished.

13. If you had customizations:

    a. In LCS, go to the Shared Assets Library.

    b. Under **Select asset type**, choose **Model** and download: Dynamics 365 for Finance and Operations on-premises, Version 10.0.x Demo Data. Select the version closest to the 10.0.x environment that you will deploy as the on-premises baseline.

    c. Use this file to create a new database (typically AXDB) using the restore backup option from SQL server. For more information, see Restore a Database Backup Using SSMS.

    d. The database will need to be configured. Follow the steps in Configure the Finance and Operations database.

    e. In LCS, set up a new environment and deploy it with version 10.0.x (Redeploy). For more information, see Set up and deploy on-premises environments (Platform update 12 and later). When you deploy, the database that you specify should be the one created in step 13c (typically AXDB).

    f. Apply your own customizations as well as ISV/VAR modules, to your newly created 10.0.x environment. Otherwise, when the environment initially syncs with the database it will delete any customization or extensions related data.

    g. Shut-down on-premises AOS, BI, and MR servers, or stop the services from the Service Fabric portal.

    h. Rename or delete the demo database (typically AXDB) used in the deploy and then rename your new database (typically AXDBupgraded) to the name that the demo database had (typically AXDB).

    i. Start on-premises AOS, BI, and MR servers, or start the services from the Service Fabric portal.

14. If you didn't have customizations:

    a. (Optional) Rename your old database (typically AXDBold) and then rename your new database (typically AXDB). Make sure that in the next step you input the name of the upgraded DB.

    b. In LCS, set up a new environment and deploy it with version 10.0.x (Redeploy). For more information, see Set up and deploy on-premises environments (Platform update 12 and later).

15. (Optional) If deployment fails because the financial reporting module failed, on the database that you are using for the new environment (typically AXDB), run the following command:

```
ALTER TABLE RETAILTERMINALTABLE ADD CONSTRAINT PK_RecId PRIMARY KEY
CLUSTERED (RECID)
```

Upgrading with VHD pointing to your database

1. Shut-down on-premises AOS, BI, and MR servers, or stop the services from the Service Fabric portal.

2. Back up your database from your on-premises environment (typically AXDB). For more information, see Create a Full Database Backup (SQL Server).

3. Restore the backup that you just created into the database server and give it a different name (AXDBtoupgrade). For more information, see Restore a Database Backup Using SSMS.

4. Once connected, go to C:\AOSService\PackagesLocalDirectory\Bin\CustomDeployablePackage and copy the MinorVersionDataUpgrade zip file.
5. Paste the file wherever you want and unzip it. For example: C:\D365FFOUpgrade\
6. Open a Command Prompt as Administrator and change the directory to the unzipped folder from step 5.
7. Open a new PowerShell as Administrator and execute:

```
.\Configure-On-Premises-Upgrade.ps1 -DatabaseName '<DB-name>' -DatabaseServer '<SqlServerName>' -
DatabaseUser '<User>' -DatabasePassword '<Password>'
```

> **NOTE**
>
> Substitute <*> with the values you require.

> **NOTE**
>
> - Only SQL Server authentication is officially supported for this upgrade. For more information, see Create a Database User.
> - You will need to add the Certificate Authority certificate that signed your SQL Server certificate to the OneBox trusted certificate authorities. For more information, see Installing the trusted root certificate.
> - Make sure the database user you use has the sysadmin server role assigned or at least All Privileges on the database you want to upgrade and has permissions to access tempDB. Step 6 of the upgrade process will fail if this is not true.
> - When you install the Certificate Authority in the OneBox, make sure you use the FQDN or IP for connecting to the database that appears there. If you can't access it by using the domain name because it doesn't point to that server, edit your hosts file and add the DN and the IP it should resolve to.

8. Using the Command Prompt from step 6, execute the following commands:

   a.
   ```
   AxUpdateInstaller.exe generate -runbookid=upgrade -runbookfile=upgrade.xml -
   topologyfile=defaulttopologydata.xml -servicemodelfile=defaultservicemodeldata.xml
   ```

   b. `AxUpdateInstaller.exe import -runbookfile=upgrade.xml`

   c. `AxUpdateInstaller.exe execute -runbookid=upgrade`

   During the execution of Cleanup for data upgrade you may encounter an error:

   ```
   Stack trace: Call to TTSCOMMIT without first calling TTSBEGIN.\' on category \'Error\'.
   ```

   To resolve this, re-run the step with this command:

   ```
   AxUpdateInstaller.exe execute -runbookid=upgrade -rerunstep=\<failed-step\>
   ```

9. If you have customizations from ISVs or VARs, verify if you have to run some post data upgrade scripts.
10. Start on-premises AOS, BI, and MR servers, or start the services from the Service Fabric portal.
11. In LCS, open the project, and then, in the **Environments** section, delete the deployment. The applications should start to disappear from Service Fabric Explorer in the environment. This process may take longer depending on the number of nodes you have.
12. If you had customizations:

    a. In LCS, go to the Shared Assets Library.

b. Under **Select asset type**, choose **Model** and download: Dynamics 365 for Finance and Operations on-premises, Version 10.0.x Demo Data. Select the version closest to the 10.0.x environment that you will deploy as the on-premises baseline.

c. Use this file to create a new database (typically AXDB) using the restore backup option from SQL server. For more information, see Restore a Database Backup Using SSMS.

d. The database will need to be configured. Follow the steps under Configure the Finance + Operations database.

e. In LCS, set up a new environment and deploy it with version 10.0.x (Redeploy). For more information, see Set up and deploy on-premises environments (Platform update 12 and later). When you deploy, the database that you should specify should be the one created in step 12c (typically AXDB).

f. Apply your own customizations as well as ISV/VAR modules, to your newly created 10.0.x environment. Otherwise when the environment initially syncs with the database it will delete any customization or extensions related data.

g. Shut-down on-premises AOS, BI, and MR servers, or stop the services from the Service Fabric portal.

h. Rename or delete the demo database (typically AXDB) used in the deploy and then rename your new database (typically AXDBupgraded) to the name the demo database had (typically AXDB).

i. Start on-premises AOS, BI, and MR servers, or start the services from the Service Fabric portal.

13. If you didn't have customizations:

    a. (Optional) Rename your old database (typically AXDBold) and then rename your new database (typically AXDB). Make sure that in the next step you input the name of the upgraded database.

    b. Set up a new environment and deploy it with version 10.0.x. For more information, see Set up and deploy on-premises environments (Platform update 12 and later).

14. (Optional) If deployment fails because the financial reporting module failed, on the database that you are using for the new environment (typically AXDB), run the following command:

```
ALTER TABLE RETAILTERMINALTABLE ADD CONSTRAINT PK_RecId PRIMARY KEY
CLUSTERED (RECID)
```

Appendix

Configure-On-Premises-Upgrade.ps1 usage

> **IMPORTANT**
>
> This script is only meant to be run from a OneBox VHD environment.
>
> The script requires that you pass at least the DatabaseName parameter. If you don't pass it, the script will automatically request it.

If you want to pass an additional parameter like DatabaseServer, or DatabaseUser you can do so but this will cause the script to ask you for all additional parameters. This happens because the script will assume that you want to point the database connection to a machine outside the VM and those parameters will be required to correctly establish the connection.

The parameters that can be passed to the script are:

- **-DatabaseName** - Database name that you want to upgrade.

- **-DatabaseServer** - Database server containing Finance and Operations (on-premises) database.
- **-DatabaseUser** - Username for SQL Authentication.
- **-DatabasePassword** - Password for SQL Authentication.

After configuration has been passed, the script will execute a database connection test with the new parameters. If the script is unable to connect we recommend that you use Microsoft SQL Server Management Studio or some other tool to debug the connection from there.

Configure-On-Premises-Upgrade.ps1

```
<#
.Synopsis
   Configures a Onebox deployment to upgrade an OnPrem 7.x database to OnPrem 10.0.x

.DESCRIPTION
   This must be executed before the upgrade process is carried out.

.EXAMPLE
   .\Configure-OnPremUpgrade.ps1 -DatabaseName 'AxDB'

   .\Configure-OnPremUpgrade.ps1 -DatabaseName 'AxDB' -DatabaseServer '127.0.0.1' -DatabaseUser 'axdbadmin'
-DatabasePassword 'secretPass'
#>
[CmdletBinding()]
param
(
    # Database server containing Microsoft Dynamics 365 for Operations, on-premises database.
    [AllowNull()]
    [string] $DatabaseServer,

    # Database name that you want to upgrade.
    [Parameter(Mandatory = $true)]
    [string] $DatabaseName,

    # Username for SQL Authentication.
    [AllowNull()]
    [string] $DatabaseUser,

    # Password for SQL Authentication.
    [AllowNull()]
    [string] $DatabasePassword
)

$webroot = "C:\AOSService\webroot"

$commandParameter = " -decrypt `"$webroot\web.config`""

$command = Resolve-Path "$webroot\bin\Microsoft.Dynamics.AX.Framework.ConfigEncryptor.exe"

Start-Process $command $commandParameter -PassThru -Wait

if([string]::IsNullOrEmpty($DatabaseUser) -and [string]::IsNullOrEmpty($DatabasePassword) -and
[string]::IsNullOrEmpty($DatabaseServer)) {

    [xml]$web = Get-Content $webroot\web.config

    $web.SelectSingleNode("configuration/appSettings/add[@key='DataAccess.Database']").value =
[string]$DatabaseName

}
else {

    if([string]::IsNullOrEmpty($DatabaseServer)){

        $DatabaseServer = if($value = Read-Host 'What is the IP or FQDN of the Database server?
```

```
[127.0.0.1]') {$value} else {'127.0.0.1'}

    }

    if([string]::IsNullOrEmpty($DatabaseUser)){

        $DatabaseUser = if($value = Read-Host 'What is the SQL Authentication username? [axdbadmin]')
{$value} else {'axdbadmin'}

    }

    if([string]::IsNullOrEmpty($DatabasePassword)){

        $dbPassEn = if($value = Read-Host 'What is the SQL Authentication password?' -AsSecureString)
{$value} else {''}

        $BSTR = [System.Runtime.InteropServices.Marshal]::SecureStringToBSTR($dbPassEn)

        $DatabasePassword = [System.Runtime.InteropServices.Marshal]::PtrToStringAuto($BSTR)

    }

    [xml]$web = Get-Content $webroot\web.config

    $web.SelectSingleNode("configuration/appSettings/add[@key='DataAccess.DbServer']").value =
[string]$DatabaseServer

    $web.SelectSingleNode("configuration/appSettings/add[@key='DataAccess.Database']").value =
[string]$DatabaseName

    $web.SelectSingleNode("configuration/appSettings/add[@key='DataAccess.SqlUser']").value =
[string]$DatabaseUser

    $web.SelectSingleNode("configuration/appSettings/add[@key='DataAccess.SqlPwd']").value =
[string]$DatabasePassword

}
#Save Configuration to webroot config
$web.Save("$webroot\web.config")

#Reloading the configuration to run test
[xml]$web = Get-Content $webroot\web.config

$TestDbServer = $web.SelectSingleNode("configuration/appSettings/add[@key='DataAccess.DbServer']").value

$TestDbName = $web.SelectSingleNode("configuration/appSettings/add[@key='DataAccess.Database']").value

$TestDbUser = $web.SelectSingleNode("configuration/appSettings/add[@key='DataAccess.SqlUser']").value

$TestDbPass = $web.SelectSingleNode("configuration/appSettings/add[@key='DataAccess.SqlPwd']").value

#Setting up connection test.

$dbConn = New-Object System.Data.SqlClient.SqlConnection

$dbConn.ConnectionString = "Data Source=$TestDbServer;User
ID=$TestDbUser;Password=`"$TestDbPass`";Database=$TestDbName"

try{

    $dbConn.Open()

    $result = $true

}

catch{

    $result = $_.Exception.Message
```

```
    $result = $_.Exception.Message

}
Finally{

    $dbConn.Close()

}

$commandParameter = " -encrypt `"$webroot\web.config`""

Start-Process $command $commandParameter -PassThru -Wait

if($result -ne $true){

    Write-Host "`nThe connection to the Database Server failed:" -ForegroundColor Red
    Write-Host $result -ForegroundColor Red

}
else{

    Write-Host "`nThe connection to the Database Server was successful!" -ForegroundColor Green

}
```

Troubleshooting

- Wrong database name or user doesn't have access to that database: Exception calling "Open" with "0" argument(s): "Cannot open database "AxDB1" requested by the login. The login failed. Login failed for user 'axdbadmin'."
- Could not establish a connection. Check ip/fqdn and ports: Exception calling "Open" with "0" argument(s): "A network-related or instance-specific error occurred while establishing a connection to SQL Server. The server was not found or was not accessible. Verify that the instance name is correct and that SQL Server is configured to allow remote connections. (provider: Named Pipes Provider, error: 40 - Could not open a connection to SQL Server)"
- Login credentials are not correct. Exception calling "Open" with "0" argument(s): "Login failed for user 'axdbadmin'."

Download updates from Lifecycle Services (LCS)

2/18/2021 • 6 minutes to read • Edit Online

This topic covers what updates you should expect to see and how you can get the latest updates using Lifecycle Services (LCS).

Get updates

To view available updates:

1. Sign in to LCS using your credentials.
2. In the LCS project, select an environment.
3. On the **Environment** page, scroll down to see the **Available updates**.

Types of updates

- **Binary updates** are pre-compiled and cumulative. Every subsequent binary update includes all previous updates. These updates don't have to be compiled in a development environment, and they can be applied directly to a non-development environment from LCS.

  If you're running an environment that has Commerce functionality and a customized instance of Cloud point of sale (POS), you must complete the additional steps that are listed under the SDK packaging. For Microsoft Dynamics 365 Commerce, all updates, even updates for application models, are released as binary updates.

  For all versions of Commerce and Finance and Operations apps that are version 8.1 and later, all updates, including updates for application models, are released as binary updates.

- **X++ updates** include updates to specific application functionality in application models. These updates can be independently downloaded and applied. You can select specific X++ updates to apply to your environment. Dependent X++ updates are automatically selected and downloaded. X++ updates are source code updates. Before they can be applied to a non-development environment, X++ updates must be compiled in a developer environment and merged with any customizations. X++ updates apply only to version 8.0 and earlier.

Update option by product and version

Based on your product and version, you will have different update options from Lifecycle Services.

Finance and Operations apps

- **Application version 8.1 and later (One Version)** - All updates for version 8.1 and later will have the One Version service update experience. It will be a cumulative, combined binary update of all of the application and platform updates. There will be no granular X++ updates starting with this release.

  Based on your environment version and the service update availability, you will have the option to choose the updates available to your environment. Each update option is associated with a version number and a build number.

  You may see one or more of the following update options.

UPDATE	DESCRIPTION	AVAILABILITY
Quality update	A quality update is a cumulative, roll-up build that contains fixes for known issues that are specific to the service update.	A quality update is available when your environment is running the same version of the current service update (n), or when your environment is running on one version older than the current service update (n-1). For example, if the current service update is version 10.0.2, you will have the option to choose a quality update if you're running version 10.0.2, or if you're running one version older, which is 10.0.1. There will be no quality update available for any version that's older than 2 versions of the current service update. You will have to apply the latest service update to stay current.
Service update	A service update is the version currently automatically applied to customer environments based on the LCS project update settings. A service update is a cumulative, roll-up build that contains new features, functionality, and the related quality update that is generally available.	A service update is available if your environment has not been updated to the current service update version available for auto-update. Only the designated sandbox or production environment will be auto-updated if you have configured the update settings for the LCS project. However, you can manually apply the current service update version to other sandbox environments or your cloud-hosted environments.
Upcoming service update	An upcoming service update is the latest version that is generally available for self-update. An upcoming service update is a cumulative, roll-up build that contains new features, functionality, and the related quality update that is generally available.	An upcoming service update will be made generally available for self-deployment approximately 2 weeks prior to when Microsoft starts automatically applying this version based on your update settings for the LCS project.

- **Application version 7.3 with Platform update 4 and later** - This release will still have the granular X++ updates. Starting with Platform update 4, no overlayering is allowed on the platform modules, which means that the **Platform binary updates** tile is available to provide the platform updates as a cumulative update.

  For customers that are on this combination, you will see the following tiles:

  - **All X++ updates** - This tile shows all the granular X++ updates released by Microsoft.
  - **Critical X++ updates** - This tile shows recommended KBs that are based on the telemetry data in your production environment. This tile will only show Production environments and a subset of the updates shown under the **All X++ updates** tile that are recommended for your environments.

- **All binary updates** - This tile shows a combined, cumulative binary update for both the Application and Platform.
- **Platform binary updates** - This tile shows only the Platform binary updates. If you want to update only the platform, you can get the update from this tile.

- **Application version 7.1, 7.2, 8.0, or earlier (except version 7.3) with Platform update 32 and earlier** - The product versions noted here are out of service. No new X++ updates are available. You can apply the X++ updates that have been released previously but no new X++ update will be published to LCS.

  Also, a platform update will not be available starting with **Platform update 33**. This means that you will not be able to apply the platform only update package if your application version is 7.1, 7.2, or 8.0 and earlier (except version 7.3). If you're running any of these versions, you need to upgrade to the latest version to stay with the latest feature and functionality. For more information, see One Version service updates FAQ.

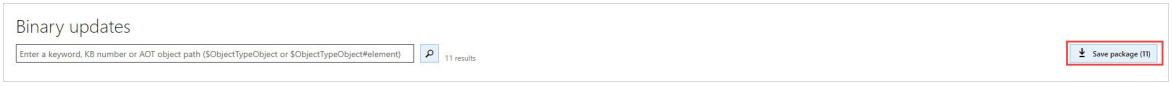
> **NOTE**
>
> If you are on a release that is noted above, you need to upgrade as soon as possible.
>
> For the X++ updates that have been released for these versions, they are available from Issue Search in Lifecycle Services.

Download binary updates

To download binary updates, follow these steps in LCS.

1. Select any of the binary update options, including Quality update, Service update, All binary updates, and Platform binary updates to view the combined list of application and platform binary updates.
2. On the **Binary updates** page, select **Save package**.

   > **NOTE**
   >
   > You will not be able to select Knowledge Base (KB) articles to be saved because binary updates will automatically save all KBs in an update package.

3. On the **Review and save updates** page, select **Save package**.
4. In the **Save package to asset library**, enter the **Name** and **Description**, and select **Save package**.
5. Select **Done** to return to the environment page.
6. You'll see the saved binary package in the asset library.

Download X++ updates

To download X++ updates, follow these steps in LCS.

1. Select the **All X++ updates** tile to view the list of available application updates for an environment, or select the **Critical X++ updates** tile for the application updates that are recommended for your production environment.
2. On the **Add updates** page, select the applicable Knowledge Base (KB) numbers, and then select **Add** to

add selected KBs to the download package.

> **NOTE**
>
> For X++ updates, you can download all available updates at this point. Click **Select all**, and then select **Add** to add all KBs to the download package.

3. Select **Download package**.
4. On the **Review and download hotfixes** page, you can review the hotfixes that you selected, discard the package, return to the hotfix selections, or download the final hotfix package.
5. Download the package, and select **Done**.

Additional resources

- Apply updates to cloud environments
- Install metadata hotfixes in development environments

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Apply updates to cloud environments

2/18/2021 • 9 minutes to read • Edit Online

This topic describes how you can use Microsoft Dynamics Lifecycle Services (LCS) to automatically apply updates to cloud environments.

> **IMPORTANT**
> Updates are applied using deployable packages. Applying updates causes system downtime. All relevant services will be stopped, and you won't be able to use your environments while the package is being applied. You should plan accordingly.

Supported environments

All environments deployed through Lifecycle Services are supported.

> **NOTE**
> If you have a build environment, you can only use LCS to apply Binary updates and Data upgrade packages. You can't use LCS to apply an Application Deployable package.

For other environments (listed below), you must use Remote Desktop Protocol (RDP) to connect to the environment and install from the command line. For information about manual package deployment, see Install deployable packages from the command line.

- Local development environments (Downloadable virtual hard disk [VHD])
- Multi-box dev/test environments in Microsoft Azure (Partner and trial projects)

Key concepts

Before you begin, you should understand *deployable packages*, *runbooks*, and the *AXInstaller*. A deployable package is a unit of deployment that can be applied in any environment. A deployable package can be a binary update to the platform or other runtime components, an updated application (AOT) package, or a new application (AOT) package. The AXInstaller creates a runbook that enables installing a package. For more details, see Packages, runbooks, and the AXUpdateInstaller in depth at the end of this topic.

Supported package types

- **AOT deployable package** – A deployable package that is generated from application metadata and source code. This deployable package is created in a development or build environment.
- **Application and Platform Binary update package** – A deployable package that contains dynamic-link libraries (DLLs) and other binaries and metadata that the platform and application depend on. This is a package released by Microsoft. This is available from the **All binary updates** tile from LCS.
- **Platform update package** – A deployable package that contains dynamic-link libraries (DLLs) and other binaries and metadata that the platform depend on. This is a package released by Microsoft. This is available from the **Platform binary updates** tile from LCS.
- **Commerce deployable package** – A combination of various packages that are generated after the Commerce code is combined.
- **Merged package** – A package that is created by combining one package of each type. For example, you can

merge one binary update package and one AOT package, or one AOT package and one Commerce deployable package. The packages are merged in the Asset library for the project in LCS.

> **NOTE**
>
> A binary package and a Commerce deployable package can't be included in the same merged package.
>
> For information about how to download an update from LCS and what you see in the tiles based on your environment version, see Download updates from Lifecycle Services (LCS).
>
> If your environment is on an application version 8.1 and later, then the **Platform Update package** does not apply to your environment. Starting with 8.1 and later releases, **Application and Platform Binary update package** is the one that applies since application and platform will be combined into a single cumulative package and will be released by Microsoft. Also note that you will no longer be applying granular X++ hotfixes and will get all application and platform updates together. This means that on the environment details page, clicking on **View detailed version information** will not have details on the granular hotfixes or KBs applied as there is no way to apply them.

Prerequisite steps

- **Make sure that the package that should be applied is valid.** When a package is uploaded to the Asset library, it isn't analyzed. If you select the package, the package status appears in the right pane as **Not Validated**. A package must pass validation before it can be applied in an environment by using the following procedures. The status of the package will be updated in the Asset library to indicate whether the package is valid. We require validation to help ensure that production environments aren't affected by packages that don't meet the guidelines.

  There are three types of validations:

  - Basic package format validations
  - Platform version checks
  - Types of packages
- **Make sure that the package is applied in a sandbox environment before it's applied in the production environment.** To help ensure that the production environment is always in a good state, we want to make sure that the package is tested in a sandbox environment before it's applied in the production environment. Therefore, before you request that the package be applied in your production environment, make sure that it has been applied in your sandbox environment by using the automated flows.
- **If you want to apply multiple packages, create a merged package that can be applied first in a sandbox environment and then in the production environment.** Application of a single package in an average environment requires about 5 hours of downtime. To avoid additional hours of downtime when you must apply multiple packages, you can create a single combined package that contains one package of each type. If you select a binary package and an application deployable package in the Asset library, a **Merge** button becomes available on the toolbar. By clicking this button, you can merge the two packages into a single package and therefore reduce the total downtime by half.
- **Make sure that the Application binary update package is applied to your dev/build environment before it is applied to your sandbox and production environment** - If the application binary package is applied directly to your Tier 2+ sandbox environment but is not applied on your dev/build environment, the next time you move an AOT package from dev/build box (which does not have the same application binaries as your sandbox environment) to sandbox, some of the application binaries will be overwritten with what is in your dev/build environment. This could result in a regression of the version of your sandbox environment.

Apply a package to a non-production environment by using LCS

Before you begin, verify that the deployable package has been uploaded to the Asset library in LCS.

1. For a binary update, upload the package directly to the Asset library. For information about how to download an update from LCS, see Download updates from Lifecycle Services (LCS). For an application (AOT) deployable package that results from an X++ hotfix, or from application customizations and extensions, create the deployable package in your development or build environment, and then upload it to the Asset library.
2. Open the **Environment details** view for the environment where you want to apply the update.
3. Click **Maintain** > **Apply updates** to apply an update.
4. Select the package to apply. Use the filter at the top to find your package.
5. Click **Apply**. Notice that the status in the upper-right corner of the **Environment details** view changes from **Queued** to **In Progress**, and an **Environment updates** section now shows the progress of the package. You can refresh the page to check the status.
6. Continue to refresh the page to see the status updates for the package application request. When the package has been applied, the environment status changes to **Deployed**, and the servicing status changes to **Completed**.

Apply a package to a production environment by using LCS

In a production environment, customers can schedule a downtime for when they want the update to be applied.

> **IMPORTANT**
> An important prerequisite for applying a package to a production environment is that the package must be successfully applied to at least one sandbox environment in the same project.

1. After the update is successfully applied in a sandbox environment, go to the project's asset library. On the **Asset library** page, select the **Software deployable package** tab, select the package that you want to move to production, and click **Release candidate**. This indicates that this package is ready for production deployment.
2. Open the **Environment details** view for the production environment where you want to apply the package.
3. Select **Maintain** > **Apply updates** to apply the package.
4. Select the package to apply in your production environment, and then click **Schedule** to submit a request to apply it.

   > **NOTE**
   > The list of packages includes only the packages that have been successfully signed off in the sandbox environment, and that have been marked as release candidates.

5. Specify the date and time to schedule the package application. Click **Submit**, and then click **OK** to confirm. Note that your environments will be unavailable to perform business while the package is being applied.
6. At the scheduled downtime, package deployment will start.
7. After the environment is serviced, you can monitor the status. The **Servicing status** field indicates the status of package application. Additionally, a progress indicator shows the number of steps that have

been run, out of the total number of steps that are available.

8. After the deployment is successfully completed, the **Servicing status** field is set to **Completed**.
9. If package application isn't successfully completed, Microsoft will investigate the issue. The **Servicing status** field will indicate that package application has failed. The environment will be rolled back to a good state.

Troubleshoot package deployment failures

If package deployment fails, see the Troubleshoot package application issues topic.

Applying updates and extensions

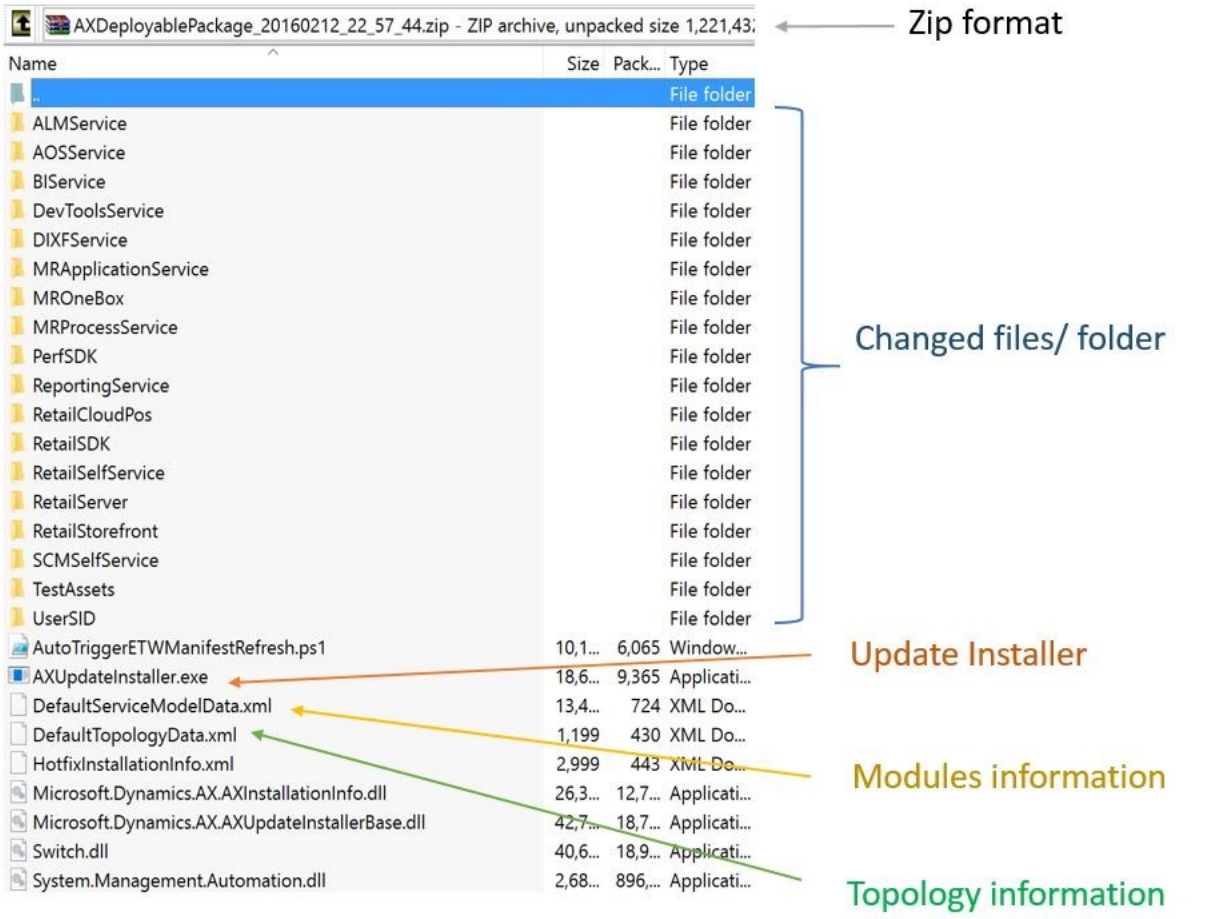
If you are updating a Tier-2 Sandbox or Production environment on application version 8.1.2.x or newer and have initialized Cloud Scale Unit, you will also need to update Commerce channel components. For more information, see Update Retail Cloud Scale Unit.

If you're using components (such as Modern POS), after you've applied updates and extensions in your environment, you must also update your in-store components. For more information, see Configure, install, and activate Modern POS (MPOS).

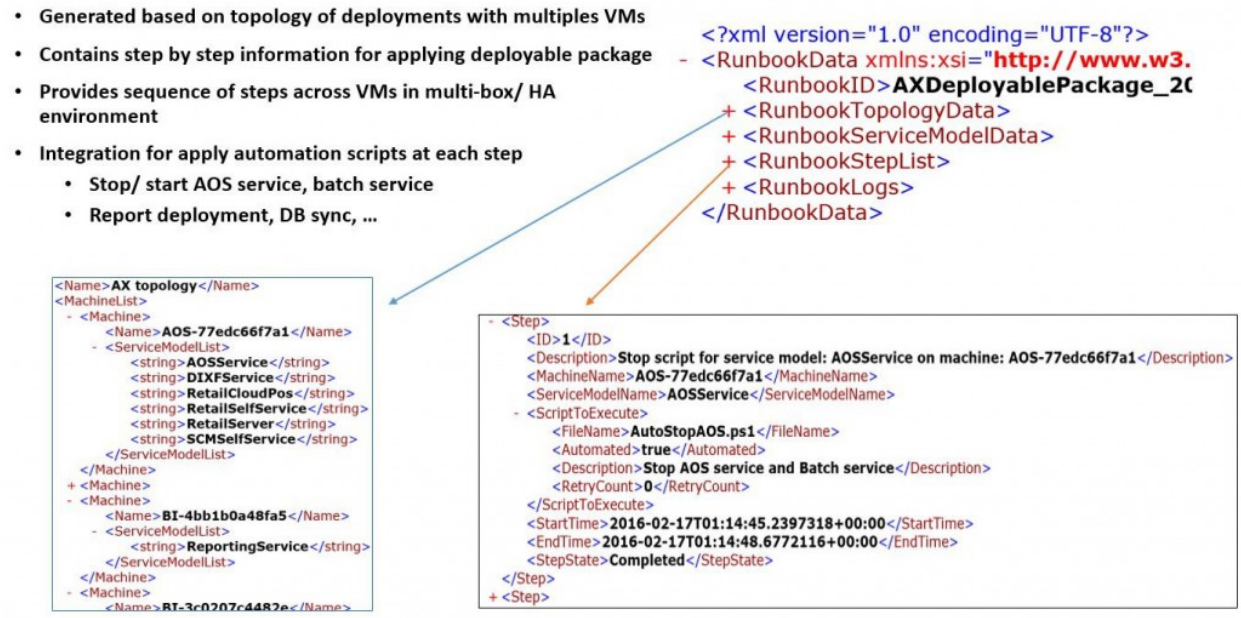
Packages, runbooks, and the AXUpdateInstaller in depth

Deployable packages, runbooks, and the AXUpdateInstaller are the tools you use to apply updates.

Deployable package – A deployable package is a unit of deployment that can be applied in an environment. A deployable package can be a binary update to the platform or other runtime components, an updated application (AOT) package, or a new application (AOT) package. Deployable packages downloaded from LCS or created in a development environment cannot be applied across product types. For example, a Finance deployable package cannot be applied in a Commerce app environment, and vice versa. If you have an existing customization for a Finance and Operations app that is compatible with the Commerce app, and you would like to apply it to a Commerce environment, you will need to re-package your source code in a Commerce development environment, and conversely if moving in the other direction.



Runbook – The deployment runbook is a series of steps that are generated in order to apply the deployable package to the target environment. Some steps are automated, and some steps are manual. AXUpdateInstaller lets you run these steps one at a time and in the correct order.



AXUpdateInstaller – When you create a customization package from Microsoft Visual Studio or a Microsoft binary update, the installer executable is bundled together with the deployable package. The installer generates the runbook for the specified topology. The installer can also run steps in order, according to the runbook for a specific topology.

Additional resources

Install deployable packages from the command line

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Install metadata hotfixes in development environments

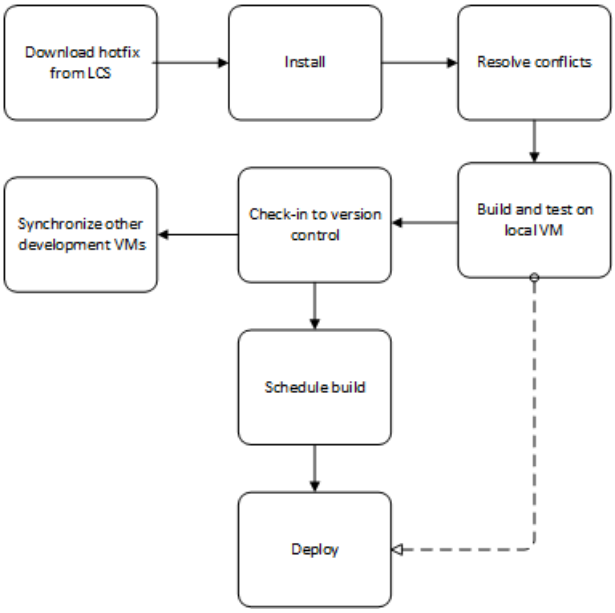
2/18/2021 • 6 minutes to read • Edit Online

This topic will guide you through installing an Application Metadata hotfix on your development environment.

A metadata hotfix package contains changes (metadata or X++ source code) to model elements (XML files) in your development environment. A hotfix can also contain new model elements. A metadata hotfix package is in the form of an SCDP file. This article describes the process for installing a metadata hotfix package and explains how to share the package with other developers who are working on the same project.

Overall flow

The following diagram shows the overall flow.



Download the hotfix from LCS

For instructions about how to download a hotfix, see Download updates from Lifecycle Services (LCS). After you download the zip file, extract the SCDP metadata hotfix package from it, and put it in a local folder.

Install the hotfix

Before you begin

- This topic assumes that your packages folder is located at c:\AOSService\PackagesLocalDirectory\Bin. On some virtual machines (VMs), it might be located at c:\Packages, i:\AOSService\PackagesLocalDirectory\Bin, or k:\AOSService\PackagesLocalDirectory\Bin.
- If you're not using Microsoft Azure DevOps or another source control system, create a backup of your packages folder (which is also known as the metadata store). We don't recommend that you do development unless you use Azure DevOps.
- If you have Azure DevOps or Microsoft Team Foundation Server (TFS) version control, make sure that there are no files in the **Pending Changes** list of your current workspace. If you have pending changes, we recommend that you submit them or shelve them before you install the metadata hotfix.

Install the metadata hotfix package

To invoke the installation of the metadata hotfix, you can call the SCDPBundleInstall.exe utility from a command prompt. SCDPBundleInstall.exe is located in your packages bin folder.

Without version control (not recommended)

If you're not using Azure DevOps or TFS for source control, use the following command.

```
SCDPBundleInstall.exe -install -packagepath=<scdp file containing the hotfix> -metadatastorepath=<metadata
packages root folder>
```

With version control (recommended)

If you're using Azure DevOps or TFS for source control, follow the steps below: **Prepare** the installation of the hotfix package using the command below. This step is not available if you are using a platform that is older than Platform update 2 (August 2016))

```
SCDPBundleInstall.exe -prepare -packagepath=<scdp file containing the hotfixes> -metadatastorepath=<metadata
packages root folder> -tfsworkspacepath=<path of local workspace folder> -tfsprojecturi=<URI of the Azure
DevOps or TFS project collection>
```

This will create a changeset of all the existing files on your environment that will be modified by the hotfix package, the prepare command will not install the hotfixes. Here is an example.

```
SCDPBundleInstall.exe -prepare -packagepath=c:\temp\hotfixbundle1234.axscdppkg -metadatastorepath=
c:\AOSService\PackagesLocalDirectory -tfsworkspacepath= c:\AOSService\PackagesLocalDirectory -
tfsprojecturi=https://myaccount.visualstudio.com/defaultcollection
```

Check-in your pending changes to create a backup of these files in your version control system. This will enable rolling back the hotfixes if needed. **Install** the hotfix package using the command below.

```
SCDPBundleInstall.exe -install -packagepath=<scdp file containing the hotfixes> -metadatastorepath=<metadata
packages root folder> -tfsworkspacepath=<path of local workspace folder> -tfsprojecturi=<URI of the Azure
DevOps or TFS project collection>
```

If you are using a platform that is older than Platform update 2 (August 2016), you do not need to specify the **-install** option. Here is an example.

```
SCDPBundleInstall.exe -install -packagepath=c:\temp\hotfixbundle1234.axscdppkg -metadatastorepath=
c:\AOSService\PackagesLocalDirectory -tfsworkspacepath= c:\AOSService\PackagesLocalDirectory -
tfsprojecturi=https://myaccount.visualstudio.com/defaultcollection
```

Azure DevOps/TFS parameters let you add the files that are modified by the package to your list of pending changes in Team Explorer.

Required parameters

```
/packagepath=[Path of the local scdp file containing the hotfixes downloaded from Lifecycle Service (LCS)]

/metadatastorepath=[Path of the local metadata store folder, such as c:\AOSService\PackagesLocalDirectory]
```

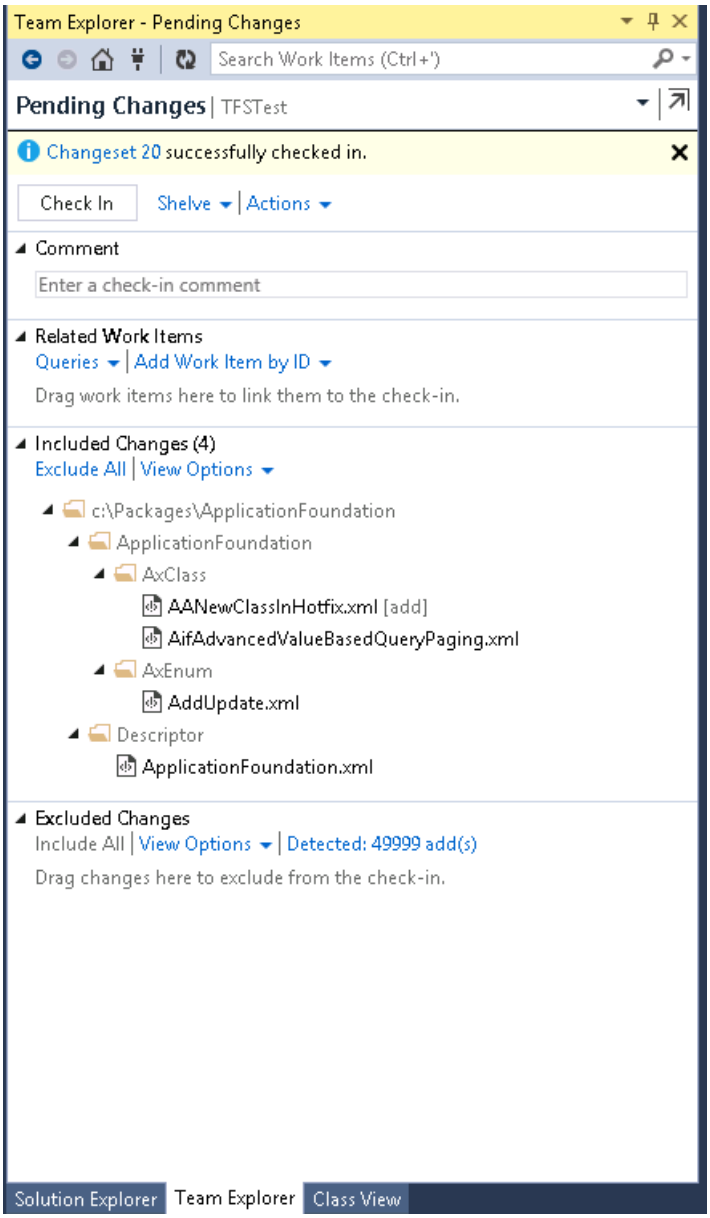
TFS parameters

If you're using Azure DevOps or TFS for source control, you should specify the following two parameters.

```
/tfsprojecturi=[URI of the TFS Project to connect to]

/tfsworkspacepath=[Path of the local workspace, usually equal to the metadatastorepath]
```

After the install command is invoked, the package installation process begins. As part of the installation process, some XML files in your metadata store folder will be updated to reflect the changes that were made in the fix itself. If you're using Azure DevOps or TFS, these files will be added to the list of included changes in the **Pending Changes** window in Team Explorer.



Resolve conflicts that are generated by the installation of the hotfix

Sometimes, a metadata hotfix package contains changes to objects that have been customized in higher-layer models. In this case, the installation process automatically generates conflicts that must be resolved after the hotfix has been installed. The development tools let you create a project that groups all items that have conflicts. For example, if you have a VAR layer model in the Application Suite package that customizes the VendTable form, and you install a hotfix that modifies the VendTable form in the Sys layer model, conflicts might occur in your VAR layer model.

1. Click **Dynamics 365** > **Addins** > **Create project from conflicts**.
2. In the dialog box, select a model to check for conflicts.
3. Click **Create project**. A project is generated that contains only those elements in the selected model that

were found to have conflicts after the hotfix was applied.

4. Open the designer for the conflicting element to view conflicts, and resolve them by using the tools that are provided.

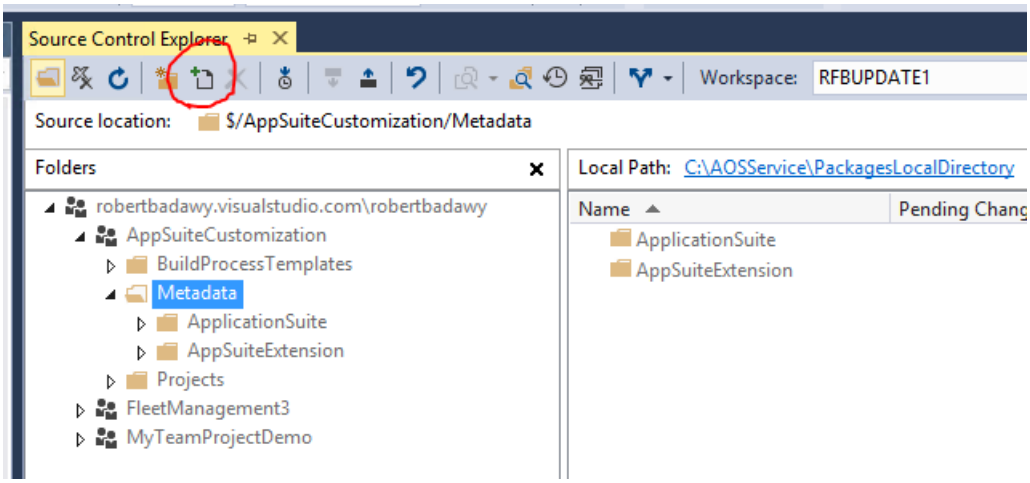
Build and test on a local VM

Build all models that are affected by the hotfix, and test your application.

Check pending changes in to version control

When you're satisfied with all changes that are related to this update, check in your pending changes to Azure DevOps by using Team Explorer in Microsoft Visual Studio. Enter a comment in the **Comment** field, and then click **Check In**. A history of the changes is preserved in your source code repository.

> **NOTE**
>
> If you use a build environment for build and test automation, the build automation process can build metadata hotfix files are in your Azure DevOps project. A hotfix can be built only if the descriptor file of the model that it belongs to is checked in to version control. For example, if the hotfix installation process modified a file that belongs to the Directory model (this file will appear in your list of pending changes), make sure that the descriptor file of the Directory model (c:\AOSService\PackagesLocalDirectory\Directory\Descriptor\Directory.xml) is already checked in to your Azure DevOps project. If it isn't checked in, manually add it to your list of pending changes before you check in by using Source Control Explorer.



Synchronize other development VMs

After a hotfix has been installed on a development VM as described in this article, you don't have to reinstall, resolve conflicts, and validate on other development VMs that are connected to the same Azure DevOps project. Developers and testers who are connected to the same Azure DevOps project can just synchronize the changes into their local VM and then build.

Deploy

After you've applied a metadata hotfix to your development environment, resolved conflicts, and validated your changes, you must create a deployable package and apply your changes to your test or sandbox environment. If you use a build instance for build and test automation, the build process will automatically create the deployable package for you. For more information, see Create deployable packages of models.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Patch SQL Server Reporting Services (SSRS) in one-box environments

2/18/2021 • 3 minutes to read • Edit Online

The following procedure is for one-box development environments only.

Patch the Reporting Service

The following procedure is for one-box development environments only.

- Download the patch .zip file from Lifecycle Services (LCS).
- If there are any font files in the Reporting Service patch's data folder, install these to the machine where SQL Server Reporting Services (SSRS) is running. For more information about installing fonts on Windows, see How to install or remove a font in Windows. Any fonts that have already been installed do not need to be installed again.
- Copy the files in the Reporting Services patch scripts folder to the Report plug-in folder located under C:\Packages\Plugins\AxReportVmRoleStartupTask.
- Change the directory to the Report plug-in folder where you stored the script files.
- Using one of the methods listed below, replace the old instance of reporting extensions.
  - Remove/reinstall the reporting extension. The remove/reinstall option requires that redeploy all reports after you have finished the reinstallation..
  - Manually copy binaries to the sql server binary folder. If you choose to manually copy the files, then you do not need to redeploy reports.

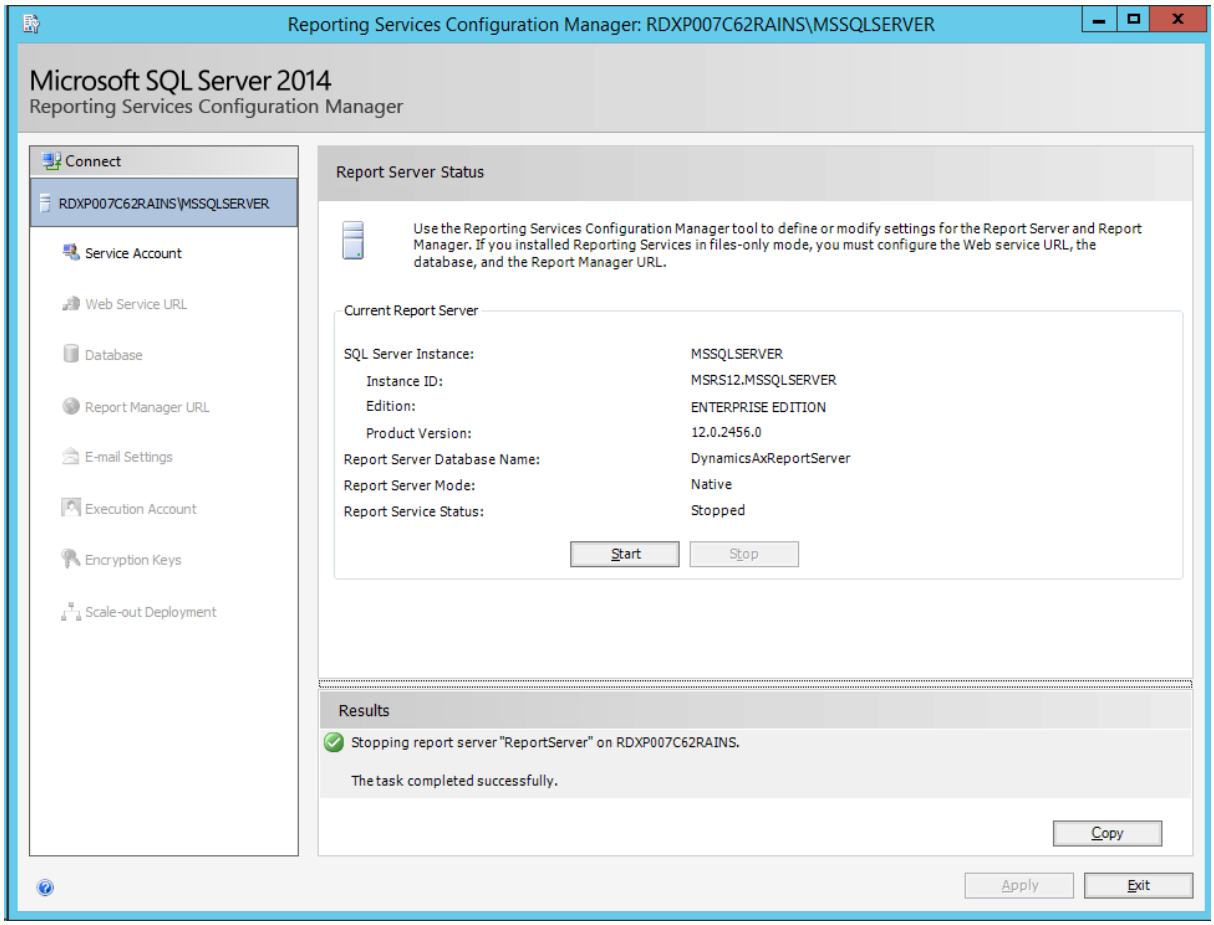
Remove/reinstall the reporting extension

Complete the following procedure as a user in the administrator group for the machine where SSRS is running.

- Using Windows PowerShell, remove the Dynamics SSRS extension by running the following script:
  - PowerShell .\DeploySsrsExtension.ps1 –UninstallOnly
- In PowerShell, reinstall the Dynamics SSRS extension by running the following script:
  - PowerShell .\DeploySsrsExtension.ps1
- Removing the reporting extension removes all the reports. If you have removed and then reinstalled the reporting extension, it is necessary to re-deploy the reports by running the following script:
  - Powershell .\DeployAllReportsToSsrs.ps1
- This task will take 20 to 30 minutes to complete.

Manually copy binaries to the SQL Server binary folder

1. Stop SQL Server Reporting Services. This can be done either from the **Services management** console or from the **Reporting Services Configuration Manager**.



2. Find the SQL Server Reporting Services binary folder. This folder is usually located at C:\Program Files\Microsoft SQL Server\MSRS11.MSSQLSERVER\Reporting Services\ReportServer\bin.
3. If any of the following files are in the patch, copy them to the SQL Server Reporting Services bin folder.* *

> **NOTE**
>
> Patches can either be full patches, which would contain all of the files used by the service, or incremental patches, which contain only the files that have changed. If you have an incremental patch, then some files may not be included. Files not included in the patch do not need to be replaced.

- Microsoft.Dynamics.AX.Framework.Services.Platform.Client.dll
- Microsoft.Dynamics.Framework.ReportsExtensions.dll
- Microsoft.Dynamics.Framework.Reports.dll
- Microsoft.Dynamics.ApplicationPlatform.SSRSReportRuntime.Instrumentation.dll
- Microsoft.Dynamics.ApplicationPlatform.SSRSReportRuntime.man
- Microsoft.Dynamics.Platform.Integration.ClientSdk.Abstraction.dll
- Microsoft.Dynamics.AX.Framework.Reports.Shared.dll
- Microsoft.Dynamics.AX.Framework.EncryptionEngine.dll
- Microsoft.Dynamics.AX.Framework.Utilities.dll
- Microsoft.Dynamics.ApplicationSuite.Reporting.BusinessLogic.dll
- Microsoft.Dynamics.ApplicationPlatform.Environment.dll
- Microsoft.Dynamics.AX.ReportConfiguration.axc
- Microsoft.WindowsAzure.ServiceRuntime.dll
- Microsoft.IdentityModel.dll
- msshrtmi.dll

Restart SQL Server Reporting Services.

Reporting service installation

The following changes are made with the reporting service installation: The following files will be copied into Reporting service bin folder (C:\Program Files\Microsoft SQL Server\MSRS11.MSSQLSERVER\Reporting Services\ReportServer\bin), and the corresponding SSRS config files will be updated so that SSRS is aware of the extension.

- Dynamics.AX.Framework.Services.Platform.Client.dll
- Dynamics.Framework.ReportsExtensions.dll
- Dynamics.Framework.Reports.dll
- Dynamics.ApplicationPlatform.SSRSReportRuntime.Instrumentation.dll
- Dynamics.ApplicationPlatform.SSRSReportRuntime.man
- Dynamics.Platform.Integration.ClientSdk.Abstraction.dll
- Dynamics.AX.Framework.Reports.Shared.dll
- Dynamics.AX.Framework.EncryptionEngine.dll
- Dynamics.AX.Framework.Utilities.dll
- Dynamics.ApplicationSuite.Reporting.BusinessLogic.dll
- Dynamics.ApplicationPlatform.Environment.dll
- Dynamics.AX.ReportConfiguration.axc
- WindowsAzure.ServiceRuntime.dll
- IdentityModel.dll
- msshrtmi.dll

An SSRS service account will be updated to use the local system. A new SSRS catalog database DynamicsAxReportServer and temp database DynamicsAxReportServerTempDB database will be created, and SSRS will be configured to use these two databases. The default catalog database ReportServer and ReportServerTempDBstill exist, but are set to not be used by reporting services. The SSRS service will be updated to use Windows Authentication. An xml configuration file ReportPVMConfiguration.xml will be created in the SSRS bin folder for the report runtime. A report root folder named **Dynamics** and a new security role named **DynamicsBrowser** will be created. Both AOS Web application AppPool identity and batch service account will be added to this custom role. Note that during deployment, the report folder will be deleted and then recreated. Therefore all the previously deployed reports will be deleted from the SSRS server. After you reinstall the reporting extension, you must redeploy the reports.

Update the Visual Studio development tools

2/18/2021 • 2 minutes to read • Edit Online

This topic explains how to update the development tools.

Use this tutorial to update your Visual Studio development tools with a new version. It explains how to uninstall your existing Visual Studio development tools and install the new extension. The new extension is in the form of an installable VSIX file. This file is a part of the binary hotfix available on the Dynamics Lifecycle Services (LCS) site. The VSIX file is located in the **DevToolsService\Scripts** folder of the binary hotfix package.

> **NOTE**
> You do not need to follow the instructions in this article if you are upgrading your Finance and Operations platform to Platform update 4 or newer. It is an automatic step that is part of the platform upgrade process.

Uninstall the existing Visual Studio extension

In order to install a new version of the development tools, you'll need to uninstall the existing version first. Verify the version of the development tools that you have installed. If you don't have it installed, you can skip this section.

Verify your current version of the Visual Studio extension

1. Open the Visual Studio **Help > About Microsoft Visual Studio** dialog and find **Finance and Operations Developer Tools**.
2. Select it and click **OK**.

Uninstall the extension

1. Open the Visual Studio **Tools > Extensions and Updates** dialog.
2. Select **Finance and Operations Visual Studio Tools** and click **Uninstall**.
3. When the extension is uninstalled, exit Visual Studio.

Install a new version of the extension

1. Make sure Visual Studio is not running.
2. Double-click (or right-click and **Open**) the VSIX file of the new version.
3. Follow the installation instructions.
4. When installation is complete, you can start Visual Studio and start developing your application.

> **NOTE**
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Removed or deprecated features in previous releases

2/18/2021 • 51 minutes to read • Edit Online

NOTE

Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. For more information, see Power Automate Blog.
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. For more information, see Terminology updates.

This topic will be updated soon to reflect the latest terminology.

IMPORTANT

This topic is no longer updated. To see a current list of features that have been removed or deprecated from Finance and Operations apps, search for **"Removed or deprecated features"** content that relates to the app you're using.

This topic describes features that have been removed or deprecated from Dynamics 365 for Finance and Operations and previous releases of that product.

- A *removed* feature is no longer available in the product.
- A *deprecated* feature is not in active development and may be removed in a future update.

This list is intended to help you consider these removals and deprecations for your own planning.

Detailed information about objects in Finance and Operations apps can be found in the Technical reference reports. You can compare the different versions of these reports to learn about objects that have changed or been removed in each version of Finance and Operations apps.

Finance 10.0.7 with Platform update 31

Chinese voucher types without Account groups selection

Reason for deprecation/removal	Changed to the feature with account groups selection.
Replaced by another feature?	Yes
Product areas affected	Application
Deployment option	All
Status	Deprecated: By December 1, 2020, we plan to no longer support Chinese voucher types setup without Account groups selection. Find more details about new feature design in What's new in 10.0.7

Finance and Operations 10.0.6 with Platform update 30

DimensionHash.getHash(str _message)

Reason for deprecation/removal	Windows is deprecating the use of SHA1, as documented in Windows Enforcement of SHA1 Certificates.
Replaced by another feature?	Yes
Product areas affected	Application
Deployment option	All
Status	Deprecated: By April 1, 2020, developers must use the platform APIs found in the class **HasFunction**.

Hash.ComputeSHA1Hash(string message)

Reason for deprecation/removal	Windows is deprecating the use of SHA1, as documented in Windows Enforcement of SHA1 Certificates.
Replaced by another feature?	Yes
Product areas affected	Platform
Deployment option	All
Status	Deprecated: By April 1, 2020, developers must use the platform APIs found in the class **HasFunction**.

FormDateTimeControl.setUtcString()

Reason for deprecation/removal	We are retiring the **setUtcString()** method, because a better replacement method is available.
Replaced by another feature?	Yes
Product areas affected	Platform
Deployment option	All
Status	Deprecated: By October 1, 2020, we plan to no longer support the **setUtcString()** method. Developers should be using the **setUtcDateTime()** method instead.

Blacklist report (IT) – Feature reference IT-00001

Reason for deprecation/removal	Not legally required.
Replaced by another feature?	No
Product areas affected	Italian localization
Deployment option	All
Status	Deprecated: By October 1, 2020, we plan to no longer support the **Blacklist report (IT) – Feature reference IT-00001**.

Domestic tax report – Feature reference IT-00003

Reason for deprecation/removal	Not legally required.
Replaced by another feature?	No
Product areas affected	Italian localization
Deployment option	All
Status	Deprecated: By October 1, 2020, we plan to no longer support the **Domestic tax report – Feature reference IT-00003**.

Finance and Operations 10.0.5 with Platform update 29

US Payroll tax updates

Reason for deprecation/removal	We are retiring tax updates for the US Payroll functionality due to low usage and enhanced functionality that is now offered via strategic integrations.
Replaced by another feature?	Yes
Product areas affected	Payroll
Deployment option	All
Status	Deprecated: By July 31, 2024, we plan to no longer provide tax updates to US Payroll customers. The functionality will remain in the product, but enhancements will no longer keep the functionality up to date, and any product defects will be evaluated on a case-by-case basis.

> **NOTE**
>
> This represents a change from the original discontinuation date of October 1, 2021. For more information, see Tax updates being retired for US Payroll feature in Microsoft Dynamics 365 for Finance and Operations.

Data management staging clean up

Reason for deprecation/removal	Does not meet the core requirements that are needed for scheduling periodic cleanup.
Replaced by another feature?	Yes, the Job history cleanup feature is being added to meet the scenarios holistically.
Product areas affected	Data management
Deployment option	All
Status	Deprecated: Target timeframe for the functionality to be removed is December 2020.

Finance and Operations 10.0.4 with Platform update 28

France: FEC Accounting data export in XML

Reason for deprecation/removal	Replaced by TXT format, **French FEC audit file** is available through **General ledger** > **Periodic tasks** > **Data export**.
Replaced by another feature?	Yes
Product areas affected	General ledger
Deployment option	All
Status	Deprecated. Target timeframe for the functionality to be removed is July 2020.

Legacy navigation bar

Reason for deprecation/removal	Header alignment with other Dynamics and Office products. For more details, see Updated navigation bar that aligns with the Office header.
Replaced by another feature?	Starting in Platform update 24, a restyled navigation bar that features search was introduced.
Product areas affected	Web client

Deployment option	All
Status	Deprecated: Starting in April 2020, the legacy navigation bar will no longer be available. Until that point, customers can revert to the legacy navigation bar through the **Client performance options** page.

Finance and Operations 10.0.2 with Platform update 26

Legacy default action behavior

Reason for deprecation/removal	The legacy behavior for default actions in grids results in an unexpected column having the default action link after grid columns have been reordered via personalization. The new sticky default action feature corrects this. For more details, see Sticky default actions in grids.
Replaced by another feature?	Starting in Platform update 21, a feature for "sticky default actions" was introduced. This feature can be enabled on the **Client performance options** page.
Product areas affected	Grids in the web client
Deployment option	All
Status	Deprecated: Starting in April 2020, sticky default actions will be the default behavior, without a mechanism to revert to the legacy behavior.

Legacy "is one of" filtering experience

Reason for deprecation/removal	The "is one of" filtering experience went through a redesign in Platform update 22,with the plan for this to eventually be the only "is one of" filtering experience.
Replaced by another feature?	Starting in Platform update 22, an improved "is one of" filtering experience became available on the **Client performance options** page. For more information, see Optimized is one of filtering experience.
Product areas affected	Web client
Deployment option	All
Status	Deprecated: Starting in April 2020, the improved "is one of" experience will be the default behavior, without a mechanism to revert to the legacy behavior.

Parameter to enable sales orders with multiple project contract funding sources

Support for creating project-based sales orders where the project contract has multiple funding sources is

enabled with the **Project management parameters** setting **Allow sales orders for project with multiple funding sources**. By default, this parameter is not enabled.

Reason for deprecation/removal	The functionality will always be enabled after the parameter is removed.
Replaced by another feature?	No. The functionality to support project-based sales orders with multiple funding sources will always be enabled.
Product areas affected	The **Allow sales orders for projects with multiple funding sources** parameter will be removed. The following methods will be modified when the parameter is removed: **ctrlSalesOrderTable** method in **ProjStatusType** class, **validate** method for **ProjId** field, and **run** method in **SalescreateOrder** form. The following methods will be deprecated when the parameter is removed: **IsSalesOrderAllowedForMultipleFundingSources** in **ProjTable** table file, **IsAllowSalesOrdersForMultipleFundingSourcesParamEnabled** method in **ProjTable** table file, **AllowSalesOrdersForMultipleFundingSources** data field in **ProjParameters** form and **ProjParameterEntity** files, **IsAssociatedToMultipleFundingSourcesContract** private method in **ProjTable** table file.
Deployment option	All
Status	Deprecation is planned for the April 2020 release wave.

Legacy workflow reports for tracking and instance status

Reason for deprecation/removal	The legacy workflow reports for tracking and instance status are being deprecated because they are no longer referenced from the navigation. The report names are WorkflowWorkflowInstanceByStatusReport and WorkflowWorkflowTrackingReport.
Replaced by another feature?	The workflow history form can be used instead.
Product areas affected	Web client
Deployment option	All
Status	Deprecated: Target timeframe for the functionality to be removed is April 2020.

Finance and Operations 10.0.1 with Platform update 25

Deprecated APIs and potential breaking changes

Deriving from internal classes is deprecated

Reason for deprecation/removal	Before Platform update 25, it was possible to create a class or table that derives from an internal class/table that is defined in another package/module. This is not a safe coding practice. As of Platform update 25, the compiler will display a warning.
Replaced by another feature?	The compiler warning will be replaced by an error in Platform update 26. This change is backward compatible at runtime, which means that Platform update 25 or newer can be deployed on any sandbox or production environment without the need to modify custom code. This change only affects development and compile time.
Product areas affected	Visual Studio development tools
Deployment option	All
Status	Deprecated: The warning will become a compilation error in Platform update 26.

Overriding internal methods is deprecated

Reason for deprecation/removal	Before Platform update 25, it was possible to override an internal method in a derived class that is defined in another package/module. This is not a safe coding practice. As of Platform update 25, the compiler will display a warning.
Replaced by another feature?	This warning will be replaced by a compile error in Platform update 26. This change is backward compatible at runtime, which means that Platform update 25 or newer can be deployed on any sandbox or production environment without the need to modify custom code. This change only affects development and compile time.
Product areas affected	Visual Studio development tools
Deployment option	All
Status	Deprecated: The warning will become a compilation error in Platform update 26.

Finance and Operations 10.0.0 with Platform update 24

Renaming released products

Reason for deprecation/removal	When you use the **Rename primary key** function to change the ItemId of a released product, only direct foreign key references are updated. Any other references to the released product, such as from production orders, will retain the old ItemId. As a result, there could be inconsistent data that will eventually block business processes.

Replaced by another feature?	No.
Product areas affected	Product information management
Deployment option	All
Status	Removed as of Finance and Operations 10.0.0 with Platform update 24.

Finance and Operations 8.1.3 with Platform update 23

SQL Server Reporting Services ReportViewer Control

Customers can use the **Export** action provided by the embedded SQL Server Reporting Services (SSRS) ReportViewer control to download documents produced by Finance and Operations applications. This HTML-based presentation of the report offers users a non-paginated preview of the document.

Reason for deprecation/removal	The non-paginated nature of the HTML-based preview experience does **not** deliver fidelity with the physical documents ultimately produced by Finance and Operations. By fully embracing PDF as the standard format for business documents, users are able to take advantage of a modern viewing experience with improved performance when producing application reports.
Replaced by another feature?	Going forward, PDF documents will be the default format for reports rendered by Finance and Operations.
Product areas affected	This change does **not** impact customer scenarios where reports are distributed electronically or sent directly to printers.
Deployment option	All
Status	Deprecated: A removal date has not been set for this feature. The functionality to automatically preview application reports using an embedded PDF viewer is planned for the May 2019 Platform update.

Client KPI controls

Embedded key performance indicators (KPIs) could be modeled in Visual Studio by a developer and further customized by the end user.

Reason for deprecation/removal	The native client controls used to define KPIs have low customer uptake and rely on a developer to add trackable metrics.

Replaced by another feature?	PowerBI.com service delivers world-class tooling for defining and managing KPIs based on data from external sources. In an upcoming release, we plan to enable you to embed solutions hosted on PowerBI.com in application workspaces.
Product areas affected	This update will prevent developers from introducing new KPI controls in Visual Studio designer.
Deployment option	All
Status	Deprecated: A removal date has not been set for this feature.

Deprecated APIs and future breaking changes

Field groups containing invalid field references

Reason for deprecation/removal	It is possible for table metadata definitions to have field groups containing invalid field references. If deployed, this can cause runtime failures in Financial Reporting and SQL Server Reporting Services (SSRS). This issue is currently categorized as a *compiler warning* rather than an *error*, meaning that the deployable package creation and deployment can proceed without fixing the issue. To fix this issue: 1. Remove the invalid field reference from the table field group definition. 2. Recompile. 3. Ensure any warnings or errors are addressed.
Replaced by another feature?	This warning will be replaced by a compile error in the future.
Product areas affected	Visual Studio development tools
Deployment option	All
Status	Deprecated: The warning is a compile-time error with platform updates for version 10.0.11 of Finance and Operations apps.

Complete list

To access the full list of APIs that are being deprecated, see Deprecation of methods and metadata elements.

Finance and Operations 8.1 with Platform update 20

Batch transfer rules for subledger journal account entries

The Synchronous transfer mode is being deprecated in the General ledger parameters. This mode is replaced by Asynchronous and scheduled batch only, which already exist as options for transfer. For additional information, see the General Ledger Parameters – Batch transfer rules blog.

Reason for deprecation/removal	We are removing the synchronous option due to performance impact to the system.
Replaced by another feature?	Asynchronous and scheduled batch are options to use in place of Synchronous.
Product areas affected	General Ledger, Accounts payable, Accounts Receivable, Procurement, Expense
Deployment option	All
Status	Deprecated: Target timeframe for the functionality to be removed is the 10.0 version.

Electronic reporting for Russia

Feature for configuring .txt and .xml file formats of declarations.

Reason for deprecation/removal	Replaced with Electronic reporting.
Replaced by another feature?	Yes.
Product areas affected	General Ledger
Deployment option	All
Status	Removed as of Finance and Operations 8.1 with Platform update 20.

Financial reports generator for Russia

A tool for setting up data collection for accounting and tax reports, and to export data to XLS and DOC report templates. Functional parts: Export data to XLS and DOC report templates, queries, fixed requisites are removed.

Reason for deprecation/removal	Removed parts are replaced with Electronic reporting.
Replaced by another feature?	Yes. Financial reports setup user interface should be used for setting up data collection rules by GL accounts or tax registers. Export data to various file types, fixed requisites and query-like data collection rules should be configured in Electronic reporting.
Product areas affected	General ledger.
Deployment option	All
Status	Removed as of Finance and Operations 8.1 with Platform update 20.

Integration with external providers for sending electronic reporting through communication channels for

Russia

Feature exporting generated electronic files of declarations to folder for further sending to official providers of electronic reporting as well as importing state back.

Reason for deprecation/removal	Replaced with electronic messages configurable feature.
Replaced by another feature?	Yes.
Product areas affected	General Ledger, Tax
Deployment option	All
Status	Removed as of Finance and Operations 8.1 with Platform update 20.

Profit tax register wizard

Feature for creating templates for new profit tax registers. This feature creates X++ objects for new registers, which are then created as templates with the appropriate calculation logic added in.

Reason for deprecation/removal	Feature is not compatible with the Finance and Operations extensibility model.
Replaced by another feature?	No
Product areas affected	Tax
Deployment option	All
Status	Removed as of Finance and Operations 8.1 with Platform update 20.

Finance and Operations 8.0 with Platform update 15

No features have been removed or deprecated with this release. Platform update 15 is cumulative and contains new or changed features from Platform update 13, Platform update 14, and Platform update 15.

Finance and Operations, Enterprise edition 7.3 with Platform update 12

Personalized product recommendations

Starting February 15, 2018, retailers will no longer be able to display personalized product recommendations on a point of sale (POS) device. For more information, see Product recommendations overview.

Reason for deprecation/removal	We are removing the current version of the product recommendation service as we redesign this feature with a better algorithm and newer retail-oriented capabilities.

Replaced by another feature?	No. However, after Spring 2018, we plan to bring back this feature to leverage a new recommendation service.
Product areas affected	Personalized product recommendations in POS.
Deployment option	All
Status	Removed as of February 15, 2018. This affects customers running Dynamics 365 for Operations 1611 and later.

Extension of the list of Electronic reporting (ER) functions

The possibility to introduce custom functions to be used in the ER expression builder (for more information, see Extend the list of Electronic reporting (ER) functions) is not supported any more. Due to changes of the ER APIs, the API to call built-in functions from the ER expression builder became internal and can't be extended any longer.

Reason for deprecation/removal	Code sealing initiative
Replaced by another feature?	None. Whenever a new built-in function is needed, a new extension request must be addressed to the ER framework team. As a temporary work around while the requested function is under development by the ER team, the required logic can be programmed as a method of a custom application class. This method can be accessed in an ER expression as a property of the added ER data source of the **Application\Class** type that refers to that custom application class.
Product areas affected	Electronic reporting framework
Deployment option	All
Status	Removed as of Finance and Operations, Enterprise edition 7.3.

Inventory by item group and Inventory by inventory dimension aging reports

These two reports are no longer supported in Finance and Operations. Instead, the **Inventory aging** report can be used to improve the user experience.

Reason for deprecation	Duplicate functionality
Replaced by another feature?	Yes. The two reports have been replaced by the **Inventory aging** report.
Product areas affected	Inventory management, Cost management

Deployment option	All
Status	Deprecated: The menu items for the two reports have been removed in version 7.3. However, the code for the reports remains in the product. The plan is to remove the code in a future release.

Power BI content packs available on AppSource

The **Cost management**, **Financial performance**, and **Retail channel performance** content packs, available on the Microsoft AppSource site, are deprecated as a consequence of product updates in Microsoft Power BI. System administration forms used to deploy these content packs to PowerBI.com are also being deprecated in Finance and Operations.

Reason for deprecation/removal	Product updates in Microsoft Power BI.
Replaced by another feature?	The **Cost management**, **Financial performance**, and **Retail channel performance** content packs, available on the AppSource site, are being replaced by analytical applications which allow for solution integrations at the database level. For more information about analytical applications, see Embedded Power BI in workspaces.
Product areas affected	Cost management, Finance, and Retail
Deployment option	Cloud only (Integration with PowerBI.com is not supported in on-premises deployments.)
Status	Deprecated: Target timeframe for the functionality removal is Q2 2018.

Standard UI in data management workspace

The standard UI in data management is the legacy UI, which is the default UI presented to the users when they visit the data management workspace.

Reason for deprecation/removal	We are investing in providing new user experiences in the new UI.
Replaced by another feature?	The new UI called *Enhanced views* is replacing the old UI.
Product areas affected	Data management workspace
Deployment option	All
Status	Deprecated: Target timeframe for the functionality to be removed is Q2 2018.

Excise, Sales Tax, Service Tax for India

These taxes have been subsumed into Indian GST.

Reason for removal or deprecation	These taxes have been subsumed into Indian GST.
Replaced by another feature?	Indian GST
Product areas affected	Tax
Deployment option	All modules
Status	Deprecated: A removal date has not been set for this feature.

File Validation Utility (FVU) for India

Reason for removal or deprecation	Lack of customer usage
Replaced by another feature?	No
Product areas affected	Indian withholding tax
Deployment option	All modules
Status	Deprecated: A removal date has not been set for this feature.

TDS/TCS certificate for India

Users can download this from the government portal.

Reason for removal or deprecation	Lack of customer usage
Replaced by another feature?	No
Product areas affected	Indian withholding tax
Deployment option	All modules
Status	Deprecated: A removal date has not been set for this feature.

Export/import (EXIM) incentive scheme for India

Reason for removal or deprecation	Lack of customer usage
Replaced by another feature?	No
Product areas affected	Import and export

Deployment option	All modules
Status	Deprecated: A removal date has not been set for this feature.

Dynamics 365 for Retail 7.2

Personalized product recommendations

Starting February 15, 2018, retailers will no longer be able to display personalized product recommendations on a point of sale (POS) device. For more information, see Product recommendations overview.

Reason for deprecation/removal	We are removing the current version of the product recommendation service as we redesign this feature with a better algorithm and newer retail-oriented capabilities.
Replaced by another feature?	No. However, after Spring 2018, we plan to bring back this feature to leverage a new recommendation service.
Product areas affected	Personalized product recommendations in POS.
Deployment option	All
Status	Removed as of February 15, 2018. This affects customers running Dynamics 365 for Retail 7.2 and later.

Finance and Operations, Enterprise edition July 2017 with Platform update 8

Currency conversion for accounting and reporting currencies

Currency conversion for accounting and reporting currencies was introduced when the euro was introduced.

Reason for deprecation/removal	Limited usage and addition of the Copy legal entity functionality as a replacement.
Replaced by another feature?	No, but the Copy legal entity and Configurations features were added to make it easier to move to a company that has changing core requirements.
Product areas affected	Financial management
Status	Deprecated: A removal date has not been set for this feature.

Warehouse mobile devices portal

Warehouse mobile devices portal (WMDP) was a standalone component that was intended for on-premises self-deployment. This component is no longer supported in Finance and Operations. A native app that improves the user experience has replaced the functionality of WMDP.

Reason for deprecation/removal	Duplicate functionality.
Replaced by another feature?	Yes. This feature has been replaced by Finance and Operations - Warehousing. For more information about setup and prerequisites, see Install and configure the Warehousing app overview.
Product areas affected	Warehouse management, Transportation management
Deployment option	Warehouse mobile devices portal (WMDP) was a standalone component that was intended for on-premises self-deployment.
Status	Deprecated: Target timeframe for the functionality to be removed is Q4 2019.

Advanced bank reconciliation matching rule for manual matching

A matching rule was used to select and mark a bank document when documents were manually matched in the reconciliation worksheet.

Reason for deprecation/removal	Limited usage.
Replaced by another feature?	No. Column filtering capabilities should be used to find documents for reconciliation.
Product areas affected	Cash and bank management
Deployment option	All
Status	Removed as of July 2017.

Dynamics 365 for Operations 1611 with Platform update 3

AEB payment formats for Spain

The Consejo Superior Bancario payment formats were used to send remittance files to the bank for customer payments and vendor payments. The content of these formats was determined by the Asociación Española de Banca. It covers Cuaderno 19, 32, 58, 34.

Reason for deprecation/removal	The payment formats are no longer used.
Replaced by another feature?	Yes, ISO20022 Credit transfer and Direct debit payment formats for Spain
Product areas affected	Accounts payable, Accounts receivable

Status	Deprecated: A removal date has not been set for this feature.

Bank payments transfer for Lithuania

Bank payment transfers were generated and printed by using the Payment transfer (LT) export format for Lithuania. The Lithuanian market began to use LITAS, the unified electronic banking system, in 2005.

Reason for deprecation/removal	The payment formats are no longer used.
Replaced by another feature?	Yes, ISO20022 Credit transfer payment format for Lithuania
Product areas affected	Accounts payable
Status	Deprecated: A removal date has not been set for this feature.

BBS Direkte Remittering payment formats for Norway

BBS Direkte Remittering payment formats include customer payment collection export (direct debit) and return message import.

Reason for deprecation/removal	The payment formats are no longer used.
Replaced by another feature?	The AvtaleGiro customer payment format for Norway can be used to generate direct debit messages. Return message import will be implemented in future releases.
Product areas affected	Accounts payable, Accounts receivable
Status	Deprecated: A removal date has not been set for this feature.

Chart of Accounts tool for Spain

This tool is used when a chart of accounts in Spain requires major changes. Users can import a new chart of accounts in Microsoft Excel or text format, and can also import financial statements.

Reason for deprecation/removal	Limited usage
Replaced by another feature?	No
Product areas affected	General ledger
Status	Deprecated: A removal date has not been set for this feature.

Dom80 payment format for Belgium

Legacy Belgian payment format for payment collection (direct debit).

Reason for deprecation/removal	The payment format is no longer used.
Replaced by another feature?	Yes, ISO 20022 Direct debit payment format for Belgium
Product areas affected	Accounts receivable
Status	Deprecated: A removal date has not been set for this feature.

DTA/EZAG payment formats for Switzerland

DTA/EZAG formats are integrated into the ESR system, because they can carry on the reference number. Because the reference number isn't mandatory, these formats can be used to process any vendor payments. These formats are used by companies that have a bank account in a location other than "Postfinance."

Reason for deprecation/removal	The payment formats are no longer used.
Replaced by another feature?	Yes, ISO20022 Credit transfer payment format for Switzerland
Product areas affected	Accounts payable
Status	Deprecated: A removal date has not been set for this feature.

EDIFACT-DIRDEB payment format for Austria

EDIFACT-DIRDEB payment format for payment collection (direct debit).

Reason for deprecation/removal	The payment format is no longer used.
Replaced by another feature?	Yes, ISO 20022 Direct debit payment format for Austria
Product areas affected	Accounts receivable
Status	Deprecated: A removal date has not been set for this feature.

EDIVAT for Belgium

EDIVAT is an obsolete Belgian standard for electronic declaration via secure mail. Dynamics AX 2012 retains the read-only solution to enable access to the historical data.

Reason for deprecation/removal	The functionality is no longer used.
Replaced by another feature?	No

Product areas affected	General ledger
Status	Deprecated: A removal date has not been set for this feature.

eGiro EDIFACT CREMUL payment import format for Norway

eGiro is based on the international UN EDIFACT CREMUL (Multiple Credit Advice Message) standard that is used for automatic posting of customer payments. In Dynamics AX, eGiro is implemented as a customer payment import format.

Reason for deprecation/removal	The payment format is no longer used.
Replaced by another feature?	Yes, the ISO20022 Camt.054 notification import.
Product areas affected	Accounts receivable
Status	Deprecated: A removal date has not been set for this feature.

External inventory for Poland

Evidence of goods that are taken from a vendor for sales without purchase. Goods that are handled in external inventory don't affect standard inventory, and can be sold and then purchased automatically. This process creates real inventory movements.

Reason for deprecation/removal	Replaced by another feature
Replaced by another feature?	Yes, the core Inbound consignment functionality
Product areas affected	Accounts payable, Inventory management
Status	Deprecated: A removal date has not been set for this feature.

Financial reports generator for Eastern Europe

A tool is used to set up data collection for accounting and tax reports, and to export data to XLS and DOC report templates.

Reason for deprecation/removal	Limited usage
Replaced by another feature?	No. The tool will be replaced by Electronic reporting configurations in future releases.
Product areas affected	General Ledger

Status	Deprecated: A removal date has not been set for this feature.

Import of customer payment transactions for Finland

You can select an import format for Finnish payments to import customer payment transactions from an external file that the bank provides.

Reason for deprecation/removal	The payment format is no longer used.
Replaced by another feature?	Yes, the ISO20022 Camt.054 notification import.
Product areas affected	Accounts receivable
Status	Deprecated: A removal date has not been set for this feature.

Import of payment transactions into a general ledger journal for Finland

A format that is specific to Finland is used to import accounting transactions into the general ledger.

Reason for deprecation/removal	The payment format is no longer used.
Replaced by another feature?	Yes, the ISO20022 Camt.053 bank statement import using Advanced Bank Reconciliation.
Product areas affected	Accounts receivable
Status	Deprecated: A removal date has not been set for this feature.

Integration with Isabel synchronized (CIS) for Belgium

Isabel is the framework for electronic banking in Europe and is a de-facto standard in Belgium.

Reason for deprecation/removal	Integration with Isabel client has been discontinued.
Replaced by another feature?	No. The payment formats that are no longer used are replaced by ISO20022 Credit transfer payment format for Belgium.
Product areas affected	Accounts payable
Status	Deprecated: A removal date has not been set for this feature.

Modifications in the chart of accounts and accounting rules for Spain

This feature is used for changes in the chart of accounts and accounting rules in Spain. It maps accounts to help

transform the old chart of accounts into the new chart of accounts, and compares the previous fiscal year with the new fiscal year, even if they were posted to different account numbers.

Reason for deprecation/removal	Limited usage
Replaced by another feature?	No
Product areas affected	General ledger
Status	Deprecated: A removal date has not been set for this feature.

Pagamento Fornittori vendor payment format

Legacy Italian payment format for credit transfers.

Reason for deprecation/removal	The payment format is no longer used.
Replaced by another feature?	Yes, ISO20022 Credit transfer payment format for Italy
Product areas affected	Accounts payable
Status	Deprecated: A removal date has not been set for this feature.

Payment export formats for Estonia

The Telehansa and Teleservice formats are used for bank payment export.

Reason for deprecation/removal	The payment formats are no longer used.
Replaced by another feature?	Yes, ISO20022 Credit transfer payment format for Estonia
Product areas affected	Accounts payable
Status	Deprecated: A removal date has not been set for this feature.

Payment file archive for Norway

When payment files are generated, the file archive automatically archives all files that are created, even files that were previously written or read.

Reason for deprecation/removal	Replaced by another feature
Replaced by another feature?	Yes, Electronic reporting archived jobs

Product areas affected	Accounts payable, Accounts receivable, Organization administration
Status	Deprecated: A removal date has not been set for this feature.

Payment import formats for Estonia

The Telehansa and TeleTeenus formats are used for bank payment import.

Reason for deprecation/removal	The payment formats are no longer used.
Replaced by another feature?	Yes, the ISO20022 Camt.054 bank notification import.
Product areas affected	Accounts receivable
Status	Deprecated: A removal date has not been set for this feature.

Payroll information in Human Resources

Human Resources Payroll information

Reason for deprecation/removal	This functionality has been replaced by core Payroll and Human Resources pages.
Replaced by another feature?	**Benefits**, **Earnings**, and other related pages that were previously in US Payroll have been reconfigured, and are now part of the core Human Resources configuration to help support external payroll processing. This functionality is accessed by using the **Human Resources 1** > **Payroll** configuration key.
Product areas affected	Human Resources, Payroll
Status	Removed as of Dynamics 365 for Operations version 1611.

Performance management goal workflow

Performance management includes goal management and integration with performance reviews.

Reason for deprecation/removal	Performance management was redesigned, and the number of goal pages was reduced to simplify the process.
Replaced by another feature?	No. Goals are visible to managers through the Manager Self Service portal, and can be changed and viewed by the manager.
Product areas affected	Human capital management

Status	Removed as of Dynamics 365 for Operations version 1611.

Postgirot and Postgirot Utland payment formats for Sweden

Postgirot and Postgirot Utland payment formats for Sweden.

Reason for deprecation/removal	The payment formats are no longer used.
Replaced by another feature?	Yes, ISO20022 Credit transfer payment format for Sweden
Product areas affected	Accounts payable
Status	Deprecated: A removal date has not been set for this feature.

Radio frequency identifier

Radio Frequency Identification (RFID) is a data-collection technology that uses electronic tags to store identification data and a no-line-of-sight requirement reader to capture the identification data.

Reason for deprecation/removal	Low customer usage and a limited feature set.
Replaced by another feature?	No
Product areas affected	Inventory management
Status	Removed as of Dynamics 365 for Operations 1611.

Report about state invoices numbering for Latvia

Latvian legislation provides specific rules about the numbering of sales invoices. The functionality lets you assign specific numbers to sales invoices, based on the user or user group. You can then generate a report or an XML file. You can also print a report about invoice numbers that are used.

Reason for deprecation/removal	The state invoice numbering no longer has to be maintained. The report about used invoice numbers is no longer required.
Replaced by another feature?	No
Product areas affected	Accounts receivable
Status	Deprecated: A removal date has not been set for this feature.

Set up the names of the manager and general accountant of a company for Lithuania

The names of the manager and the general accountant of a company can be specified in the company

information and used in different local report printouts.

Reason for deprecation/removal	Replaced by another feature
Replaced by another feature?	Yes, the setup of officials can be used for the same purpose.
Product areas affected	Accounts payable, Accounts receivable, Cash and bank management
Status	Deprecated: A removal date has not been set for this feature.

Shipping carrier interface

Reason for deprecation/removal	Duplicate functionality
Replaced by another feature?	Partially replaced by Transportation management
Product areas affected	Sales and marketing, Inventory management
Status	Removed as of Dynamics 365 for Operations version 1611.

Telepay payment formats for Norway

Telepay payment formats include vendor payment export (credit transfer) and customer payment collection (direct debit).

Reason for deprecation/removal	The payment formats are no longer used.
Replaced by another feature?	Yes, ISO20022 Credit transfer payment format and AvtaleGiro customer payment format for Norway, as well as pain.002 and camt.054 bank notification return files import.
Product areas affected	Accounts payable, Accounts receivable
Status	Deprecated: A removal date has not been set for this feature.

Vendor payment export formats for Finland

Two formats for exporting payments are available for Finland. LM02 (FI) is used for domestic payments, and LUM2 (FI) is used for foreign payments.

Reason for deprecation/removal	The payment formats are no longer used.
Replaced by another feature?	Yes, ISO20022 Credit transfer payment format for Finland

Product areas affected	Accounts payable
Status	Deprecated: A removal date has not been set for this feature.

Warehouse management II

Reason for deprecation/removal	The Warehouse management II solution (WMS II) that was available in the **Inventory management** module duplicates functionality that is in the **Warehouse management** module that was released in Dynamics AX 2012 R3.
Replaced by another feature?	The **Warehouse management** module that was released in AX 2012 R3, Dynamics AX 2012 R3 CU8, and Dynamics AX 2012 R3 CU9 replaces the Warehouse management II features. The new module has more advanced features and more flexible warehouse management processes than Warehouse management II.
Product areas affected	Inventory management, Sales and marketing, Procurement and sourcing
Status	Removed as of Dynamics 365 for Operations version 1611.

Worker reminders in Human Resources

Human Resources Payroll information

Reason for deprecation/removal	Low usage
Replaced by another feature?	No
Product areas affected	Human resources
Status	Removed as of Dynamics 365 for Operations version 1611

Workflow for creating goals

A workflow for managing the creation of employee goals is one of several workflows that were available to help coordinate the performance management process.

Reason for deprecation/removal	Performance management has been completely redesigned in Finance and Operations.

Replaced by another feature?	The redesigned Performance management feature gives more control over the content of the goals, the measurements that are used to track progress, and the attachment of supporting documentation. Goals can be stored as templates and then reused. This feature can help you set up additional goals for your employees more quickly.
Product areas affected	Human capital management
Status	Removed as of Dynamics 365 for Operations version 1611.

Dynamics AX 7.0

Ability to cancel changes to a vendor invoice

Reason for deprecation/removal	Performance enhancement
Replaced by another feature?	No
Product areas affected	Accounts payable
Status	Removed as of Dynamics AX 7.0.

AIF, AxD, and AxBC integrations

In Application Integration Framework (AIF), data can be exchanged with external systems through business logic that is exposed as services. Dynamics AX includes services that are based on documents and .NET Business Connector (AxBC). A document is created by using XML. The XML includes header information that is added to create a *message* that can be transferred into or out of Dynamics AX. Examples of documents include sales orders and purchase orders. However, almost any entity, such as a customer, can be represented by a document. Services that are based on documents use the **Axd <Document>** classes.

Reason for deprecation/removal	The architecture of AIF and AxDs could not be scaled to a cloud service. There were performance issues around bulk import.
Replaced by another feature?	This feature is replaced by the Data Import/Export framework, which supports recurring bulk import/export. For AxBC, we recommend that you use the actual tables.
Product areas affected	AxDs, AxBCs, and AIF
Status	Removed as of Dynamics AX 7.0.

Billing code rate scripts

Billing scripts were used to calculate billing rates for billing codes. This scripts required custom development in the C Sharp or Visual Basic programming language. In the current version of Dynamics AX, the **billing code rate scripts** are not supported.

Reason for deprecation/removal	The support for the custom C Sharp or Visual Basic scripts was not added in Dynamics AX 7.0.
Replaced by another feature?	No
Product areas affected	Public sector, Accounts receivable
Status	Removed as of Dynamics AX 7.0.

BOMs without BOM versions

When the **BOM versions** configuration key was disabled, bill of materials (BOM) versions were hidden in all forms, and the system forced a 1:1 relationship between released products and BOMs. In the current version of Dynamics AX, the **BOM versions** configuration key can't be disabled.

Reason for deprecation/removal	Using a configuration key to control BOM versions doesn't scale in a cloud environment.
Replaced by another feature?	No
Product areas affected	Product information management, Inventory management
Status	Removed as of Dynamics AX 7.0.

Brazilian Bordero

Specific method of payment for Brazilian companies

Reason for deprecation/removal	Support for the Brazilian Bordero method of payment has been discontinued from Brazilian localization
Replaced by another feature?	No
Product areas affected	Accounts payable
Status	Deprecated: A removal date has not been set for this feature.

Brazilian Sintegra statement

Federal tax statement for ICMS tax

Reason for deprecation/removal	This statement is no longer applicable in some Brazilian states.
Replaced by another feature?	No. Users can use Generic Electronic reporting tool to configure the statement if required under specific situations.

Product areas affected	Fiscal books
Status	Deprecated: A removal date has not been set for this feature.

Brazilian SCAN contingency mode for NF-e

(SCAN) contingency environment is used to generate, export, and import the status of a Nota Fiscal eletrônica (NF-e) when the environment of Secretaria da Fazenda (SEFAZ) is not available.

Reason for deprecation/removal	This method of contingency is no longer applicable in all Brazilian states
Replaced by another feature?	No
Product areas affected	Accounts receivable
Status	Deprecated: A removal date has not been set for this feature.

Business Analyzer

This mobile application let users review key business metrics.

Reason for deprecation/removal	This functionality has been replaced by another feature.
Replaced by another feature?	The Monitor financial performance content pack for Microsoft Power BI will include key financial metrics that were previously available in Business Analyzer.
Product areas affected	General ledger
Status	Deprecated: The use of Business Analyzer has been deprecated.

Business statistics

The setup of business statistics inquiries that can help you analyze the performance of the organization

Reason for deprecation/removal	Legacy approach to business intelligence (BI), low customer usage, and a limited feature set
Replaced by another feature?	New BI solutions for the current version of Dynamics AX
Product areas affected	Procurement and sourcing, Accounts payable, Sales and marketing, Accounts receivable
Status	Removed as of Dynamics AX 7.0.

Change document date function in Invoice approval journal

Reason for deprecation/removal	Low usage
Replaced by another feature?	Yes. The document date on the posted vendor transaction can be changed.
Product areas affected	Accounts payable
Status	Removed as of Dynamics AX 7.0.

ClieOp03 payment format for the Netherlands

Reason for deprecation/removal	The format is no longer applicable in the Netherlands, because it has been replaced by SEPA functionality.
Replaced by another feature?	SEPA payments export
Product areas affected	All modules
Status	Deprecated: A removal date has not been set for this feature.

Compliance Center

The Compliance Center was an Enterprise Portal site for managing the documentation requirements for compliance initiatives that are related to the Sarbanes-Oxley law.

Reason for deprecation/removal	Lack of customer usage. Microsoft SharePoint includes the same capability that was available in the Compliance Center.
Replaced by another feature?	No
Product areas affected	Compliance and internal controls
Status	Removed as of Dynamics AX 7.0.

Connector for Microsoft Dynamics

This tool was used to integrate key data from Microsoft Dynamics CRM to Dynamics ERP applications.

Reason for deprecation/removal	This functionality has been replaced by another feature.
Replaced by another feature?	Dataverse
Product areas affected	Connector for Dynamics

Status	Removed as of Dynamics AX 7.0.

Container unit and multi dimension on-hand

Reason for deprecation/removal	Duplicate functionality
Replaced by another feature?	Yes. Since AX 2012, this functionality has been replaced by the consolidated batch orders feature set. This feature set includes the consolidated on-hand view.
Product areas affected	Product information management, Production control, Inventory management, Sales and marketing
Status	Removed as of Dynamics AX 7.0.

Cue group metadata

Reason for deprecation/removal	Cue groups were used to display one or more Cues in the FactBox area. There was limited uptake, and there were also performance concerns, because a record change in a parent form caused one query per Cue in the Cue group.
Replaced by another feature?	No
Product areas affected	All modules
Status	Removed as of Dynamics AX 7.0.

Cue metadata

Reason for deprecation/removal	Cue metadata was limited to count or sum information.
Replaced by another feature?	Tile metadata was introduced to provide more flexibility for modeling. For example, you can model current counts, navigation, and key performance indicators (KPIs). Count tile metadata is the direct replacement of the Cue metadata.
Product areas affected	All modules
Status	Removed as of Dynamics AX 7.0

Danish check format

Reason for deprecation/removal	Support for the Danish check format layout has been discontinued, and the report has been removed from DK localization.
Replaced by another feature?	No
Product areas affected	All modules
Status	Deprecated: A removal date has not been set for this feature.

Data partitions

Data partitions provide a logical separation of data in the Dynamics AX database.

Reason for deprecation/removal	Data partitions were introduced in Dynamics AX 2012 R2 to enable data isolation. In a common scenario, a company has subsidiaries, and the data from one subsidiary should not be visible to another subsidiary, even though both subsidiaries are managed by the same IT department. However, extra scripts and management overhead throughout the program were required in order to create new partitions and populate them with data, and to back up partition data. In the cloud, where we have access to platform as a service (PaaS) database services (Microsoft Azure SQL Database), it's much more efficient to use a database as the isolation container than to do isolation in the program. Regardless of whether data partitioning is required for subsidiaries, for multiple tenants, or just for scale, we believe that the scenarios can be handled better through multiple instances of Finance and Operations.
Replaced by another feature?	Customers using data partitions must use multiple instances of Finance and Operations if database level separation is a critical issue.
Product areas affected	All modules
Status	Removed as of Dynamics AX 7.0.

Database and file share storage for attachments

Dynamics AX 2012 allowed storage of attachments in the database and in file shares. Both of those options are no longer supported.

Reason for deprecation/removal	Files share storage is no longer supported because cloud-hosted environments cannot communicate with local file shares. Database storage has been deprecated in favor of Azure Blob storage. Azure Blob storage is equivalent to storage in the database, as documents can only be accessed through Finance and Operations client forms. This provides the added benefit of providing storage that doesn't negatively affect the performance of the database. Blob storage is the default storage mechanism for Document Management and works immediately.
Replaced by another feature?	Database storage has been deprecated in favor of Azure Blob storage.
Product areas affected	All modules
Status	Removed as of Dynamics AX 7.0.

Delimitation

Reason for deprecation/removal	No use of the functionality was found.
Replaced by another feature?	No
Product areas affected	Time and attendance
Status	Removed as of Dynamics AX 7.0.

Desktop client

Reason for deprecation/removal	The Dynamics AX client experience has been redesigned to improve usability across multiple platforms and devices.
Replaced by another feature?	The new web client is based on the desktop Form metadata and programming model that have been modified to provide a rich web platform.
Product areas affected	All modules
Status	Removed as of Dynamics AX 7.0.

Direct database connection

In Dynamics AX 2012 R3, Retail Modern POS could connect directly to the Channel DB in similar fashion to Enterprise POS. This was in addition to the standard communication method of Retail Modern POS communicating through Retail Server.

Reason for deprecation/removal	Direct database connectivity required lower security protocols and was primarily used to achieve the highest levels of performance. Due to the performance and security enhancements that have occurred in Finance and Operations, this functionality now causes more issues than it solves.
Replaced by another feature?	No. Only standard Retail Server communication is now supported.
Product areas affected	Channel DB/Retail Modern POS
Status	Removed as of Dynamics AX 7.0.

Dutch SWIFT MT940

Reason for deprecation/removal	Generic functionality is now used instead of localized functionality.
Replaced by another feature?	Yes, this functionality has been replaced by Advanced bank reconciliation functionality.
Product areas affected	All modules
Status	Deprecated: A removal date has not been set for this feature.

eBilanz (XBRL for Germany)

This functionality provided eXtensible Business Reporting Language (XBRL) output that is intended specifically for the German eBilanz taxonomy.

Reason for deprecation/removal	Lack of customer usage
Replaced by another feature?	This feature hasn't been replaced by another feature, but multiple specialized XBRL packages that provide rich XBRL functionality are available for the German market.
Product areas affected	Management Reporter
Status	Deprecated: A removal date has not been set for this feature.

Enterprise Portal client

Reason for deprecation/removal	A single client platform has been provided.

Replaced by another feature?	The new web client is based on the desktop form metadata and programming model that have been modified to provide a rich web platform.
Product areas affected	All modules
Status	Removed as of Dynamics AX 7.0.

Environmental sustainability

Reason for deprecation/removal	Low customer usage and a limited feature set
Replaced by another feature?	No
Product areas affected	Compliance and internal controls, Accounts payable
Status	Removed as of Dynamics AX 7.0.

Form ActiveX and Managed Host controls

Reason for deprecation/removal	The ActiveX and Managed Host controls are based on the deprecated desktop client.
Replaced by another feature?	The extensible control framework supports building new controls that are based on HTML, CSS, and JavaScript, and is a first-class control in the Microsoft Visual Studio Tooling environment.
Product areas affected	All modules
Status	Removed as of Dynamics AX 7.0.

Generate prenotes by using a batch

Prenote generation can't be done by using a batch, but it can still be done by a user.

Reason for deprecation/removal	No form exists to persist and display the resulting prenote file when it's generated by using a batch.
Replaced by another feature?	Prenotes can still be generated, and the user has control over the location where the file is saved.
Product areas affected	Accounts payable, Accounts receivable, Cash and bank management
Status	Removed as of AX 7.0.

German DTAUS payment export and account statement import (totals and transactions)

Reason for deprecation/removal	The format is no longer applicable in Germany, because it has been replaced by Single Euro Payments Area (SEPA) functionality.
Replaced by another feature?	Yes, this functionality has been replaced by SEPA payment export and advanced bank reconciliation functionality for importing account statements.
Product areas affected	All modules
Status	Deprecated: A removal date has not been set for this feature.

German DTAZV payment format in domestic Currency

Reason for deprecation/removal	The format is no longer applicable in Germany, because it has been replaced by SEPA functionality.
Replaced by another feature?	SEPA payments export
Product areas affected	Accounts payable
Status	Deprecated: A removal date has not been set for this feature.

German MT940 import

Reason for deprecation/removal	Generic functionality is now used instead of localized functionality.
Replaced by another feature?	Yes, this functionality has been replaced by Advanced bank reconciliation functionality.
Product areas affected	All modules
Status	Deprecated: A removal date has not been set for this feature.

German XML EU Sales list

Reason for deprecation/removal	The XML format for German EU Sales List reporting is no longer supported. Only the ELMA5 text file format can be used to submit the EU Sales List report to the German Tax Office.
Replaced by another feature?	No

Product areas affected	Tax
Status	Deprecated: A removal date has not been set for this feature.

GL SSRS reports

Reports that include the following menu items have been removed: **Summary trial balance**, **Detailed trial balance**, **Chart of accounts**, **Audit trail**, **Balances**, and **Balance list**.

Reason for deprecation/removal	Financial Microsoft SQL Server Reporting Services (SSRS) reports have been replaced by Management Reporter capabilities and default reports.
Replaced by another feature?	Management Reporter (labeled **Financial reporting** in the current version of Dynamics AX)
Product areas affected	General ledger
Status	Removed as of Dynamics AX 7.0.

InfoPart and FormPart metadata

Reason for deprecation/removal	InfoPart and FormPart metadata enabled the creation of FactBoxes for two different clients.
Replaced by another feature?	InfoPart metadata, which was a simplified form definition, is converted into a Form by upgrade tooling. FormPart metadata, which referenced a Form, is replaced by a more direct reference that is created by upgrade tooling.
Product areas affected	All modules
Status	Removed as of Dynamics AX 7.0.

Main account list page

A list of accounts for the legal entity and related balance information

Reason for deprecation/removal	Balance information is available on the **Trial balance** list page by account and dimension.
Replaced by another feature?	**Main accounts** contains the same list of accounts that the **Main account** list page contained. The grid view in **Main accounts** also shows an even smaller, grid-like view.
Product areas affected	General ledger

Status	Removed as of Dynamics AX 7.0.

Malaysia and Singapore bank cash flow report

This feature let the user print a cash flow report that shows transactions and details of the cash inflows and outflows for a specific date range for selected bank accounts.

Reason for deprecation/removal	The same information can be obtained from the Inquiry bank transaction.
Replaced by another feature?	The Inquiry bank transaction
Product areas affected	Cash and bank management
Status	Deprecated: A removal date has not been set for this feature.

Mexican CFD electronic invoice

This feature enabled the generation of Mexican electronic invoices by using the Comprobante Fiscal Digital (CFD) method, where the company signs the invoice by requesting the related authorization from the government. This feature also provides a monthly report that includes all electronics invoices that were issued in the period.

Reason for deprecation/removal	The method is no longer applicable. The generation of electronic invoices by using the CFD method was deprecated by the tax authorities and replaced by the Comprobante Fiscal Digital a través de Internet (CFDI) method, where the signing is delegated to the third-party provider (PAC). The monthly report has been removed, and an inquiry option lets users inquire about historical transactions.
Replaced by another feature?	No
Product areas affected	Account receivables, Project
Status	Deprecated: A removal date has not been set for this feature.

Mexico realized and unrealized VAT

Dynamics AX 2012 managed unrealized value-added tax (VAT) by using Mexico-specific functionality for unrealized tax.

Reason for deprecation/removal	Duplicate functionality
Replaced by another feature?	Yes, this functionality has been replaced by standard conditional sales tax functionality that is provided by Core.

Product areas affected	Tax
Status	Deprecated: A removal date has not been set for this feature.

Microsoft Outlook integration

Reason for deprecation/removal	This functionality has been replaced by Microsoft Exchange Server integration.
Replaced by another feature?	Yes
Product areas affected	Sales and marketing
Status	Removed as of Dynamics AX 7.0.

Private blocking of inventory and warehouse management journals

The inventory and warehouse journals no longer support the ability to mark a journal as private for a selected user. Only the process of blocking journals as private for user groups and blocking during editing is supported.

Reason for deprecation/removal	No use of the functionality was found.
Replaced by another feature?	No
Product areas affected	Inventory management
Status	Removed as of Dynamics AX 7.0.

Product builder

Product builder was used to dynamically configure items from a sales order, purchase order, production order, sales quotation, project quotation, or item requirement. Based on a product model that had modeling variables, the user could select values to meet the customer requirements and get a unique product variant that had a BOM and route.

Reason for deprecation/removal	Product builder exposed X++ code to end users and isn't supported in the current version of Dynamics AX. It has been removed to avoid duplicate maintenance efforts on overlapping, sizeable codebases.
Replaced by another feature?	Yes. The constraint-based configuration was introduced in Dynamics AX 2012 where the depreciation of Product builder in future versions was already announced. The constraint-based configuration technology is selected on the product masters to enable the configuration. To learn more, see Product configuration overview.

Product areas affected	Product information management, Sales and marketing
Status	Removed as of Dynamics AX 7.0.

Production Floor app

This is the app for tablet devices running Windows 8.1 RT and Windows 8.1 Pro.

Reason for deprecation/removal	With the change to a web-based client, it is possible to deliver similar functionality through the native Dynamics AX 7.0 client. The Job Card Device provides a production floor user interface that is optimized for touch and tablet form factors.
Replaced by another feature?	Yes. The Job Card Device, which is a native part of Dynamics AX 7.0.
Product areas affected	Production control
Status	Deprecated: A removal date from the Microsoft store has not yet been set for this feature.

Rename product dimension

This feature let you change the name of one of the three standard product dimensions (size, color, or style) to a name that better suited your business requirements. Renaming included all the labels where the product dimension name was used.

Reason for deprecation/removal	The current version of Dynamics AX doesn't support label changes at run time.
Replaced by another feature?	No
Product areas affected	Product information management
Status	Removed as of Dynamics AX 7.0.

Retail Server connectivity using HTTP

In Dynamics AX 2012 R3, the Retail Server could function using HTTP communication (non-secured). This was in addition to the standard communication using HTTPS.

Reason for deprecation/removal	Due to new security requirements, only secured communication using TLS 1.2 (or above, as available) is now supported. The self-service installer will automatically configure the computer for this communication.
Replaced by another feature?	No. Only standard HTTPS communication is now supported.

Product areas affected	Retail Server
Status	Removed as of Dynamics AX 7.0.

Role Center pages

Reason for deprecation/removal	Role Center pages were built on the deprecated Enterprise Portal platform, which has been replaced by the new web client platform in the current version of Dynamics AX.
Replaced by another feature?	The new Workspace form pattern provides users with a process-centered design that provides easy access to commonly used tasks within that process.
Product areas affected	All modules
Status	Removed as of Dynamics AX 7.0

Sales tax jurisdictions

Reason for deprecation/removal	Low customer usage and a limited feature set
Replaced by another feature?	No
Product areas affected	US sales tax
Status	Removed as of Dynamics AX 7.0.

Sites Services

Sites Services let you build websites that extend your business processes to the Internet without IT support.

Reason for deprecation/removal	The Microsoft Azure infrastructure that is used by Dynamics AX has new capabilities that can be used instead (for example, Azure sites).
Replaced by another feature?	No
Product areas affected	HR recruiting, Case management, Request for quotes, Vendor registration, Collaborative workspaces for opportunities and campaigns
Status	Removed as of Dynamics AX 7.0.

SSAS demand forecasting strategy

Reason for deprecation/removal	The design of the feature cannot be supported in the new cloud architecture.
Replaced by another feature?	Azure Machine Learning demand forecasting strategy
Product areas affected	Master planning
Status	Removed as of Dynamics AX 7.0.

Vendor invoice pool excluding posting details

Reason for deprecation/removal	Low usage. This functionality has been replaced by the Invoice journal that has workflow functionality.
Replaced by another feature?	Workflow capabilities of the Invoice journal.
Product areas affected	Accounts payable
Status	Removed as of Dynamics AX 7.0.

Virtual company accounts

The virtual companies feature is no longer supported in Dynamics AX. The virtual companies feature let users set up tables that could be shared by a set of companies. For a description of the feature, see Company accounts and Virtual company accounts. The feature works by grouping tables into collections that are assigned to virtual companies, which are groups of existing "real" companies. Queries are created so that all the companies in the virtual company can access the data in the tables of the associated table collections.

Reason for deprecation/removal	- Virtual companies must be set up before data is stored in the tables. Retrofitting virtual companies onto an existing implementation is very difficult. - Because there has been so much data normalization in the current version of Dynamics AX, it has become difficult to know what to add to the table collections. For example, it's difficult to know which tables to share. All the tables referenced from tables that are in a virtual company must also added. Because of table normalization, even simple master data that is spread across multiple tables must be part of the virtual company. Any mistake that is made here will cause functional issues. - When a table is part of a virtual company, it loses information about the origin of the data, and only the virtual company is recorded.
Replaced by another feature?	Global tables can be used to make tables accessible from all companies. Currently, there is no replacement.
Product areas affected	All modules

Status	Removed as of Dynamics AX 7.0.

Windows 8 tablet app

The Windows 8 tablet app provided functionality for expense entry and approval.

Reason for deprecation/removal	Finance and Operations is compatible with tablets. The tablet app is no longer required.
Replaced by another feature?	No.
Product areas affected	Expense management
Status	Removed: This functionality is only available for Dynamics AX 2012 R3.

Workplanner

Reason for deprecation/removal	Low usage
Replaced by another feature?	No, but the **Profile relation** page, which is opened from the **Profile groups** page, supports the same business scenario as the deprecated **Workplanner** page.
Product areas affected	Time and attendance
Status	The code has not been removed. However, the form, JmgWorkPlanner, was not migrated.

X++ financial statements

Reason for deprecation/removal	This functionality has been replaced by another feature.
Replaced by another feature?	Management Reporter (labeled **Financial reporting** in the current version of Dynamics AX)
Product areas affected	General ledger
Status	Removed as of Dynamics AX 2012

Deprecation of methods and metadata elements

2/18/2021 • 2 minutes to read • Edit Online

As the Microsoft code base continues to evolve, some methods and metadata elements will no longer be required. Microsoft will mark these obsolete methods and metadata elements for deprecation.

- Methods are marked with the **SysObsolete** attribute. Typically, this attribute recommends an alternative to the method.
- For metadata elements, the **IsObsolete** property is set to **Yes**.

The deprecation is compatible with both binaries and design time. The referencing code will continue to work as expected, and no immediate action is required. During compilation, any references to deprecated artifacts are reported as compile **warnings**.

Cleanup of deprecated elements

After a period of at least 12 months, Microsoft might delete obsolete methods and metadata elements.

However, if telemetry shows that any obsolete methods or metadata elements are still used, Microsoft will **not** delete them, to reduce the risk that consumers will be broken.

Minimize your risk of being affected

Here are some tips that you, as a consumer of the Microsoft code base, can use to avoid being affected when methods and metadata elements are deprecated:

- Compile your code base at least every 12 months on top of the latest code base. If you receive any warnings because deprecated artifacts are used, address those warnings as soon as possible.
- Avoid **new** dependencies on deprecated artifacts. Microsoft might have just deleted the artifact, because there is a time window between when releases and telemetry are available.

List of deprecated methods and metadata elements

For reference, download the Microsoft Excel file, available on the Deprecation of methods and metadata elements page of CustomerSource, which shows the artifacts that have been marked for deprecation in each major release.

Deprecated APIs

2/18/2021 • 14 minutes to read • Edit Online

This document provides the list of deprecated APIs and migration guidance for some of the deprecated APIs.

Overview

A number of APIs from Dynamics AX 2012 have been identified. The reason for the deprecation for each API varies. Most commonly, the reasons are one of the following:

- Not suited/applicable to the new client.
- Degrade performance.
- Chatty (cause lot of traffic back and forth between server and client).
- Redundant (framework automatically handles these now).

Throughout this table, under the
Reason for Deprecation heading, "the client" refers to the web client.

List of deprecated APIs

OBJECT	TYPE	NAME	NOTES
ActionPane	Method	tabChanged	Updates to ActionPanes (or controls inside of ActionPanes) should be done based on the active row, not when the tab becomes active.
ActionPaneTab	Method	selectionChanged	Updates to ActionPaneTabs (or controls inside of ActionPaneTabs) should be done based on the active row, not when the tab becomes active.
Box	Method	yesNoTextMenu-LinkText	
ComboBox	Method	getEditText	**Overview** N/A **Reason for deprecation** Redundant. **Migration notes** Use getText instead.

OBJECT	TYPE	NAME	NOTES
DataSet DataSetNode DataSetRun	Class		**Overview** Used in Dynamics AX 2012 with Enterprise Portal. **Reason for deprecation** Not applicable in the client. **Migration notes** **Remove calls to these APIs from your code.**
DataSourceMethodInfo DataSourceMethodInfoList	Class		
DDEClient DDEServer DLL DLLFunction HDC HWnd Thread WinAPINative WinGDI	Class		**Overview** N/A **Reason for deprecation** Specific to Dynamics AX 2012 Windows client and not compatible with the client. **Migration notes** Remove usage of these APIs from your code.
DocumentManagement-Helper	Class		
Form	Method	addhistory currentHistoryName currentHistoryState updateHistory	**Overview** Used in Dynamics AX 2012 with address bar. **Reason for deprecation** Navigation model in the client has changed. **Migration notes** Remove calls to these APIs from your code.
Form	Method	arrange	
Form	Method	controlCallingMethod	
Form	Method	controlMethod-Overload controlMethod-OverloadObject	**Overview** Used in Dynamics AX 2012 to register override methods. **Reason for deprecation** This is not a clean and recommended way to register override methods. **Migration notes** Use registerOverrideMethod instead.
Form	Method	copy cut paste	

OBJECT	TYPE	NAME	NOTES
Form	Method	delAutoCompleteString getAutoCompleteString setAutoCompleteString	**Overview** Used in Dynamics AX 2012 to set, get, and delete automatic suggestions. **Reason for deprecation** Specific to Dynamics AX 2012 Windows client. **Migration notes** Remove calls to these APIs from your code.
Form	Method	firstField	
Form	Method	formOnTop	**Overview** Used in Dynamics AX 2012 to manage windows and navigation. **Reason for deprecation** The client has a new navigation model. **Migration notes** Remove calls to these APIs from your code.
Form	Method	hWnd installMessageProc removeMessageProc	**Overview** N/A **Reason for deprecation** Specific to Dynamics AX 2012 Windows client and not compatible with the client. **Migration notes** Remove usage of these APIs from your code.
Form	Method	isPreloadedInstance	**Overview** Used in Dynamics AX 2012 with preloading. **Reason for deprecation** Preloading is not applicable in the client. **Migration notes** Remove calls to these APIs from your code.
Form	Method	lastField nextField nextGroup prevField prevGroup	

OBJECT	TYPE	NAME	NOTES
Form	Method	Lock lockWindowUpdate unLock	**Overview** These methods were used to prevent the redrawing of windows when performing a set of UI updates. Without these the window would be redrawn in response to each individual change leading to bad end-user experience and degraded performance. **Reason for deprecation** These methods are specific to the Windows client and are no longer needed for the client. **Migration notes** A code upgrade rule has been provided to remove occurrences of these APIs. You can safely remove any calls to these APIs from your code.
Form	Method	print printPreview send	**Overview** Used in Dynamics AX 2012 to override the Auto Report generation for the form **Reason for deprecation** Microsoft 365 integration offers a better user experience in the client. The 'Export' function is available for the user in the Dynamics AX client forms. **Migration notes** Remove calls to these APIs from your code.
Form	Method	redraw resetStatusBar-BackgroundColor setStatusBar-BackgroundColor sysColorChanged	**Overview** Used to control styles or colors. **Reason for deprecation** Remove ability for developers to specify the colors via API for consistent visuals. **Migration notes** A code upgrade rule has been provided to remove occurrences of the redraw API. Remove usage of these APIs from your code.
Form	Method	reload	

OBJECT	TYPE	NAME	NOTES
Form	Method	resetSize	**Overview** This method was used when controls were added/removed from a form causing its size to change. Without it the window might not be correctly sized to account for the added/removed controls. **Reason for deprecation** These methods are specific to the Windows client and are no longer needed for the client. **Migration notes** You can safely remove any calls to these APIs from your code.
Form	Method	resize	
FormActiveXControl FormAnimateControl FormBuildActiveXControl FormBuildAnimateControl FormBuildManaged-HostControl FormBuildSegmented-EntryControl FormManagedHostControl FormSegmented-EntryControl	Class		**Overview** These were used to host or create various custom controls for Dynamics AX 2012. **Reason for deprecation** These technologies will not work with the client. **Migration notes** Application developers need to build replacement controls where needed using the control extensibility features.
FormControl	Method	beginDrag dragDrop dragLeave dragOver dragOverEx dragText drop dropEx dropFile endDrag	**Overview** Used to enable drag-and-drop scenarios in Dynamics AX 2012. **Reason for deprecation** Drag-and-drop scenarios are not supported in the client. **Migration notes** Remove usage of these APIs from your code and refactor to enable the scenarios without dependency on drag-and-drop functionality.
FormControl	Method	calcControlSize	

OBJECT	TYPE	NAME	NOTES
FormControl	Method	command processBase processForm processLink processPicture processTitle	**Overview** Was marked for deprecation in Dynamics AX 2012. **Reason for deprecation** N/A **Migration notes** Remove calls to these APIs from your code.
FormControl	Method	context showContextMenu	**Overview** This method was used when controls were added/removed from a form causing its size to change. Without it the window might not be correctly sized to account for the added/removed controls. **Reason for deprecation** These methods relied on APIs that are specific to the Windows client. **Migration notes** Use getContextMenuOptions and selectedMenuOptions instead.
FormControl	Method	copy cut paste	
FormControl	Method	dateTextChange	
FormControl	Method	editControl	
FormControl	Method	hasControl-PositionOverride	
FormControl	Method	helpField	
FormControl	Method	hWnd	**Overview** N/A **Reason for deprecation** Specific to Dynamics AX 2012 Windows client and not compatible with the client. **Migration notes** Remove usage of these APIs from your code.
FormControl	Method	inputSearch	
FormControl	Method	itemChanging	
FormControl	Method	keyDown	

OBJECT	TYPE	NAME	NOTES
FormControl	Method	labelMouseDblClick mouseDblClick	**Overview** The FormControl.labelMouseDblClick (int x, int y, int button, Boolean Ctrl, Boolean Shift) method is called when the label for a control is double-clicked. It provides the x, y co-ordinates of the mouse pointer, a Boolean to indicate which mouse button was clicked and Booleans to indicate whether the Ctrl and Shift key were pressed. The FormControl.mouseDblClick (int x, int y, int button, Boolean Ctrl, Boolean Shift) method is similar in function to the labelMouseDblClick method. The difference is that this method is called whenever there is a double-click (not just on the labels). **Reason for deprecation** The double-click action does not translate well to web-based application and touch-based scenarios. Additionally they might end up being chatty in many instances. **Migration notes** The recommended replacement for these methods is to use a button and the **clicked** event.
FormControl	Method	labelMousedown labelMouseup mouseDown mouseEnter mouseLeave mouseMove mouseUp	**Overview** Used to detect and respond to mouse events. **Reason for deprecation** These are not touchscreen friendly and not supported in the client. **Migration notes** Remove usage of these APIs from your code and refactor to enable the scenarios without dependency on mouse events.
FormControl	Method	onHScroll onVScroll	
FormControl	Method	paint	

OBJECT	TYPE	NAME	NOTES
FormControl	Method	prefColumnSize	**Overview** Used in Dynamics AX 2012 to control width and height **Reason for deprecation** Not applicable in the client. **Migration notes** Set the width and height explicitly instead.
FormControl	Method	selectionChanging	
FormControl	Method	setScrollInfo	
FormControl	Method	size	
FormControl	Method	updateWindow	
FormControl / FormDesign	Property	AcquireFocus	
FormControl / FormDesign	Property	ActiveBackCol ActiveBackColor ActiveBackColorRGB ActiveForeColor ActiveForeColorRGB AlternateRowShading BackgroundColor BackgroundColorRGB BackStyle BackStyleRGB CharacterSet ColorScheme DrawFocusRect ForegroundColor ForegroundColorRGB GridLines GridLinesStyle PromptRect	**Overview** Used to control styles or colors. **Reason for deprecation** Remove ability for developers to specify the colors via API for consistent visuals. **Migration notes** Remove usage of these APIs from your code.
FormControl / FormDesign	Property	AlignChild AlignChildren AlignControl Border BottomMargin BottomMarginMode ColumnSpace ColumnSpaceMode ColumnSpaceValue Left LeftMargin LeftMarginMode LeftMode RightMargin RightMarginMode SizeHeight SizeWidth TabAppearance TabAutoChange TabLayout TabMode TabPlacement Top TopMargin TopMarginMode TopMode VerticalSpacing VerticalSpacingMode VerticalSpacingValue	**Overview** Used to control layout. **Reason for deprecation** Remove ability for developers to control layout using this property to achieve a consistent layout. **Migration notes** **Remove usage of these APIs from your code. Use styles or CSS instead.**

OBJECT	TYPE	NAME	NOTES
FormControl / FormDesign	Property	AllowDocking AlwaysOnTop ArrangeGuide ArrangeWhen ContainerScroll-HorizontalOffset ContainerScroll-VerticalOffset IMEMode MaximizeBox MinimizeBox Mode NeededAccessLevel ProgressType Securable SecurityKey StatusBarStyle WindowResize	**Overview** N/A **Reason for deprecation** Specific to Dynamics AX 2012 Windows client, no longer needed. **Migration notes** Remove usage of these APIs from your code.
FormControl / FormDesign	Property	Bold	
FormControl / FormDesign	Property	CanScroll	
FormControl / FormDesign	Property	DisabledImage DisabledImageLocation DisabledResource	
FormControl / FormDesign	Property	DisplayTarget HyperLinkDataSource HyperLinkMenuItem SaveFilter SaveSize	**Overview** Used in Dynamics AX 2012 with Enterprise Portal **Reason for deprecation** Not applicable in the client. **Migration notes** Remove calls to these APIs from your code.
FormControl / FormDesign	Property	Font	
FormControl / FormDesign	Property	FontSize	
FormControl / FormDesign	Property	Frame FramePosition	**Overview** N/A **Reason for deprecation** Remove ability for developers to control frames via metadata. **Migration notes** Remove usage of these APIs from your code.
FormControl / FormDesign	Property	HideToolbar HorizontalScrollBarVisible Scrollbars VerticalScrollBarVisible	**Overview** N/A. **Reason for deprecation** Remove ability for developers to control scrollbars via metadata. **Migration notes** Remove usage of these APIs from your code.
FormControl / FormDesign	Property	ImageMode	
FormControl / FormDesign	Property	ImageName	

OBJECT	TYPE	NAME	NOTES
FormControl / FormDesign	Property	ImageResource	
FormControl / FormDesign	Property	Italic	
FormControl / FormDesign	Property	LabelAlignment	
FormControl / FormDesign	Property	LabelBold	
FormControl / FormDesign	Property	LabelCharacterSet	
FormControl / FormDesign	Property	LabelFont	
FormControl / FormDesign	Property	LabelFontSize	
FormControl / FormDesign	Property	LabelForegroundColor LabelForegroundColorRGB	
FormControl / FormDesign	Property	LabelGuide	
FormControl / FormDesign	Property	LabelHeight LabelHeightMode LabelHeightValue	
FormControl / FormDesign	Property	LabelItalic	
FormControl / FormDesign	Property	LabelUnderline	
FormControl / FormDesign	Property	LabelWidth LabelWidthMode LabelWidthValue	
FormControl / FormDesign	Property	Location	
FormControl / FormDesign	Property	NormalResource	
FormControl / FormDesign	Property	ParentPage	
FormControl / FormDesign	Property	SearchAfterInput SearchMode	
FormControl / FormDesign	Property	SelectControl	
FormControl / FormDesign	Property	SendExternalContext	
FormControl / FormDesign	Property	ShortKey	
FormControl / FormDesign	Property	Underline	
FormDataRow	Class		

OBJECT	TYPE	NAME	NOTES
FormDataSource	Property	autoNotify	**Overview** Was marked for deprecation in Dynamics AX 2012. **Reason for deprecation** N/A **Migration notes** Remove usage from your code.
FormDataSource	Method	cacheOnlyMode	
FormDataSource	Method	cacheRemoveRecord	
FormDataSource	Method	defaultMark	

OBJECT	TYPE	NAME	NOTES
FormDataSource	Method	findRecord findValue	**Usage** The FormDataSource.findRecord (Common record) method finds a specific record in the data source and makes it the current record. The FormDataSource.findValue(FieldId field, str value) method find a specific value in a specific field in the data source and makes the corresponding record the current record. It uses the FormDataSource.findRecord method for this. **Reason for deprecation** These methods use linear searching and load a large number of records in memory and negatively impact performance. **Migration notes** Replace with new APIs. Replace findRecord with positionToRecord and findValue with positionToRecordByValue. New APIs do not work in some cases, most notably with Temp tables and Views. The framework will throw an exception in those cases. If replacing with new APIs is not possible, recommended replacement is to call **element.args(). lookupRecord(recordToFind)** Followed by **FormDataSource.research(false);** FormDataSource is the data source that contains the record you want to find. Passing in a “false” argument to research causes it to not retain the current position since we want to change the position to the record we found using args.lookupRecord avoids resetting sort order, ranges, etc.
FormDataSource	Method	getDataRow	
FormDataSource	Method	markAllLoadedRecords	

OBJECT	TYPE	NAME	NOTES
FormDataSource	Method	maxPagingRowCountValue pagingEnabled startRowIndex setPagingParameters totalNumberOfRows	**Overview** Used in Dynamics AX 2012 with Enterprise Portal **Reason for deprecation** Not applicable in the client. **Migration notes** Remove calls to these APIs from your code.
FormDataSource	Method	print	
FormDesign	Method	cssClass localWebMenu showWebHelp supportReload	**Overview** Used in Dynamics AX 2012 with Enterprise Portal **Reason for deprecation** Not applicable in the client. **Migration notes** Remove calls to these APIs from your code.
FormObjectSetNotify	Method	onPaging- ParametersChanged	
FormObjectSetPaging- ParamsChangedEvtArgs	Class		
Global xInfo	Method	endLengthyOperation startLengthyOperation	**Overview** These methods were used to show/stop showing a progress indicator during long running operations. **Reason for deprecation** In the client, the system automatically takes care of showing/hiding the progress indicator and calls to these APIs are not needed. **Migration notes** You can safely remove any calls to these APIs from your code.
Image	Method	captureScreen captureWindow clipboardCopy clipboardPaste crop displayImage displayOrign exportBitmap flip getImageDimensionUnits getPixel height imageInfo imageSpotlight promoteColor reduceColorOctree resize rotate saveImage saveType transparent width	

OBJECT	TYPE	NAME	NOTES
ListPage Page	Method	activeActionPane-TabNames	**Overview** This method was used to find the active action pane tab. **Reason for deprecation** In the client, Action Pane tabs are handled client-side only, the server is not aware of the state. **Migration notes** **Remove usage of this API from your code.**
MessageWin	Class		
Object	Method	notify notifyAll wait	**Overview** Used to block and wait for an interaction/operation and notify to unblock. **Reason for deprecation** These calls are deprecated for all objects except formRun and it's derivatives. **Migration notes** Calls to these APIs from formRun or it's derivatives are allowed. Calls to these APIs from any other object should be removed.
Object	Method	objectOnServer	**Overview** Used to determine whether an object is on the server. **Reason for deprecation** This is redundant and no longer required because all objects are on the server. **Migration notes** You can safely remove calls to these APIs from you code. It will always evaluate to true.

OBJECT	TYPE	NAME	NOTES
Object	Method	setTimeOut	**Overview** This method existed on Object, but was non-functional. The implementation on FormRun was used as a timer to delay the execution of a piece of logic. **Reason for deprecation** The browser based client no longer supported this implementation. **Migration notes** Use the new setTimeOutEx method on the FormRun instead. Note that the setTimeOutEx method expects the callback to accept a parameter of type AsyncTaskResult, example: myCallBack(AsyncTaskResult result).
PopupMenu	Class		**Overview** Used in Dynamics AX 2012 to get splitters that let users change the size of the two parts that are split. **Reason for deprecation** Relied on APIs that are specific to the Dynamics AX 2012 Windows Client and cannot be used with the client. **Migration notes** Use ContextMenu instead.
SysExcel	Class		**Overview** The SysExcel classes used COM to create and edit Excel workbooks. **Reason for deprecation** SysExcel relied on calls to Excel COM objects from the client. Those COM objects are not on the server and COM calls are highly discouraged going forward. **Migration notes** Use the OpenXML .NET framework APIs instead. We are investigating the creation of an assembly that wraps OpenXML to make it easier to call from X++.

OBJECT	TYPE	NAME	NOTES
SysINetMai SysMailer SmmOutlook	Class		**Overview** These email related classes used predominantly client-side technologies that are no longer available and/or are highly discouraged. **Reason for deprecation** The SysINetMail class is being deprecated because it used client-side MAPI. The SysMailer class is being deprecated because it used CDO (a variant of OLE messaging). The classes beginning with SmmOutlook are being deprecated since they use Outlook COM objects. **Migration notes** Sending email via SMTP using the SysMailerNet class will be supported going forward. We are also actively working on client-side interactive email capabilities.
SysFormSplitter	Class		**Overview** Used in Dynamics AX 2012 to get splitters that let users change the size of the two parts that are split. **Reason for deprecation** No longer needed in the client. **Migration notes** Controls automatically provide the functionality. You can safely remove any calls to these APIs from your code. A code upgrade rule may be created in the future to automatically remove the usage.
SysListPageHelper	Class		
SysSetupFormRun	Class		**Overview** Used to by classes to indirectly extend FormRun. **Reason for deprecation** Has been merged with FormRun class. **Migration notes** Use the FormRun class instead.
TextBuffer	Method	fromFile	Use the .NET StreamReader class instead.

OBJECT	TYPE	NAME	NOTES
TextBuffer	Method	toFile	Use the .NET StreamWriter class instead.
Thread	Class		**Overview** N/A **Reason for deprecation** Specific to Dynamics AX 2012 Windows client and not compatible with the client. **Migration notes** Consider replacing with the new runAsync method or remove usage of these APIs from your code.
WinAPI	Class		**Overview** N/A **Reason for deprecation** Specific to Dynamics AX 2012 Windows client and not compatible with the client. **Migration notes** Remove usage of these APIs from your code. Replace file access APIs, such as WinAPI::getTempPath, WinAPI::fileExists, with the new file APIs.
WinAPIServer	Method	cryptProtectData cryptUnprotectData	**Overview** The WinAPIServer::cryptProtectData(CryptoBlob _unEncryptedDataBlob) and WinAPIServer::cryptUnProtectData(CryptoBlob _encryptedDataBlob) methods were used to encrypt and decrypt sensitive data. **Reason for deprecation** These methods are best suited to desktop usage and not recommended for web-based application usage. They also have a negative impact on performance. **Migration notes** Use the .NET framework APIs and well-known hashing/security algorithms instead.

OBJECT	TYPE	NAME	NOTES
xApplication	Method	runAsync	**Overview** In Dynamics AX 2012 the xApplication::runAsync method was used to make asynchronous calls to methods. **Reason for deprecation** Replaced with methods better suited to the client. **Migration notes** Use runAsync methods on the Global or FormRun classes instead. These new versions of runAsync enable the caller to make an async call to a static X++ class method. They leverage the .NET System.Threading.Tasks library to execute an async method in X++. The use of the System.Threading.Tasks.Task type allows the developer to take advantage of the rich set of features available in .NET.
xGlobal	Method	clientKind	**Overview** Most commonly used to detect presence of client, such as an interactive session. **Reason for deprecation** Replaced with a method better suited to the client. **Migration notes** Use global::hasGUI method instead.
xGlobal	Method	computerName	
xGlobal	Method	forceFormPreload	**Overview** Used in Dynamics AX 2012 with preloading. **Reason for deprecation** Preloading is not applicable in the client. **Migration notes** Remove calls to these APIs from your code.
xGlobal	Method	terminalServer	
xInfo	Method	directory	
xInfo	Method	navPane	

OBJECT	TYPE	NAME	NOTES
XmlDocument	Method	LoadSave	
XmlWriter	Method	CreateNewFile	
XppCompiler	Class		

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

User interface development home page

2/18/2021 • 3 minutes to read • Edit Online

This topic contains links to topics about developing user interface elements.

The user interface for Finance and Operation applications differs significantly from the interface for Microsoft Dynamics AX 2012. The client in Dynamics AX 2012 is a Microsoft Win32 application that has extensions that use ActiveX, WinForm, or WPF controls. The X++ application logic runs on the client for the form and table methods, and some logic occurs on the server. For controls, both the X++ logic application programming interface (API) and the physical Win32 control are tightly connected on the client. The client is an HTML web client that runs in all major browsers. These browsers include Microsoft Edge, Internet Explorer 11, Chrome, and Safari (see System requirements). The move to a web client has produced the following changes to client forms and controls:

- The physical presentation of forms and controls is now HTML, JavaScript, and CSS within the browser.
- Form controls are split into logical and physical parts. The X++ logical API and related state run on the server.
- The logical and physical parts are kept in sync through service calls that communicate changes from each side. For example, a user action on the client creates a service call to the server that is either sent immediately or queued so that it can be sent later.
- The server tier keeps the form state in memory while the form is open.

The form metamodel continues to be used to define controls and application logic. This approach supports almost all the existing Form, Form DataSource, and Form Control metamodel and X++ override methods. However, some control types, properties, and override methods have been removed, either because of incompatibility with the new platform or for performance reasons. For example, ActiveX and ManagedHost controls can no longer be used to add custom controls, because they are incompatible with the HTML platform. Instead, a new extensible control framework has been added that lets you add additional controls.

Tutorials

- Build the Rental Charge Type form
- Build the customer form

Forms

- Navigation concepts
- Page layout in the web client
- Dynamics Symbol font
- Test forms that use custom patterns

Controls

- Action controls
- Input controls and grid column sizes
- Check box support in tree controls
- Filtering options
- Display pages side-by-side using the Open in New Window icon
- Code migration - Context menu code
- Code migration - Mouse double-click logic

- Contextual data entry for lookups
- HierarchyViewer control
- Lookup controls
- File upload control
- System-defined buttons
- Images on a page or in a grid
- Font and background colors for input, table, and grid controls
- Right-to-left language support and bidirectional text
- Create icons for workspace tiles
- Keyboard shortcuts for extensible controls
- Extensible controls – public JavaScript APIs

Messaging

- Slider and MessageBox
- Messaging API: Message center, Message bar, Message details
- Messaging the user

Form pattern guidelines

- Selecting a form pattern
- Form styles and patterns
- Form pattern add-ins

FORM PATTERNS	SUPPORT FORM PATTERNS	SUB PATTERNS
Details Master Details Transaction Form Part Section List List Page Simple Details Simple List Simple List and Details Table Of Contents Task Single Task Double Wizard Workspace General Form Guidelines	Advanced Selection Dialog Drop Dialog Lookup Factbox	Custom Filter Group Dimension Entry Control Dimension Expression Builder Fields and Field Groups Filters and Toolbar Fill Text Horizontal Fields and Buttons Group Image Preview List Panel Nested Simple List and Details Section Chart Section Power BI Section Related Links Section Stacked Chart Section Tabbed List Section Tiles Tabular Fields Toolbar and List Toolbar and Fields Workspace Filter Group

Control extensibility

- Building an extensible control
- Extensible control programming reference
- Control extensibility
- Create localizable labels
- Extensible control layout guidelines

- Control the text that Task Recorder generates for a control

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Build the Rental Charge Type form

2/18/2021 • 5 minutes to read • Edit Online

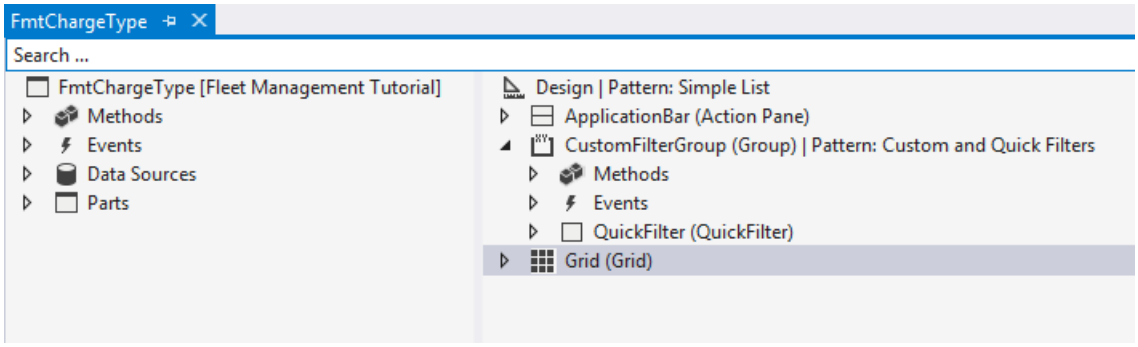
In this lab you'll create a Simple List form. A Simple List form can show reference or secondary data that has six or fewer fields. For example, the form that you create will list and describe the types of rental charges.

Prerequisites

For this tutorial, you'll need to access the environment using Remote Desktop, and be provisioned as an administrator on the instance. For more information, see Access Instances.

Overview

To create the form, you'll start from the existing form, **FmtChargeType**. This form uses the Simple List pattern. The following illustration shows the **FmtChargeType** form with the required controls from the Simple List pattern.



Adhering to the form pattern ensures that this Simple List form has the same structure and layout as other Simple List forms.

Key concepts

- Create a Simple List form using a pattern.
- Bind a table to the form.
- Add controls to the form.
- View the form using Visual Studio and a browser.

Setup

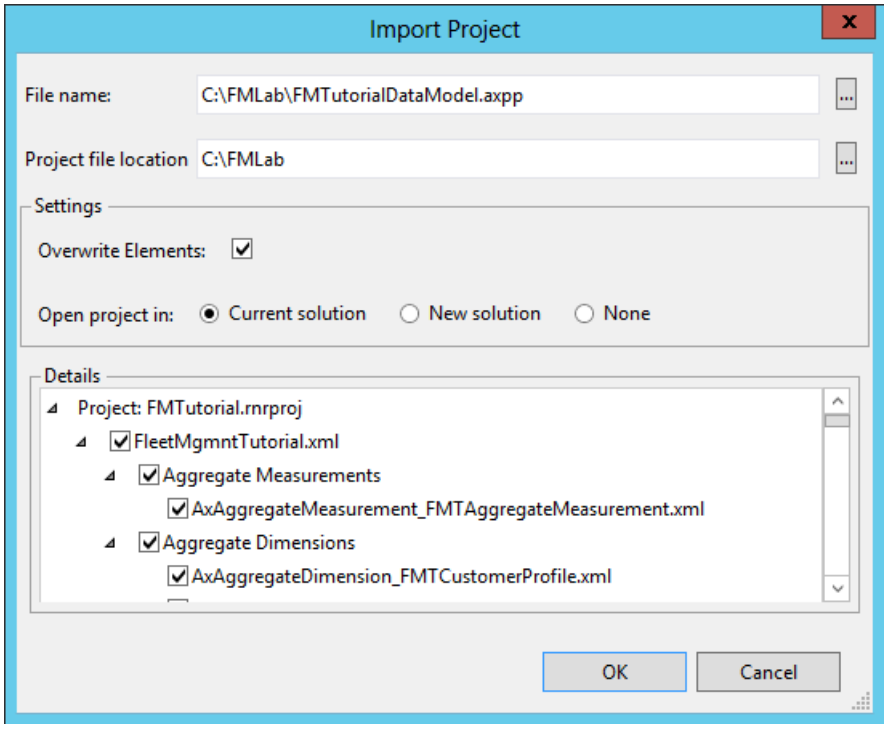
Import the tutorial project and transactional data

Use Visual Studio to import the tutorial project. The tutorial project includes the artifacts that you'll use to complete this tutorial. Use Visual Studio to open the FMTutorial project and load the data for the tutorial. You'll use the FMTDataHelper class to load data for the Fleet Management tutorial. If this is the first tutorial you're working on, review Access Microsoft Instances and make sure you provision your administrator user if you're working on a local VM.

1. Download the Fleet Management sample from https://github.com/Microsoft/FMLab, save it to **C:**, and unzip it.

2. On the desktop, double-click the Visual Studio shortcut to open the development environment.

3. On the **Finance and Operations** menu, click **Import Project**.
4. In the **Import Project** window, next to the **Filename** text box, click the ellipsis button.
5. In the **Select the file to import** window, browse to C:\FMLab, click FMTutorialDataModel.axpp, and then click **Open**.
6. In the **Project file location** text box, enter C:\FMLab.
7. Select the **Overwrite Elements** option, and the **Current solution** radio button. The following illustration shows the completed **Import Project** dialog box.



8. Click **OK**.
9. In **Solution Explorer**, expand **Classes, and** under the **FMTutorial** project, right-click **FMTDataHelper**, and then click **Set as Startup Object**.
10. On the **Build** menu, click **Rebuild Solution**. Use the rebuild to make sure that all of the files in the project are built regardless of timestamps. You can view the build progress in the **Output** window.
11. After the build completes, press **Ctrl+F5** to run the project. The browser will open and run the class that imports the data.

Open the FMTutorial project

Use Visual Studio to open the FMTutorial project. If you have Visual Studio open and have already loaded the FMTutorial project, you can continue to the next section.

1. If the development environment isn't already open, on the desktop, double-click the Visual Studio shortcut to open the development environment.
2. On the **File** menu, click **Open** > **Project/Solution**.
3. In the **Open Project** dialog box, browse to C:\FmLab\FMTutorial, select the **FMTutorial** solution, and then click **Open**
4. The FMTutorial project appears in **Solution Explorer**.

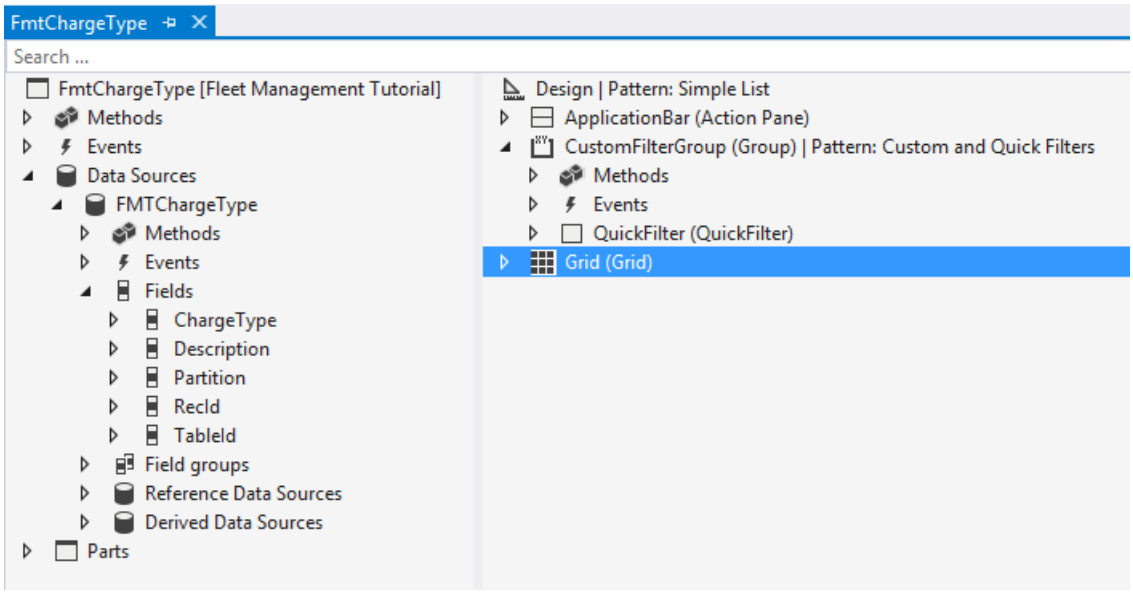
Use a template to create the form

Use Visual Studio to create the **FmtChargeType** form. You'll use a template for building the Simple List form. You'll also add a data source to the form and add fields to the data grid.

1. In **Solution Explorer**, right-click the **FMTutorial** project, point to **Add**, and then click **Existing Item**.
2. In the **Add Existing Item** window, browse to C:\FmLab, click **AxForm_FmtChargeType**, and then click **Add**. The **FmtChargeType** form appears at the bottom of the **FMTutorial** project in **Solution Explorer**.
3. In **Solution Explorer**, double-click **FmtChargeType**. The form opens in the Form designer.
4. Add the **FmtChargeType** table as the data source for the form. Right-click **Data Sources,** and then click **New Data Source**. A data source node is added.
5. Click the data source node from the previous step. In the **Properties** window, populate the following properties with the specified values.

PROPERTY	VALUE
Table	FMTChargeType
Name	FMTChargeType *Be sure to specify the value for the Table property first. This property will automatically update to use that same value.*

The following illustration shows **Data Sources** after you add the **FMTChargeType** table.



6. In the Form designer, click **Design**. In the **Properties** window, populate the following properties with the specified values.

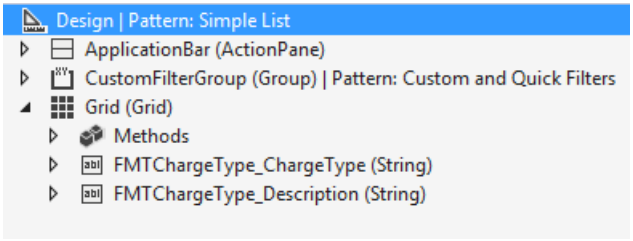
PROPERTY	VALUE
Caption	Rental charge types *This is the label that appears at the top of the form.*
Data Source	FMTChargeType *Use this property to specify the data source for the form.*

7. In the Form designer, click **Design** > **Grid**.
8. Bind the FMTChargeType data source to the grid that appears in the simple list form. In the **Properties** window, in **Data Source**, enter **FMTChargeType**.

9. You have to specify the data source before you add fields to the grid. You can then use the fields from the data source to add columns to the grid. Add two fields from the data source to the grid. The fields you add will appear as columns on the Simple List form. Expand the FMTChargeType data source **Fields** node in the left pane. Press **Ctrl** and then click the following fields:

    - ChargeType
    - Description

10. Drag the selected fields to **Design** > **Grid** in the right pane. The following illustration shows the grid after the grid node is expanded and the two fields are added.



11. In the Form designer, click **Design** > **CustomFilterGroup** > **QuickFilter**.

12. In the **Properties** window, click **TargetControl**, and then select **Grid** to bind the **QuickFilter** control to the grid on the form.

13. Click **File** > **Save FmtChargeType**.

View the form

Use Visual Studio to build and run the **FmtChargeType** form.

1. In **Solution Explorer**, right-click the **FmtChargeType** form, and then click **Set as Startup Object**.

2. Press Ctrl+F5 to build and run the form.

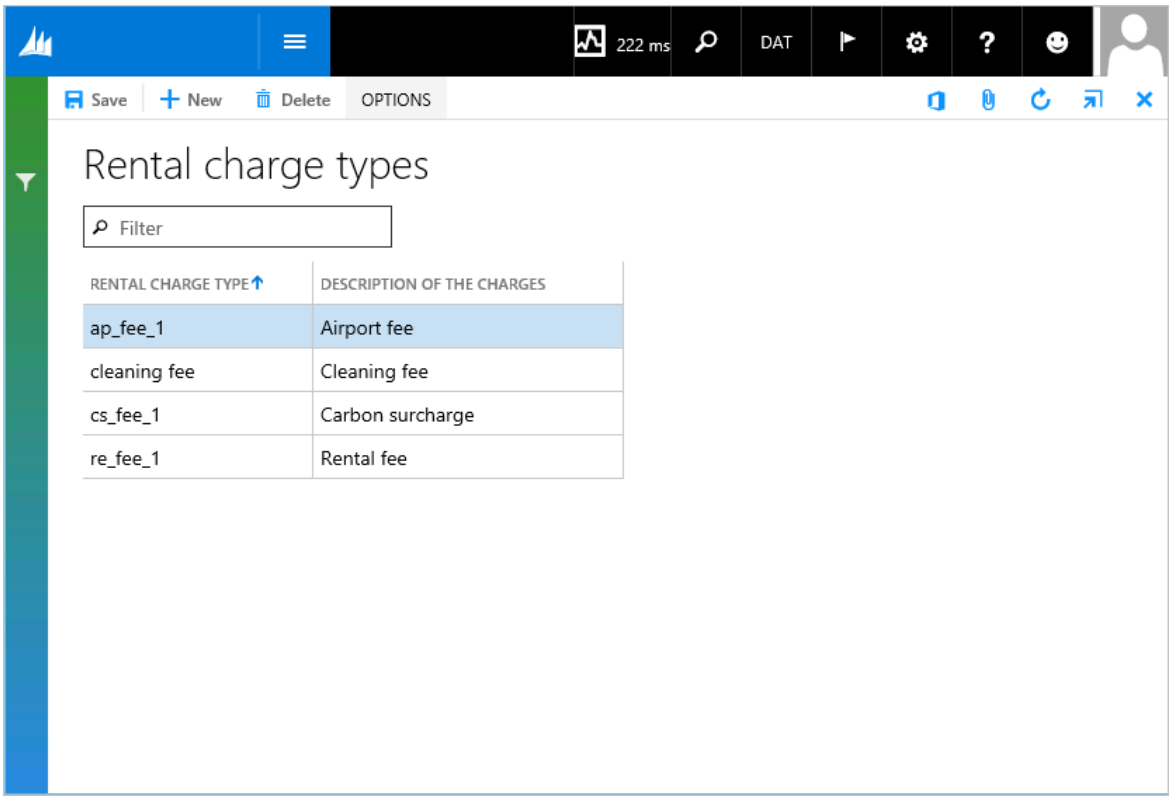
3. The form opens in Internet Explorer.

4. To add a rental charge type, click **New** in the Action Pane at the top of the form. Add the following information.

RENTAL CHARGE TYPE	DESCRIPTION
cleaning	Cleaning fee

5. In the Action Pane, click **Save**.

6. Refresh the browser to see the new record in the list. The following illustration shows how the form should look.



7. The form opens in view mode. Click **Edit** in the Action Pane to switch the form into edit mode. To return to view mode, click **Options** and then **Read mode**.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Build the Customer form

2/18/2021 • 10 minutes to read • Edit Online

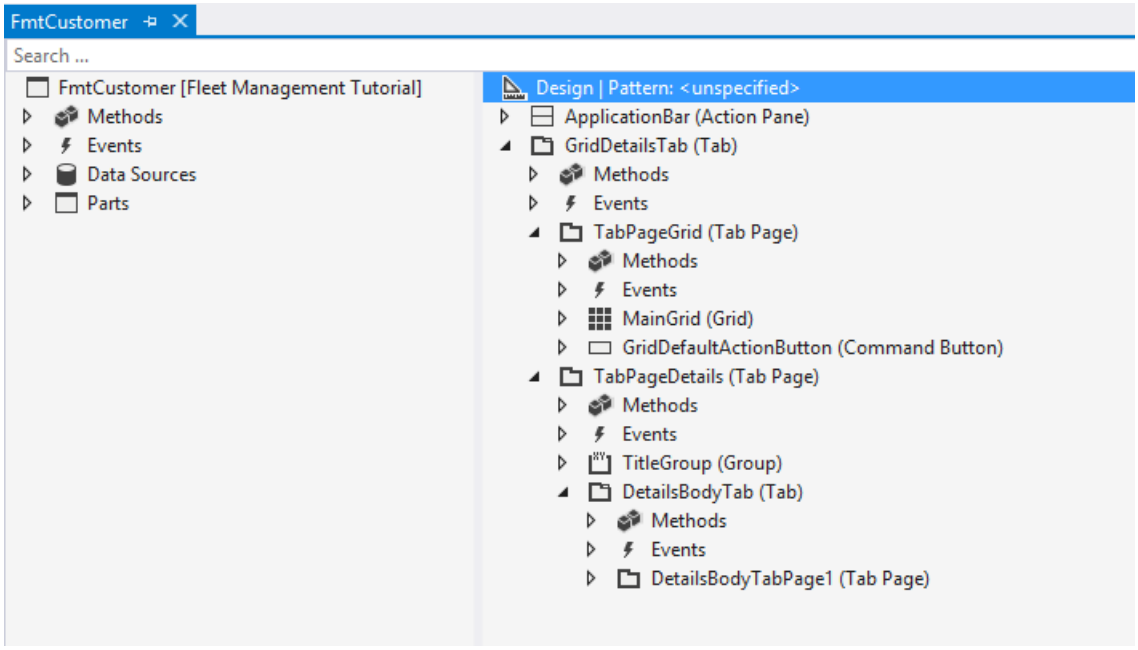
In this lab you'll create a Master Details form and apply the appropriate form pattern and subpatterns. A Master Details form shows primary data that has many fields. For example, the form that you create will show customer information.

Prerequisites

For this tutorial, you will need to access the environment using Remote Desktop, and be provisioned as an administrator on the instance. For more information, see Access Instances.

Overview

To create the form, you'll start from the existing form, **FmtCustomer**. The form represents the old Master Details template. As a part of the tutorial, you'll apply the Master Details pattern, which will enforce a consistent structure for this form type. The following illustration shows the **FmtCustomer** starting artifact.



Key concepts

- Create a Master Details form.
- Apply a form pattern to a form.
- Use the Visual Studio pattern add-ins to get information about form/model pattern coverage.
- Apply subpatterns to form controls.
- View the form using Visual Studio and a browser.
- Determine the amount of remaining patterns work in a model.

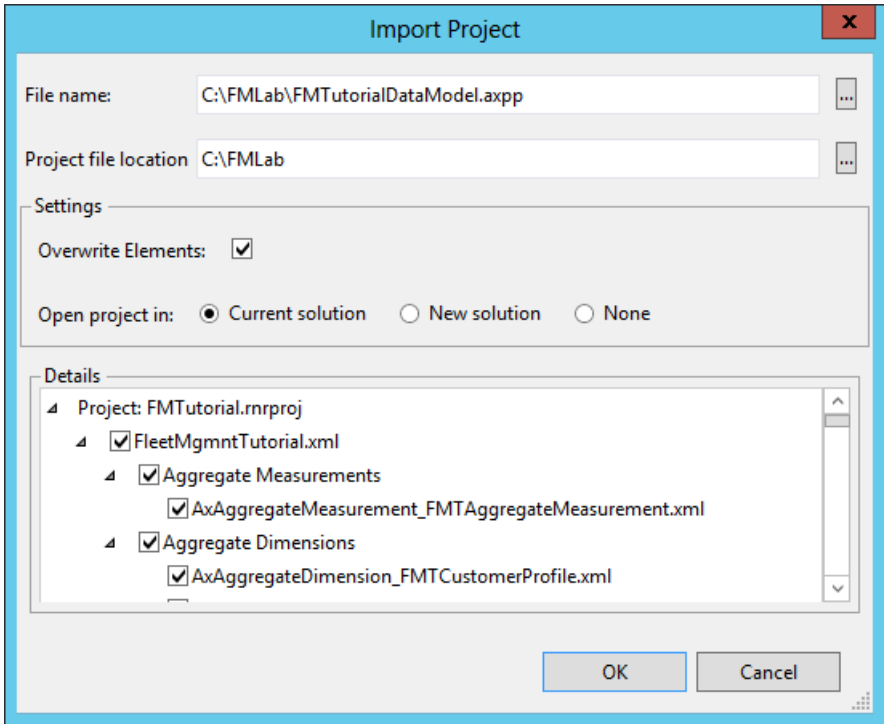
Setup

Import the tutorial project and transactional data

Use Visual Studio to import the tutorial project. The tutorial project includes the artifacts you will use to

complete this tutorial. Use Visual Studio to open the FMTutorial project and load the data for the tutorial. You will use the FMTDataHelper class to load data for the Fleet Management tutorial. If this is the first tutorial you are working on, review Access Instances and make sure you provision your administrator user if you're working on a local VM.

1. Download the Fleet Management sample from https://github.com/Microsoft/FMLab, save it to **C:**, and unzip it.
2. On the desktop, double-click the Visual Studio shortcut to open the development environment.
3. On the **Finance and Operations** menu, click **Import Project**.
4. In the **Import Project** window, next to the **Filename** text box, click the ellipsis button.
5. In the **Select the file to import** window, browse to **C:\FMLab**, click FMTutorialDataModel.axpp, and then click **Open**.
6. In the **Project file location** text box, enter **C:\FMLab**.
7. Select the **Overwrite Elements** option and the **Current solution** radio button. The following illustration shows the completed **Import Project** dialog box.



8. Click **OK**.
9. In **Solution Explorer**, expand **Classes**, and under the **FMTutorial** project, right-click **FMTDataHelper**, and then click **Set as Startup Object**.
10. On the **Build** menu, click **Rebuild Solution**. You use the rebuild to make sure all files in the project are built regardless of timestamps. You can view the build progress in the Output window.
11. After the build completes, press **Ctrl+F5** to run the project. The browser will open and run the class that imports the data.

Open the FMTutorial project

Use Visual Studio to open the FMTutorial project. If you have Visual Studio open and have already loaded the FMTutorial project, you can continue to the next section.

1. If the development environment is not already open, on the Desktop, double-click the Visual Studio shortcut

to open the development environment.

2. On the **File** menu, click **Open** > **Project/Solution**.
3. In the **Open Project** dialog box, browse to C:\FmLab\FMTutorial, select the **FMTutorial** solution, and then click **Open**.
4. The FMTutorial project appears in **Solution Explorer**.

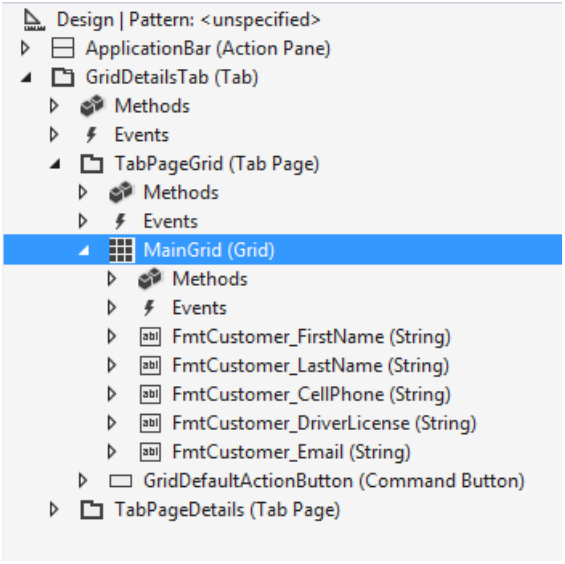
Use a template to create the form

Use Visual Studio to create the **FmtCustomer** form. You'll use a template to create a new master details form. The data source for this tutorial is provided by the starter form. However, you'll add fields to the grid and details view and apply the Master Details form pattern.

1. In **Solution Explorer**, right-click the **FMTutorial** project, point to **Add**, and then click **Existing Item**.
2. In the **Add Existing Item** window, browse to C:\FmLab, select **AxForm_FmtCustomer**, and then click **Add**. The **FmtCustomer** form appears at the bottom of the **FMTutorial** project in Solution Explorer.
3. In Solution Explorer, double-click **FmtCustomer**. The form opens in the form designer.
4. In the Form designer, click **Design**. In the **Properties** window, specify the following values.

PROPERTY	VALUE
Data Source	FmtCustomer
Caption	Customers

5. In the Form designer, click **Design** > **GridDetailsTab** > **TabPageGrid** > **MainGrid**, and then click **MainGrid**.
6. In the **Properties** window, click **Data Source**, and then select **FmtCustomer** to bind the **FmtCustomer** table to the grid. You can now use the fields from the data source to add columns to the grid.
7. Click **Data sources** > **FmtCustomer** > **Fields** to add fields to the grid.
   a. Click **FirstName**, press and hold the **Ctrl** key, and then select the following additional fields in the order shown:
      - LastName
      - CellPhone
      - DriverLicense
      - Email
   b. Drag the highlighted fields to **Design** > **GridDetailsTab**> **TabPageGrid** > **MainGrid**. The following illustration shows the grid after expanding the grid node and adding the fields.



8. Click **Save**.
9. Click **Design** > **GridDetailsTab** > **TabPageDetails** > **TitleGroup** to add the record header to the details view.
10. Click **HeaderTitle**. In the **Properties** window, specify the following values.

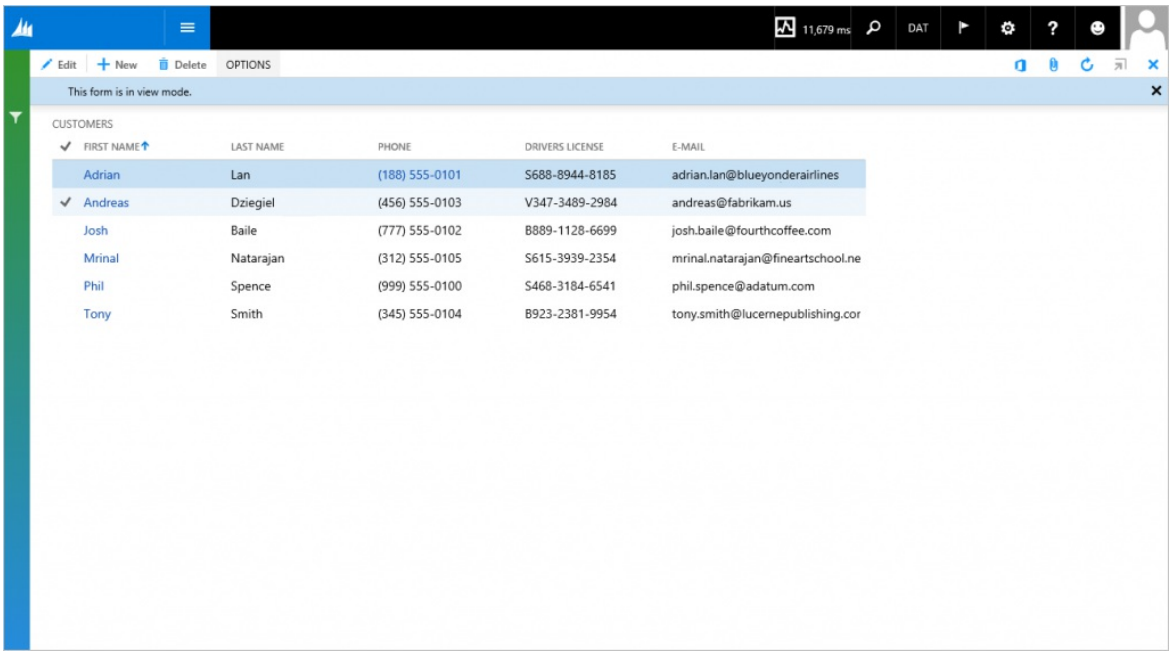
PROPERTY	VALUE
Data Source	FmtCustomer
Data Method	titleFields

11. Click **Design** > **GridDetailsTab** > **TabPageDetails** > **DetailsBodyTab** > **General** to add content to the details view.
    a. Click **FmtCustomer** > **Data sources** > **FmtCustomer** > **Fields**, press and hold the Ctrl key, and then select the following fields:
        - FirstName
        - LastName
        - CellPhone
        - DriverLicense
        - Email
    b. Drag the highlighted fields onto **General**, and then click **Save**.

View the form

Run the form to verify that it loads correctly.

1. In **Solution Explorer**, right-click **FmtCustomer**, and then click **Set as Startup Object**.
2. Press **Ctrl+F5**. The grid view should render like the following illustration.

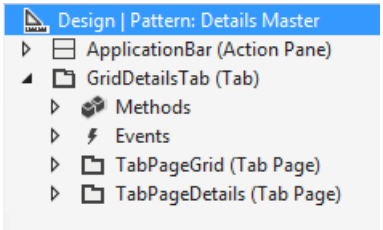


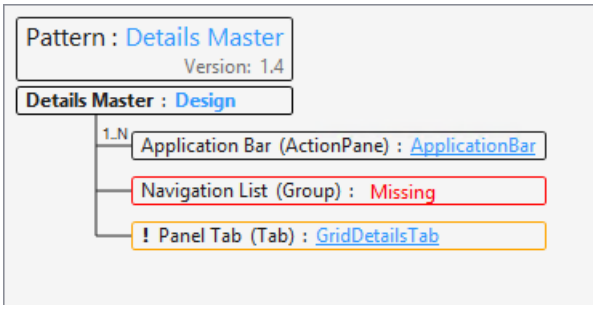
3. On the application bar, click **Open in Microsoft Office** > **Export to Excel** > **Customers** to send the information in the grid view to a Microsoft Excel spreadsheet. (If a dialog appears asking if you're sure you want to leave the page, click "Leave this page".) When asked, click **Open** to view the data in Excel.
4. Close Excel.
5. Click **Tony** to navigate to the details view for that record.
6. Click **Close** (or the browser Back button) to go back to the grid view.

Apply a pattern to the form

Use Visual Studio to apply the Master Details form pattern to the **Customer** form. Applying a form pattern ensures your form has the expected structure. It also simplifies the design experience by automatically setting the values of properties in the nodes that are part of the pattern.

1. Right-click **Design**, point to **Apply pattern,** and then click **Details Master**.

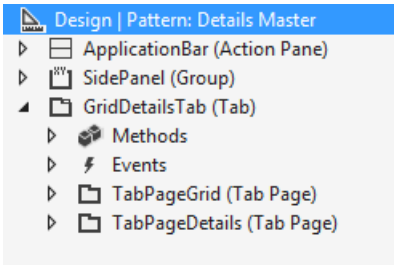


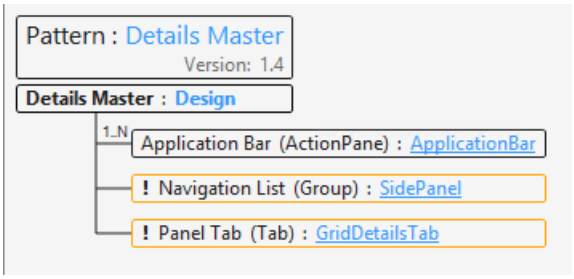


2. Add the missing Navigation List group. The red highlighting in the Patterns Information Panel indicates that this control is missing.
   a. Right-click **Design**, point to **New**, and then click **Group**.
   b. In the **Properties** window, in the **Name** property, enter **SidePanel**.

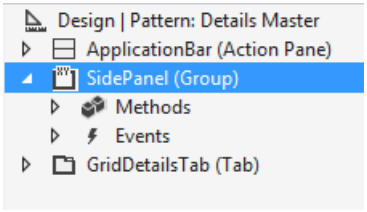
c. Click **SidePanel**, and press **Alt+Up** to move this group above the **GridDetailsTab (Tab)**.

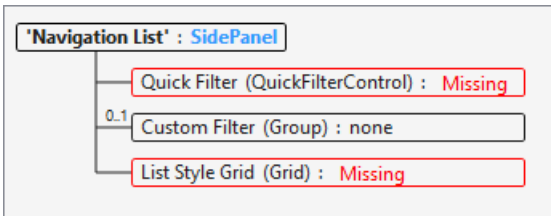
3. Click **Design** again. The yellow highlighting around the **Navigation List** and the **Panel Tab** indicate that there are problems that need to be resolved under each of these nodes before the pattern can be successfully applied.





4. In the Patterns Information Panel, click **SidePanel**.





5. Add the missing controls.

a. Right-click **SidePanel**, point to **New**, and then click **QuickFilter**.

b. In the **Properties** window, specify the following values.

PROPERTY	VALUE
Name	SidePanelQuickFilter
Target Control	MainGrid *This QuickFilter should have the same columns available for filtering as the main grid on the form*

c. Right-click **SidePanel**, point to **New**, and then click **Grid.**

PROPERTY	VALUE
Name	NavigationList

PROPERTY	VALUE
Datasource	FmtCustomer

6. Add identifying fields to the Navigation list. Right-click **NavigationList**, point to **New**, and then click **String**.

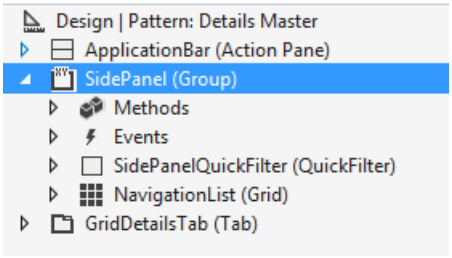
   a. In the **Properties** window, specify the following values.

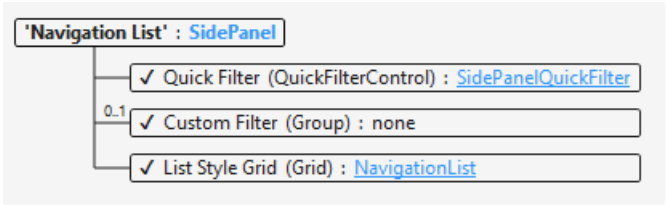
PROPERTY	VALUE
DataSource	FmtCustomer
DataMethod	fullName
Name	FmtCustomer_FullName

   b. Expand **Data Sources** > **FmtCustomer** > **Fields** to add phone numbers to the Simple list.

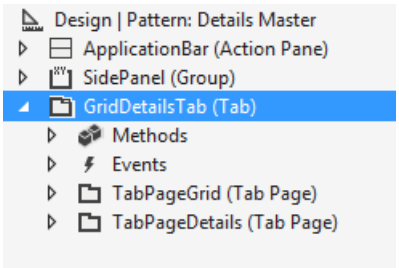
   c. Drag the **CellPhone** field onto the grid under **Design** > **SidePanel** > **NavigationList**.

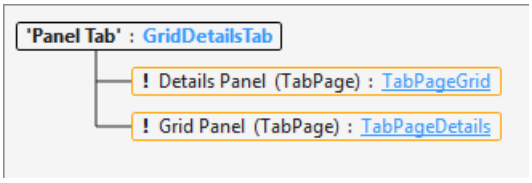
7. Click **SidePanel**. Notice the **Patterns Information Panel** is now indicating that the controls in this subtree are in full compliance with the pattern.





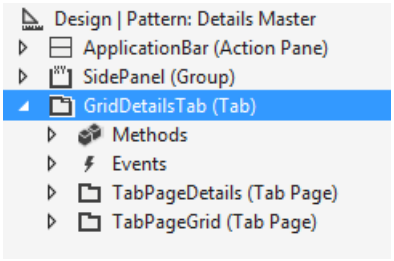
8. Click **Design** > **GridDetailsTab**. The yellow highlighting around the subnodes indicates that there are problems that need to be resolved under both nodes before the form pattern can be successfully applied.

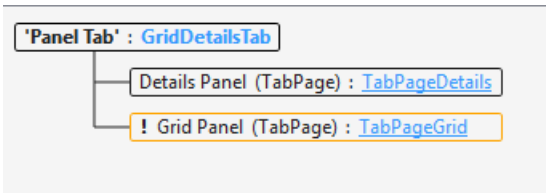




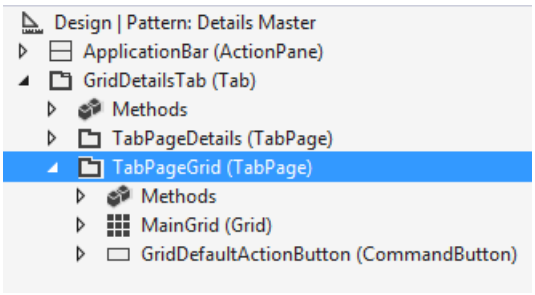
9. Notice that the pattern expects the **Grid Panel** to be after the **Details Panel**. Click **TabPageGrid** and press **Alt+Down** to move that tab below the **Details Panel**.

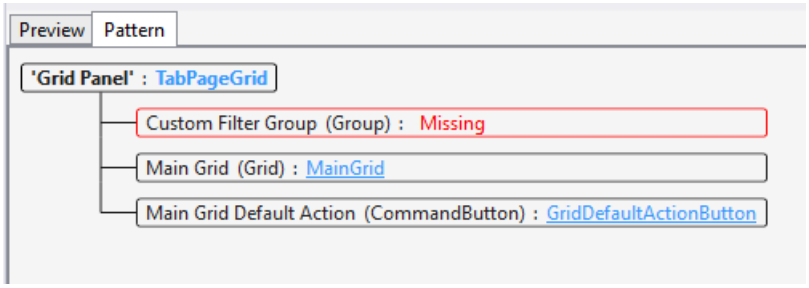
10. Click **GridDetailsTab**. The **TabPageDetails** tab page now adheres to the pattern. However, the **TabPageGrid** tab page needs additional attention.





11. Click **TabPageGrid**. Focus in the designer is now on **TabPageGrid**, and the **Patterns Information Panel** has been updated.



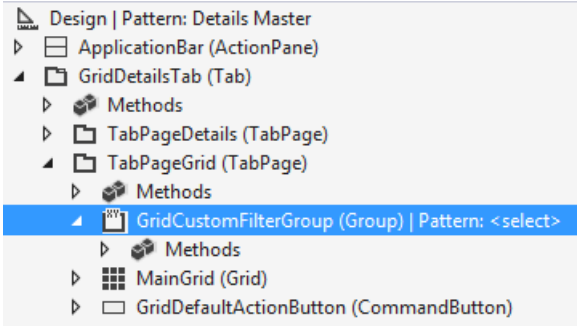


12. The **Patterns Information Panel** now indicates a missing Group control at the top of the **TabPageGrid** container.

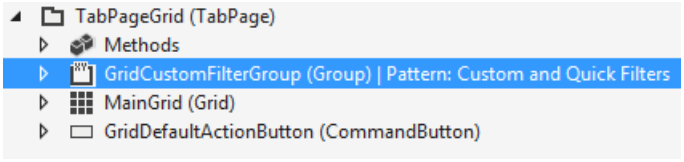
    a. Right-click **TabPageGrid**, point to **New**, and then click **Group**.

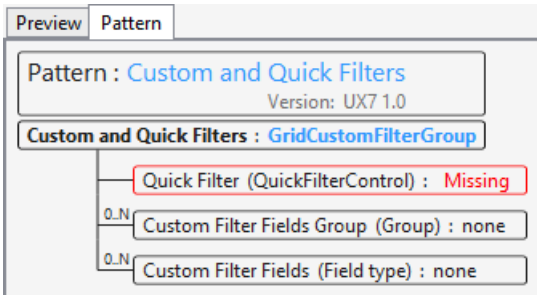
    b. Press **Alt+Up** two times to position the group as the first control in the group.

    c. In the **Properties** window, in the **Name** property, enter **GridCustomFilterGroup**.



13. The pattern is looking for a subpattern to be applied to **GridCustomFilterGroup**. Right-click **GridCustomFilterGroup**, point to **Apply pattern**, and then click **Custom and Quick Filters**.





14. The **Custom and Quick Filters** subpattern requires a QuickFilter control.

    a. Right-click **GridCustomFilterGroup**, point to **New**, and then click **QuickFilter**.

    b. In the **Properties** window, specify the following values.

PROPERTY	VALUE
Name	MainGridQuickFilter
Target Control	MainGrid

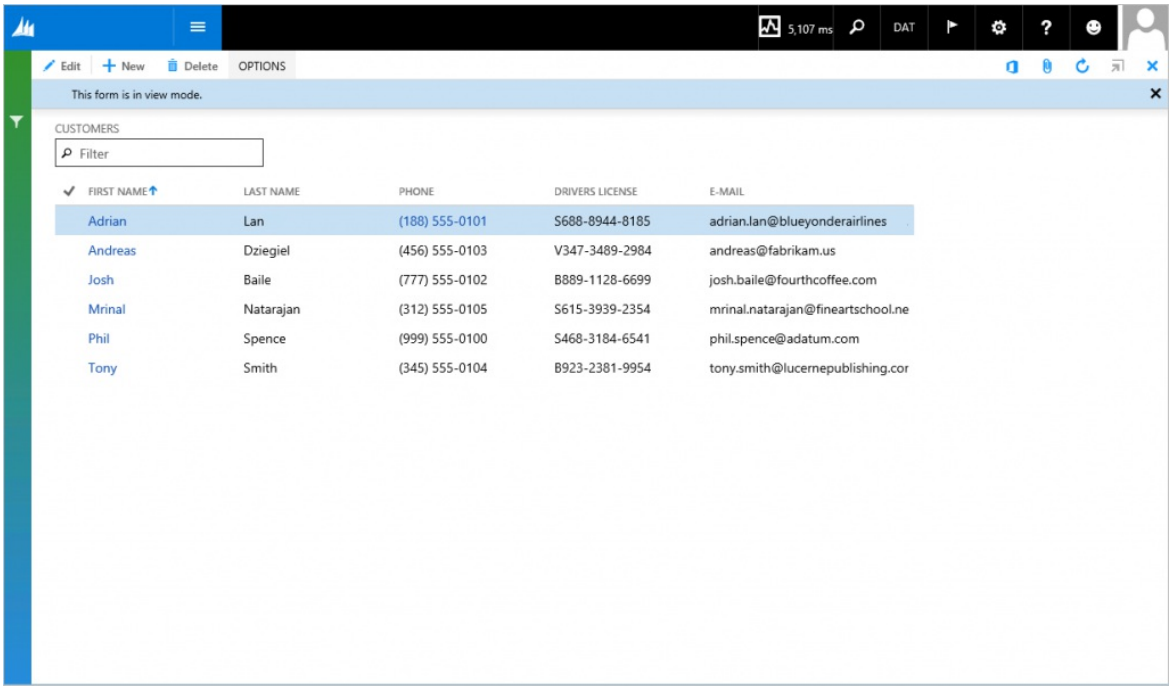
15. Browse up the control tree to design and notice how the **Patterns Information Panel** now shows no issues under each of the controls.

16. Press **Ctrl+S** to save the form.

View the details form

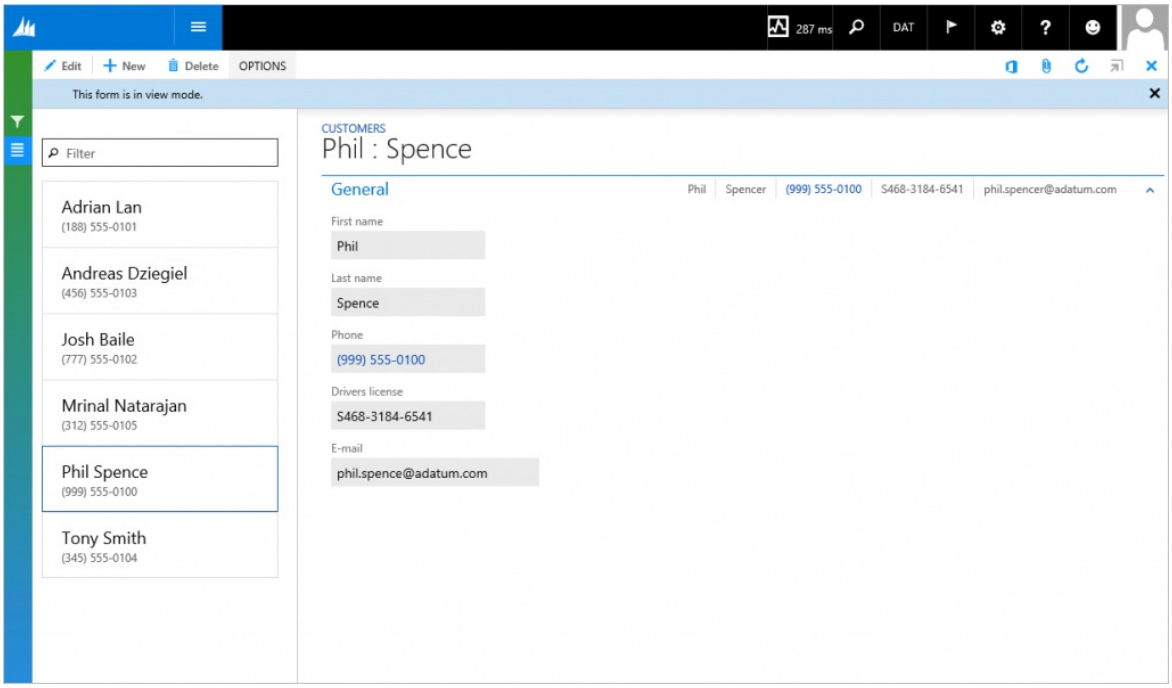
Run the form to see the Details view and the Grid view.

1. Press **Ctrl+F5** to run the project. The following illustration shows how the grid view appears.



2. Click **Phil** to go to the details view for that record.

3. Click the **Show list** button on the left side of the form to open the navigation list.

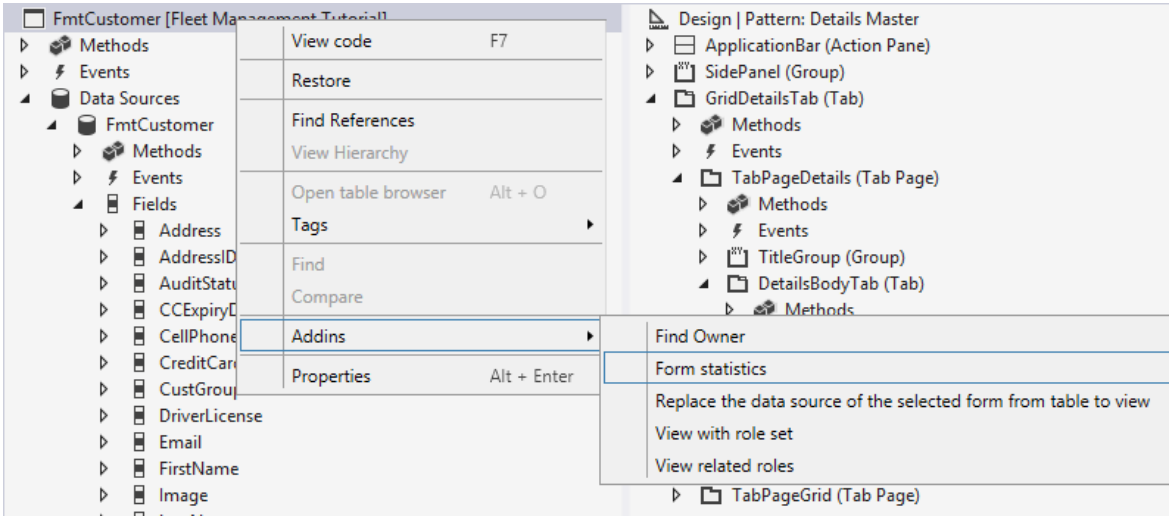


4. To go back to the grid view, click **Close** (or the browser Back button).

5. Return to Visual Studio.

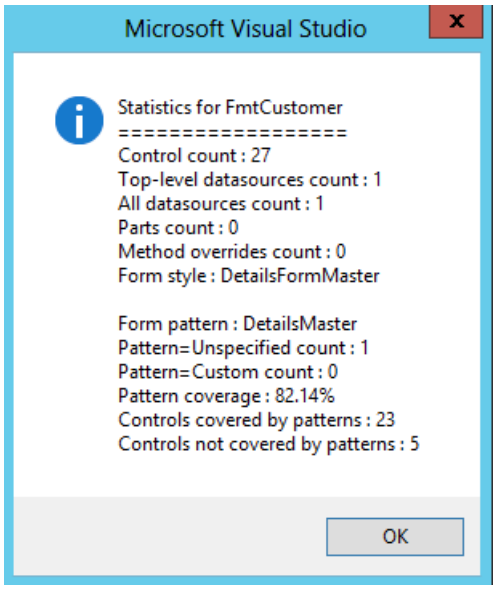
Add subpatterns

1. In Visual Studio, in the Form designer, right-click **FmtCustomer**, point to **Addins**, and then select **Form statistics**.

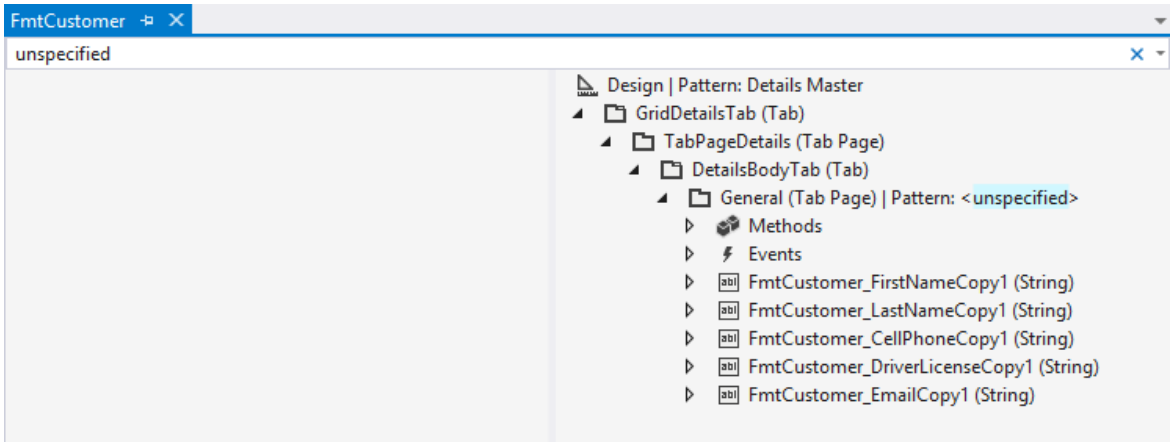


The **Form Statistics** add-in provides several useful data points about the state of the form. This includes:

- **Pattern=Unspecified count** – The number of nodes for which no form pattern or subpattern has been applied.
- **Pattern=Custom count** – The number of nodes for which a custom pattern was applied, meaning the structure did not fit with any existing pattern.
- **Pattern coverage** – The percentage of controls on the form that are covered by the form pattern or a subpattern. A value of 100% indicates a fully covered form.

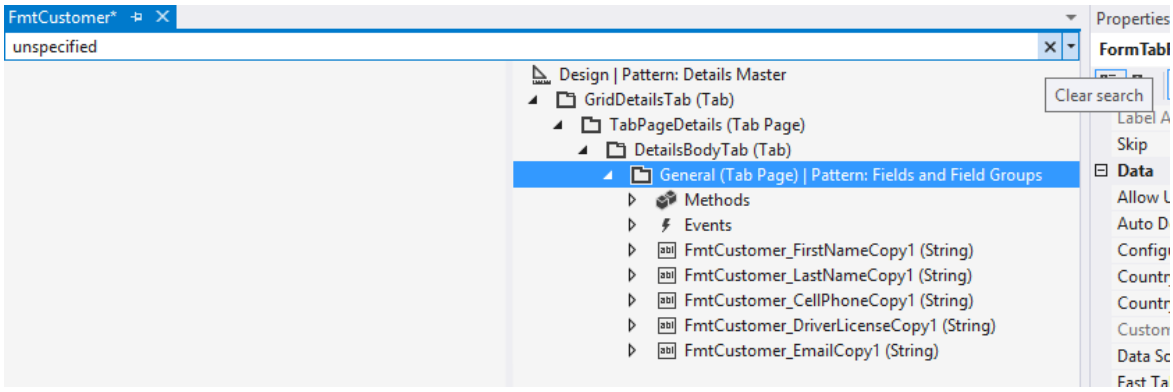


2. To complete pattern coverage for this form, the Pattern=Unspecified count should be zero. Use the Visual Studio form search to find all instances of “unspecified “in the form.



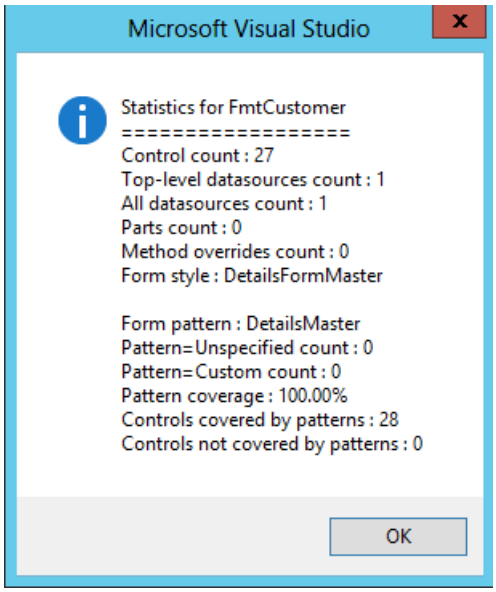
3. Because the **General** tab page contains only input controls and no custom layout is required for this FastTab, the Fields and Field Groups pattern should be applied to guarantee a responsive layout. Right-click **General**, point to **Apply pattern**, and then select **Fields and Field Groups**.

4. On the far right of the screen, click **Clear search**.



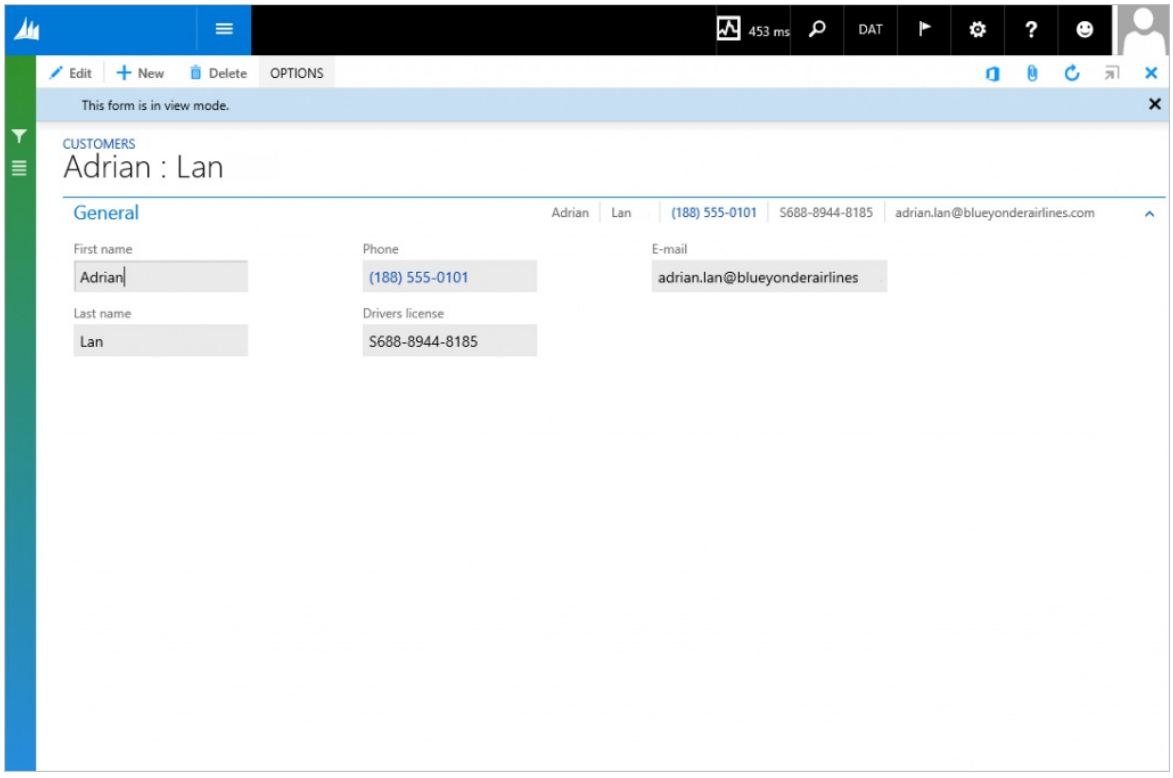
5. Press **Ctrl+S** to save the form.

6. Repeat step 1 to run the **Form Statistics** add-in a second time to verify the form is fully covered by patterns.



7. Press **Ctrl+F5** to run the project and see the updated form.

8. Click **Adrian** to go to the details view. The following illustration shows how the details view now appears after applying the Fields and Field Groups subpattern so that the fields lay out responsively. By changing the browser width, you'll see how the field layout adjusts to better fill the width of the browser.

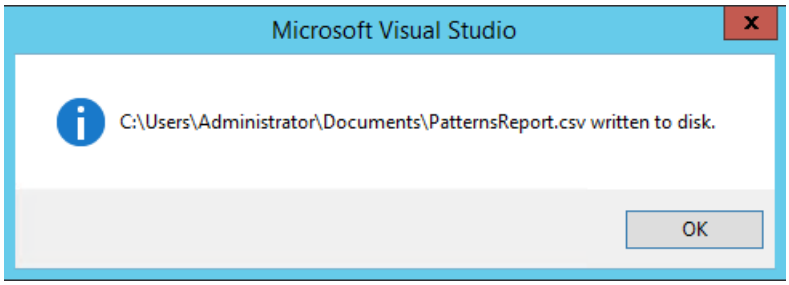


9. Return to Visual Studio

Determine the amount of remaining patterns work in a model

1. Click **Finance and Operations**, point to **Addins**, and then select **Run form patterns report**.

   A notification dialog will be shown when the form patterns report has been generated.

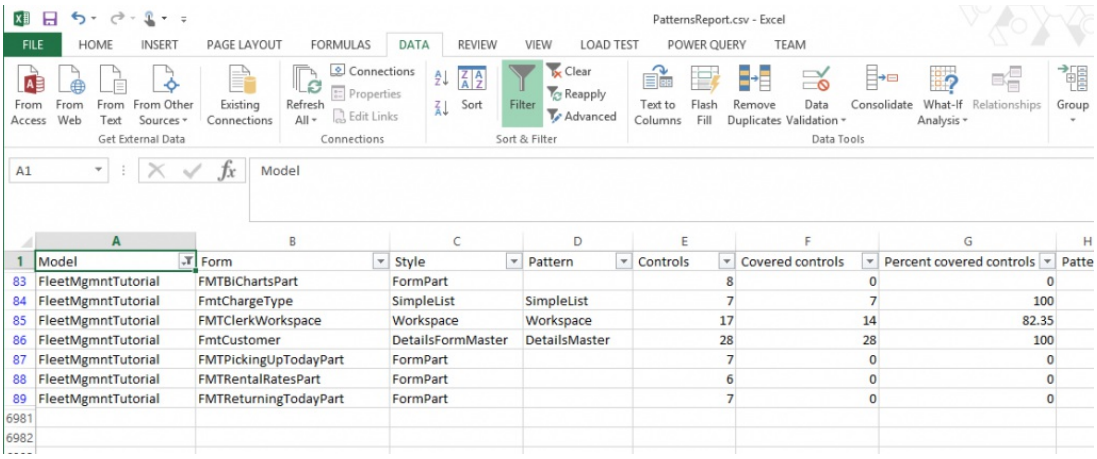


2. Open the PatternsReport file in Excel.

3. Filter the report to the Fleet Management Tutorial model.

   a. Click **Data** > **Filter**.

   b. Filter the **Model** column to FleetMgmntTutorial.



	A	B	C	D	E	F	G	H
1	Model	Form	Style	Pattern	Controls	Covered controls	Percent covered controls	Patte
83	FleetMgmntTutorial	FMTBiChartsPart	FormPart		8	0	0	
84	FleetMgmntTutorial	FmtChargeType	SimpleList	SimpleList	7	7	100	
85	FleetMgmntTutorial	FMTClerkWorkspace	Workspace	Workspace	17	14	82.35	
86	FleetMgmntTutorial	FmtCustomer	DetailsFormMaster	DetailsMaster	28	28	100	
87	FleetMgmntTutorial	FMTPickingUpTodayPart	FormPart		7	0	0	
88	FleetMgmntTutorial	FMTRentalRatesPart	FormPart		6	0	0	
89	FleetMgmntTutorial	FMTReturningTodayPart	FormPart		7	0	0	
6981								
6982								

The report shows pattern-related information regarding the forms in this model including the top-level form pattern currently applied, and the percentage of controls on the form covered by patterns. This can be used to track the remaining patterns work in one or more models.

Build navigation

2/18/2021 • 6 minutes to read • Edit Online

In this tutorial, you will add navigational elements to a workspace and the navigation pane.

Prerequisites

For this tutorial, you need to access the environment using Remote Desktop, and be provisioned as an administrator on the instance. For more information, see Access Instances.

Key concepts

- A *workspace* is an overview page that is specific to a particular subject area. Workspaces are common to all users. In this tutorial, you will add content into an existing workspace.
- The *dashboard* is the default home page for each user.
- *Tiles* are securable objects that can be shown on a workspace or the dashboard. They can be secured by using menu items.

Setup

If this is the first tutorial that you are working on, review Access Instances and make sure that you provision your administrator user if you are working on a local VM.

Import the tutorial project

If you have already imported the Fleet management tutorial project, skip to the next section.

1. Download the Fleet Management sample from https://github.com/Microsoft/FMLab, save it to **C:**, and unzip it.
2. In Visual Studio, on the **Finance and Operations** menu, click **Import Project**.
3. In the **Import Project** window, next to the **Filename** text box, click the ellipsis button.
4. In the **Select the file to import** window, browse to **C:\FMLab**, click **FMTutorialDataModel.axpp,** and then click **Open**.
5. In the Project file location text box, enter **C:\FMLab.**
6. Select the **Overwrite Elements** option, and then click **OK**.

Import transactional data

1. In Visual Studio, open the **FMTutorial** project. On the **File** menu, point to **Open**, and then click **Project/Solution**.
2. In the **Open Project** dialog box, browse to C:\FMLab\FMTutorial, and then click FMTutorial. Click **Open**. The **FMTutorial** project appears in **Solution Explorer**.
3. Use the FMTDataHelper class to load data for the Fleet Management tutorial. In **Solution Explorer**, in the FMTutorial project, expand **Classes**, right-click **FMTDataHelper**, and then click **Set as Startup Object**.
4. From the **BUILD** menu, click **Rebuild Solution**. You use the rebuild to update the timestamps of the imported artifacts. You can view the build progress in the **Output** window.
5. Press **Ctrl+F5** to run the project and load the data.

Add a tile to the tutorial workspace

First, we will add a new tile to the form FMTClerkWorkspace.

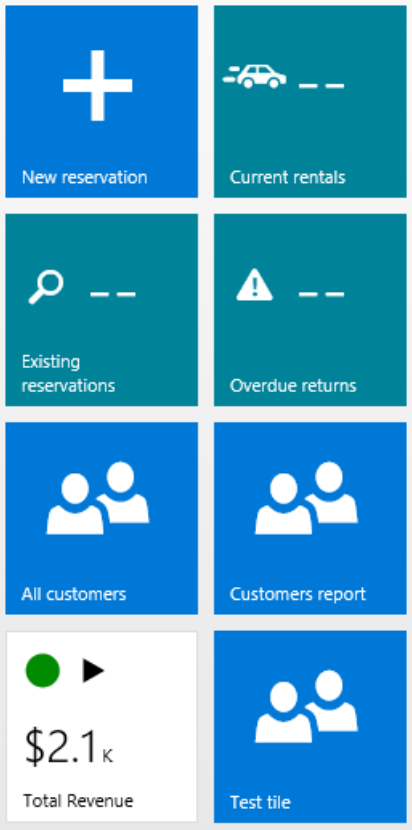
1. In **Solution Explorer**, expand **Forms** and then double-click **FMTClerkWorkspace**.
2. In the designer, expand **PanoramaBody**.
3. Right-click **TileContainer**, and then click **New** > **Tile Button.**
4. Specify the following properties for the new tile button.

PROPERTY	VALUE
Text	Test tile
Tile	FMTAllCustomersTile

   This will create a duplicate of the existing **All customers** tile.

5. In **Solution Explorer**, click **Forms** > **FMTClerkWorkspace**, right-click, and then select **Set as Start-up Object**. Setting a start-up object is necessary to allow Visual Studio to launch when you press Ctrl+F5 in step 7. Setting this form as the start-up object will cause the work-in-progress Fleet management clerk workspace to appear after you press Ctrl+F5. We will preview this form again later in detail.
6. Right-click **FMTutorial**, and then click **Rebuild**.
7. Press **Ctrl+F5** to run the project.

After you build and run the project, the Fleet management clerk workspace will launch. The new tile named, **Test tile**, that you created will be included in the first section of the workspace, at the end of the set of tiles.



NOTE

The tile will not navigate anywhere when clicked. To enable this, you can define a Menu Item Name on FMTAllCustomersTile, under **Tiles** in **Solution Explorer**.

Add a new workspace to the navigation pane

Next, we will add the FMTClerkWorkspace form to the navigation pane. We will do this in two locations:

- The **All workspaces** list.
- A new item in the area list containing a menu structure that shows the workspace.

Create a menu item that points to the FMTClerkWorkspace workspace

1. Right-click **FMTutorial**, point to **Add**, and then click **New Item**.
2. Click **AX Artifacts** > **User Interface** > **Display Menu Item**. In the **Name** property, enter **FMTClerkWorkspace**.
3. Click **Add**.
4. Specify the following properties for the new menu item.

PROPERTY	VALUE
Label	Reservation management tutorial
Object	FMTClerkWorkspace

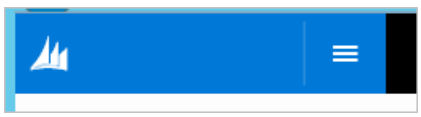
Create a tile that points to the FMTClerkWorkspace workspace menu item

1. Right-click **FMTutorial**, point to **Add**, and then click **New Item**.
2. Click **AX Artifacts** > **User Interface** > **Tile**. In the **Name** property, enter **FMTClerkWorkspace**.
3. Click **Add**.
4. Specify the following properties for the new tile.

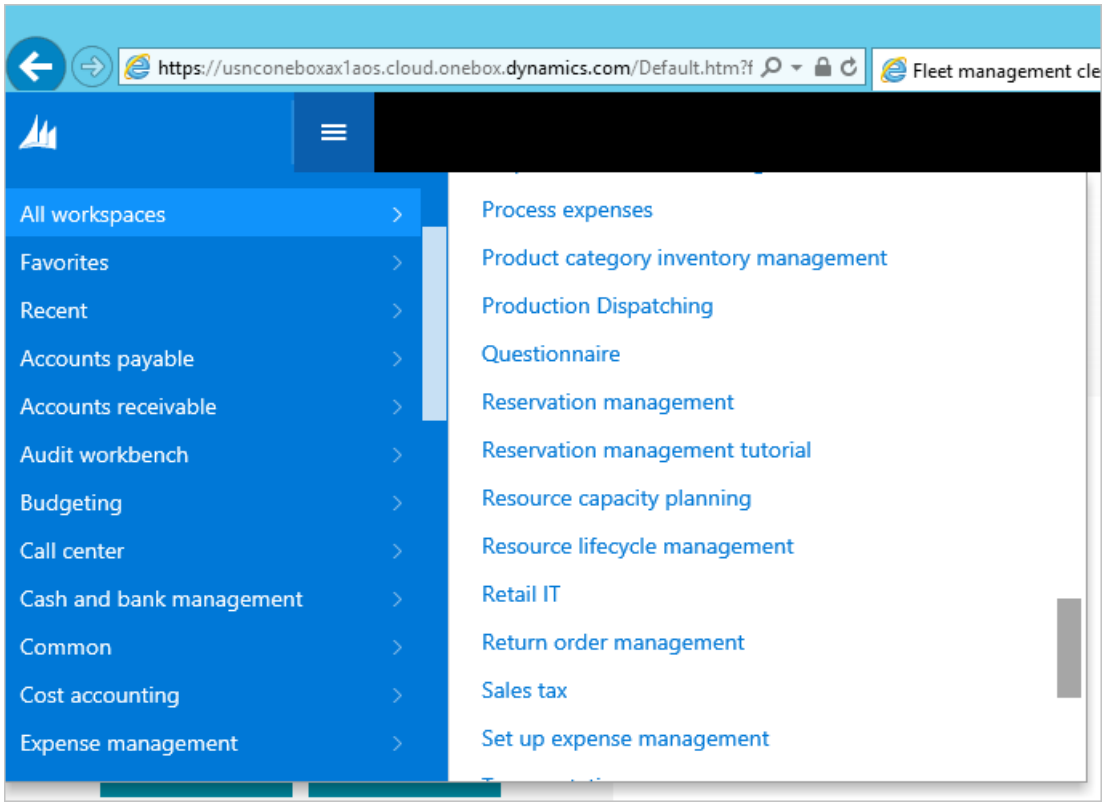
PROPERTY	VALUE
MenuItemName	FMTClerkWorkspace

Add a menu extension for the navigation pane

1. In **Application Explorer**, click **User Interface** > **Menus**, right-click **NavPaneMenu**, and then click **Create extension**.
2. In **Solution Explorer**, double-click **NavPaneMenu.Extension**.
3. In the designer, right-click **NavPaneMenu.Extension**, point to **New**, and then click **Submenu**.
4. Select the new submenu. In the **Name** property, enter **NavPaneMenuFleetTutorial**.
5. In **Solution Explorer** or **Application Explorer**, locate the **FMTClerkWorkspace** tile, and drag it onto the newly created submenu. Click **Save**.
6. Right-click **FMTutorial**, and then click **Rebuild**.
7. Press **Ctrl+F5** to run the project. After you build and run the project, the navigation pane will contain a link to the new workspace. Open the navigation pane by clicking the navigation pane button (three lines) at the top right of the application window.



8. When you open the navigation pane, select **All workspaces**, and scroll down in the list after it opens. You should see the following new Reservation management tutorial workspace in the list.



Add the form to the main menu structure

Now you'll add a new main menu section that contains a tile that points to the tutorial workspace. You will then add a link to the same form in this section. This will demonstrate the appearance of a non-workspace form link.

1. In Visual Studio, in **Solution Explorer**, right-click **FMTutorial**, point to **Add**, and then click **New Item**.
2. Click **AX Artifacts** > **User Interface** > **Menu**. In the **Name** property, enter **FleetManagementTutorial**.
3. Click **Add**.
4. In **Solution Explorer**, double-click the new menu **FleetManagementTutorial** if it isn't already open.
5. In the properties list, set the **Label** property to **Fleet management tutorial**.
6. In the designer, right-click **FleetManagementTutorial**, and click **New** > **Submenu**.
7. Specify the following properties for the new submenu.

PROPERTY	VALUE
Name	Workspaces
Label	Workspaces

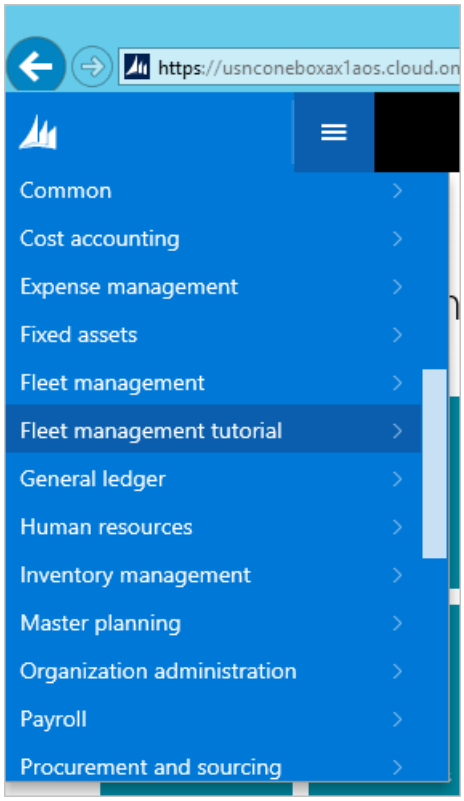
8. In **Solution Explorer** or **Application Explorer**, locate the **FMTClerkWorkspace** display menu item and drag it onto the new **Workspaces** submenu.
9. In the designer, right-click **FleetManagementTutorial**, and then click **New** > **Submenu**.
10. Specify the following properties for the new submenu.

PROPERTY	VALUE
Name	Common
Label	Common

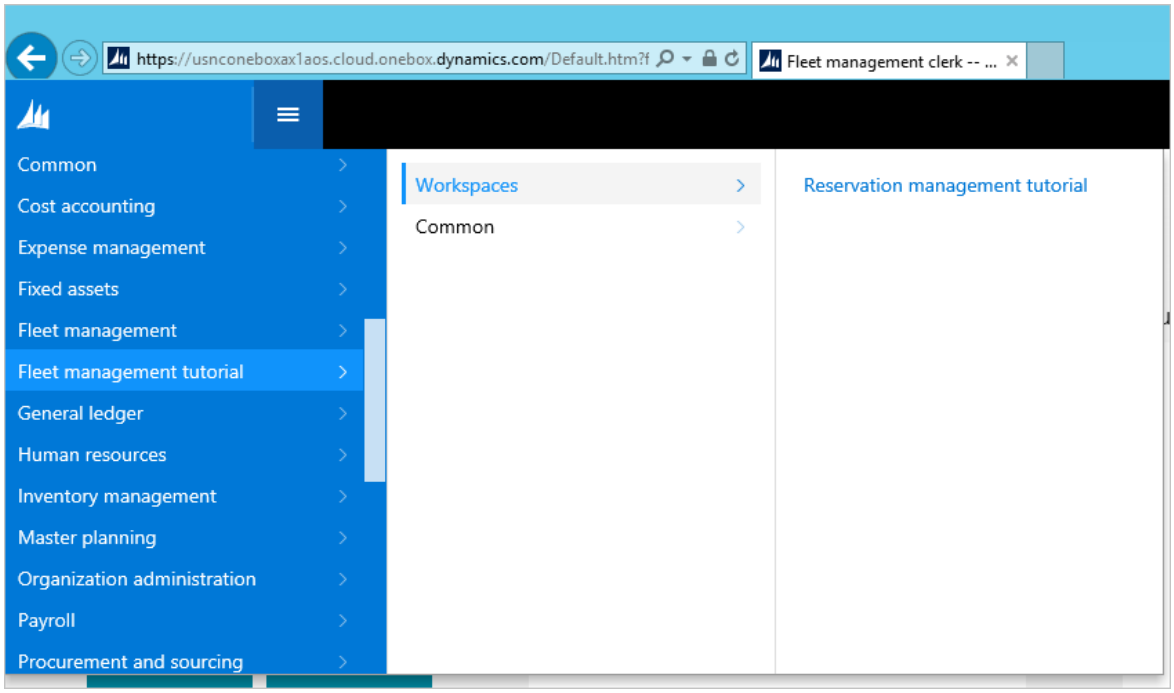
11. In **Solution Explorer** or **Application Explorer**, locate the **FMTClerkWorkspace** display menu item and drag it onto the new **Common** submenu.
12. In **Application Explorer**, click **User Interface** > **Menus** > **MainMenu**. Right-click **MainMenu**, and then click **Create extension**.
13. In **Solution Explorer**, locate and open the new extension. Select and double-click **MainMenu.Extension** to open it.
14. In the designer, right click **MainMenu.Extension**, point to **New**, and then click **Menu reference**.
15. Specify the following properties for the new menu reference.

PROPERTY	VALUE
Name	FleetManagementTutorial
Menu Name	FleetManagementTutorial

16. Click **Save**.
17. Right-click **FMTutorial**, and then click **Build**.
18. Press **Ctrl+F5** to run the project.
19. Go to the main menu section you just modified. Open the navigation pane and scroll down until you see the new top-level **Fleet management tutorial** menu. You may need to clear your browser cache by pressing **Ctrl+F5**.



20. Click **Fleet management tutorial** > **Workspaces** to expand that submenu. Your navigation pane should look like the following.



If you click on the **Common** submenu, you will see the menu item that you modeled there. You can click either of these links to check that you have set up the references correctly. If you have set up the references correctly, the tutorial workspace you're working on should open when clicked on.

Modify a workspace with a tile, list, and data cache

2/18/2021 • 22 minutes to read • Edit Online

In this tutorial, you will create a new tile and include it in the summary section of a workspace, build a new list for a workspace, and create a data cache for the list in the workspace.

Prerequisites

For this tutorial, you must access the environment by using Remote Desktop, and you must be provisioned as an administrator on the instance. For more information, see Deploy and access development environments.

Key concepts

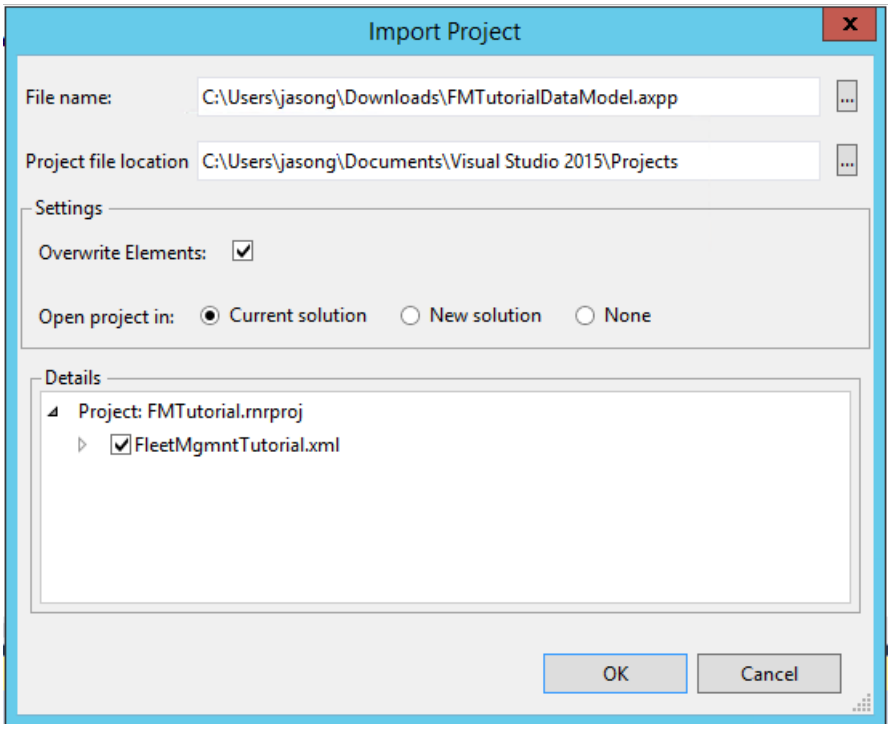
- Learn about and use form patterns that are related to workspaces.
- Create a new tile, and include it in the **Summary** section of a workspace.
- Build a new list for a workspace.
- Create a data cache for the list in the workspace.

Setup

Import the tutorial project and transactional data

Use Microsoft Visual Studio to import the tutorial project. The tutorial project includes the artifacts that you will use to complete this tutorial. Use Visual Studio to open the FMTutorial project and load the data for the tutorial. You will use the **FMTDataHelper** class to load data for the Fleet Management tutorial. If this is the first tutorial that you're working on, review Deploy and access development environments, and make sure that you provision your administrator user if you're working on a local virtual machine (VM).

1. Download the **FMTutorialDataModel.axpp** file from the Microsoft Dynamics Lifecycle Services (LCS) methodology, and copy it to the **Downloads** folder of the VM.
2. On the desktop, double-click the Visual Studio shortcut to open the development environment.
3. On the **Dynamics 365** menu, click **Import Project**.
4. In the **Import Project** dialog box, next to the **File name** field, click the ellipsis (...) button.
5. In the **Select the file to import** dialog box, browse to the **Downloads** folder, click **FMTutorialDataModel.axpp**, and then click **Open**.
6. Select the **Overwrite Elements** check box and the **Current solution** option. The following illustration shows the completed **Import Project** dialog box.



7. Click **OK**.
8. In Solution Explorer, expand **Classes**, and then, under the **FMTutorial** project, right-click **FMTDataHelper**, and then click **Set as Startup Object**.
9. On the **Build** menu, click **Rebuild Solution**. Use the rebuild to make sure that all the files in the project are built, regardless of timestamps. You can view the build progress in the **Output** window.
10. After the build is completed, press **Ctrl+F5** to run the project. The browser opens and runs the class that imports the data.

Open the FMTutorial project

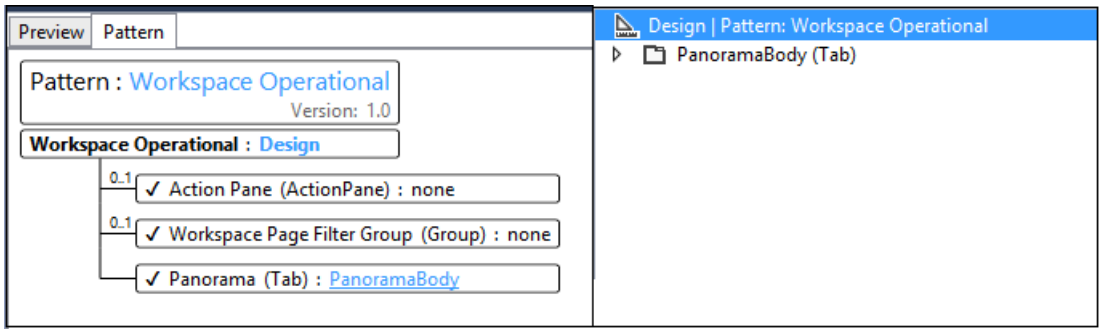
Use Visual Studio to open the FMTutorial project. If you have Visual Studio open and have already loaded the FMTutorial project, you can continue to the next section.

1. If the development environment isn't already open, on the desktop, double-click the Visual Studio shortcut to open it.
2. On the **File** menu, click **Open** > **Project/Solution**.
3. In the **Open Project** dialog box, browse to **Documents** > **Visual Studio 15.0** > **Projects**, select the **FMTutorial** solution, and then click **Open**.
4. The FMTutorial project appears in Solution Explorer.

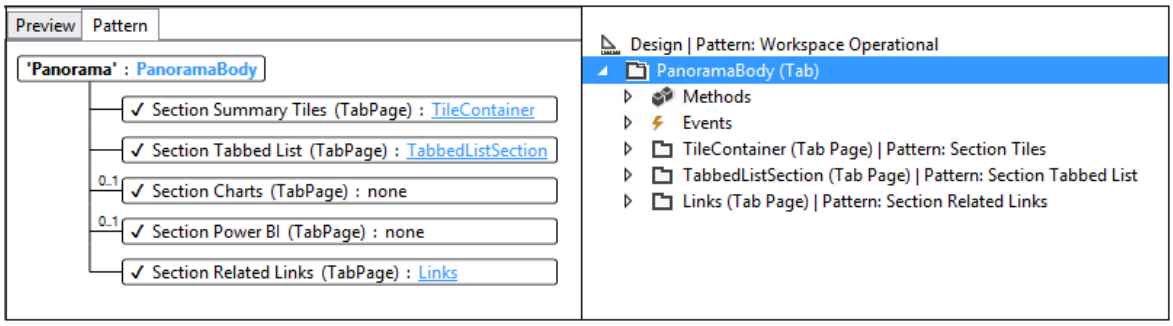
Exercise 1: Understand the operational workspace pattern

Before you start to make adjustments to **FmtClerkWorkspace** form, you will look at the current state of the form to better understand what content is already there and how that contents fits the Operational Workspace pattern.

1. In Solution Explorer, double-click the **FmtClerkWorkspace** form to open it in the designer.
2. Click the **Design** node.
3. Click the **Pattern** tab. Operational workspaces have an optional Action Pane and optional filter group (as indicated by the 0..1 notation to the left of those nodes). However, the panorama-style tab is required by this pattern. The **Patterns** tab shows that the PanoramaBody control matches the required Tab in the pattern, but there are no corresponding controls for the optional items at this level of the pattern.



4. Click **PanoramaBody**.



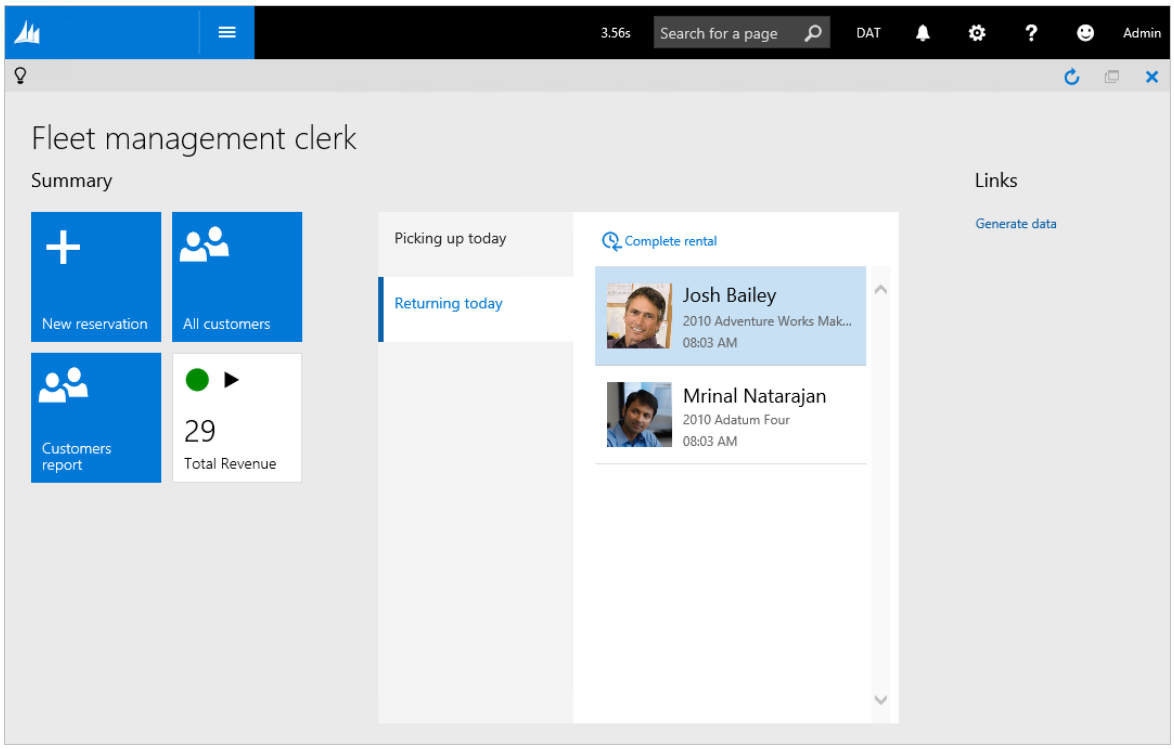
All workspaces have three required sections:

- **Summary section** – This section is intended to contain tiles or form parts, which correspond to a card or chart.
- **Tabbed list** – This section consists of one or more lists of data that is relevant to the user's work. Only one list is shown at a time, and each list can optionally include local filters and actions. Individual lists are modeled inside form part controls.
- **Related links** – This section consists of important or commonly used links for this activity or persona.

Operational workspaces can optionally include a panorama section that contains up to two charts (the Section Charts tab page) and a Power BI section.

View the workspace

1. In Solution Explorer, right-click the **FmtClerkWorkspace** form, and then click **Set as Startup Object**.
2. Press **Ctrl+F5** to build and run the form. The form opens in Internet Explorer.



Exercise 2: Create a new tile for the workspace

Now that you understand the content structure of a workspace, you will see how to add content to a workspace. For example, one important piece of information for this workspace might be the number of rentals that are currently in progress. In this section, you will add the required metadata to add a new tile to the **Summary** section of the **FmtClerkWorkspace** form to show this information. To make this tile work correctly, you will have to add four metadata artifacts: a query, a menu item, a tile, and a tile button.

Add a query that retrieves current rentals

All tiles require a backing query to retrieve the correct information.

1. In Solution Explorer, in the **FMTutorial** project, right-click the **Queries** folder, point to **Add**, and then click **New item**.
2. Click **Dynamics 365 Items** > **Data Model** > **Query**. For the **Name** property, enter **FMTRental_Current**.
3. Click **Add**.
4. If the new **FMTRental_Current** query isn't already open in the designer, double-click it in Solution Explorer.
5. In the designer, right-click **Data Sources**, and then click **New Data Source**.
6. In the **Properties** window, set the following properties.

PROPERTY	VALUE
Table	FMTRental
Dynamics Fields	Yes

7. Right-click **Ranges**, and then click **New Range**.
8. In the **Properties** window, set the following properties.

PROPERTY	VALUE
Field	State
Value	InProgress

9. Right-click **Order By**, and then click **New Field**.
10. In the **Properties** window, set the following properties.

PROPERTY	VALUE
Direction	Descending
Data Source	FMTRental
Field	StartDate

11. Press **Ctrl+S** to save.

Add the corresponding menu item

1. In Solution Explorer, in the **FMTutorial** project, right-click the **Menu items** folder, point to **Add**, and then click **New item**.
2. Click **Dynamics 365 Items** > **User Interface** > **Display menu item**. Set the **Name** property to **FMTRental_Current**.
3. Click **Add**.
4. If the new **FMTRental_Current** menu item isn't already open in the designer, double-click it in Solution Explorer.
5. In the **Properties** window, set the following properties.

PROPERTY	VALUE
Label	@FMT197 This value corresponds to "Current rentals".
Object	FMTRental
Query	FMTRental_Current

6. Press **Ctrl+S** to save.

Add a tile

1. In Solution Explorer, in the **FMTutorial** project, right-click the **Tiles** folder, point to **Add**, and then click **New item**.
2. Click **Dynamics 365 Items** > **User Interface** > **Tile**. Set the **Name** property to **FMTCurrentRentalsTile**.
3. Click **Add**.
4. If the new **FMTRental_Current** tile isn't already open in the designer, double-click it in Solution Explorer.
5. In the **Properties** window, set the following properties.

PROPERTY	VALUE
Size	ShortWide
Menu Item Name	FMTRental_Current
Type	Count

6. Press **Ctrl+S** to save.

Tiles also have a refresh frequency property that controls how often the counts on the tiles are automatically updated. The value that is set for this property should be based on the demand for updated counts, together with query execution speed against volume data. For guidance about how to set this property, see Tile and List Caching for Workspaces. For this lab, the default value of 10 minutes will be enough.

Add a tile button to the workspace form

1. In Solution Explorer, double-click the **FmtClerkWorkspace** form to open it in the designer.
2. Right-click **Design** > **PanoramaBody** > **TileContainer**, point to **New**, and then click **Tile Button**.
3. Press **Alt+Up arrow** four times to move the tile button to the top of TileContainer.
4. In the **Properties** window, set the following properties.

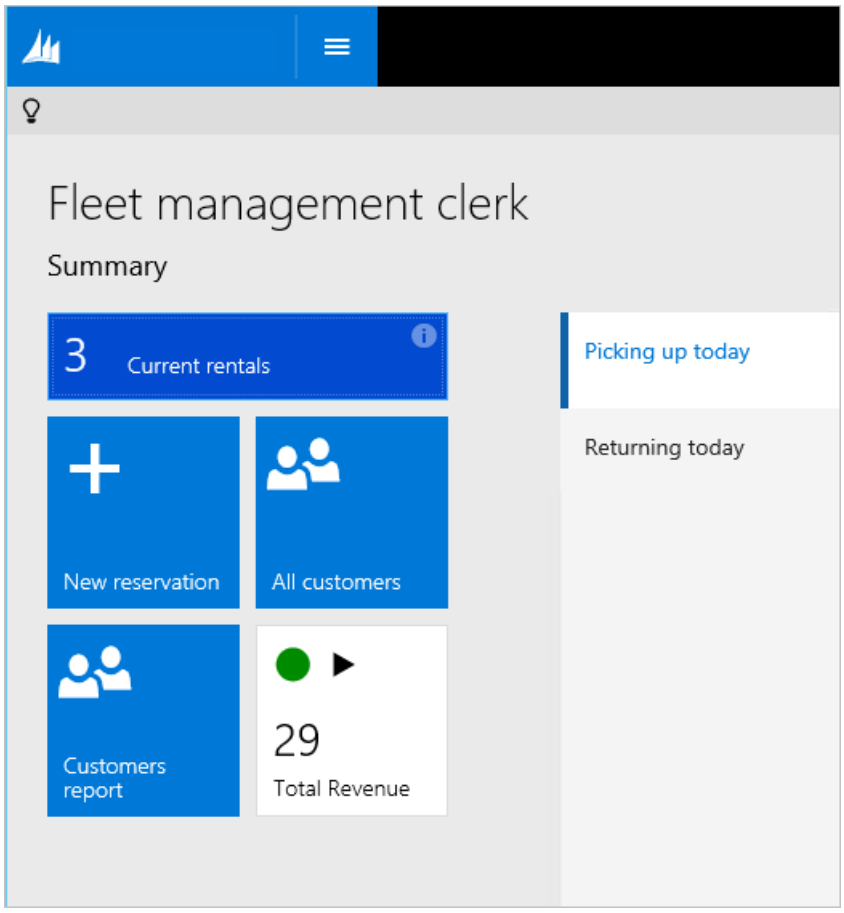
PROPERTY	VALUE
Name	FMTCurrentRentalsTile
Tile	FMTCurrentRentalsTile

5. Press **Ctrl+S** to save.

View the new tile on the workspace

Use Visual Studio to build and run the updated **FmtClerkWorkspace** form.

1. In Solution Explorer, right-click the **FmtClerkWorkspace** form, and then click **Set as Startup Object**.
2. Press **Ctrl+F5** to build and run the form. The form opens in Internet Explorer.

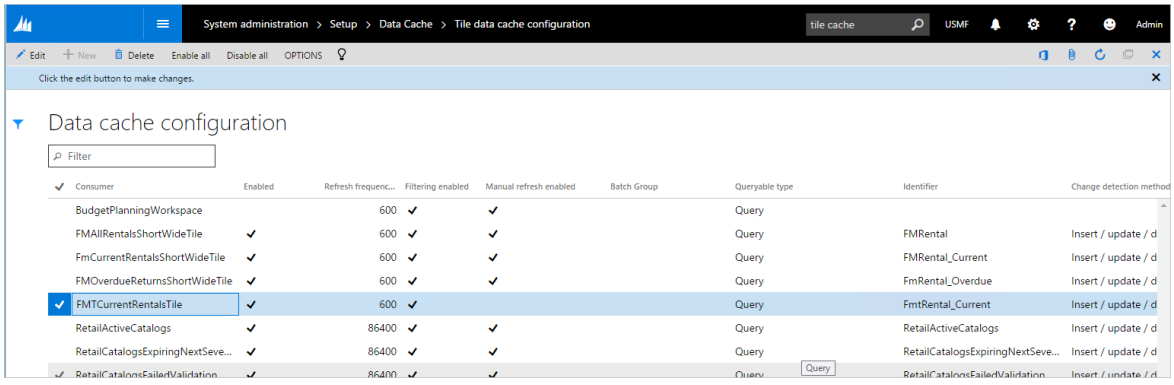


3. Click the **Current rentals** tile. You go to the **Rentals** page, which should be filtered to the three current rentals.
4. Click the **Back** button or the **Close** button to return to the workspace.
5. Click on the small **i** button in the upper-right corner of the **Current rentals** tile. You see information about how current the data in the tile is. Additionally, a link is provided that you can use to manually refresh the tile to view updated data.

View tile data cache values at run time

A system administrator can modify tile cache parameters at run time by using the **Tile data cache configuration** page.

1. Click in the navigation search field on the navigation bar.
2. Type **Tile data**, and then click **Tile data cache configuration** in the search results.
3. Find the **FMTCurrentRentalsTile** record.



From this page, the system administrator can perform several run-time modifications to a tile cache. For example, the system administrator can enable/disable the data cache, modify the refresh frequency, and

enable/disable the ability to manually refresh the count tile. Note that tile caches are registered when a form that has a tile is first opened. Therefore, the list of tiles that is shown in your environment might differ from the list in the preceding illustration.

Exercise 3: Create a new tabbed list in the workspace

Next, you will next see how to include an additional list in the workspace. This section will give you some experience with building forms and will also expose you to form patterns. You will add a list of available vehicles, so that you will be able to initiate a new rental by selecting an available vehicle. In this section, you will just add the new list to the new workspace but won't add the action to initiate the rental. To add this list, you will have to complete the following tasks:

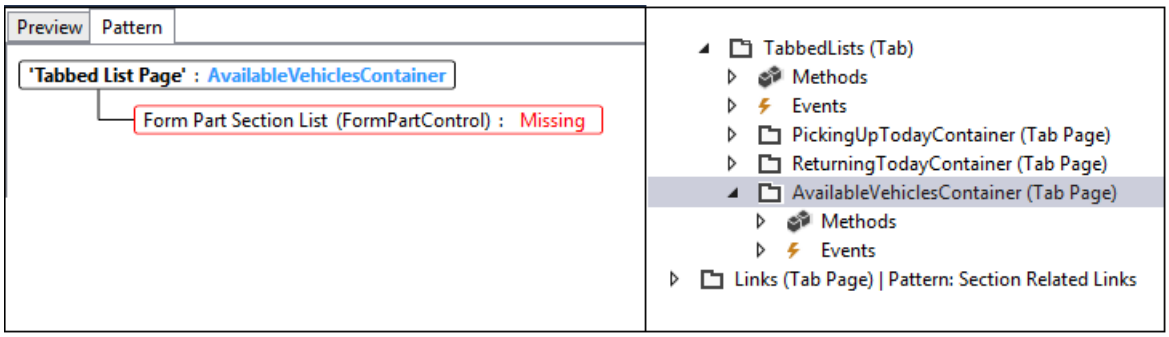
1. Add a new tab page to the workspace.
2. Add a new form that has the list content.
3. Add a new menu item that points to the new form.
4. Add a new query to limit the vehicles to available vehicles.

Add space in the workspace for a new list

1. In Solution Explorer, double-click the **FmtClerkWorkspace** form to open it in the designer.
2. Right-click **Design** > **PanoramaBody** > **TabbedListSection** > **TabbedLists**, and then click **New Tab Page**.
3. In the **Properties** window, set the following properties.

PROPERTY	VALUE
Name	AvailableVehiclesContainer
Caption	@FMT199 This value corresponds to “Available vehicles”.

4. Right-click **AvailableVehiclesContainer**, point to **New**, and then click **Form Part**. **Form Part** is the only control type that the Operational Workspace pattern allows here. This control will be used to link to the form that you will build to hold the content for this section.



5. In the **Properties** window, set the **Name** property to **AvailableVehiclesPart**.
6. Press **Ctrl+S** to save.

Add a new form that has the new workspace content

1. In Solution Explorer, in the **FMTutorial** project, right-click the **Forms** folder, point to **Add**, and then click **New item**.
2. Click **Dynamics 365 Items** > **User Interface** > **Form**. Set the **Name** property to **FMTAvailableVehicles**.
3. Click **Add**.

4. If the new **FMTAvailableVehicles** form isn't already open in the designer, double-click it in Solution Explorer.

5. Add the **FmtVehicle** table as a data source for the form.

   a. Right-click **Data Sources**, and then click **New Data Source**.

   b. Click the new data source node. In the **Properties** window, set the following properties.

PROPERTY	VALUE
Table	FMTVehicle
Name	FMTVehicle **Note:** Be sure to specify the value for the **Table** property first. This property is automatically updated so that it uses the same value.

6. Add the **FmtVehicleModel** table as a second data source for the form.

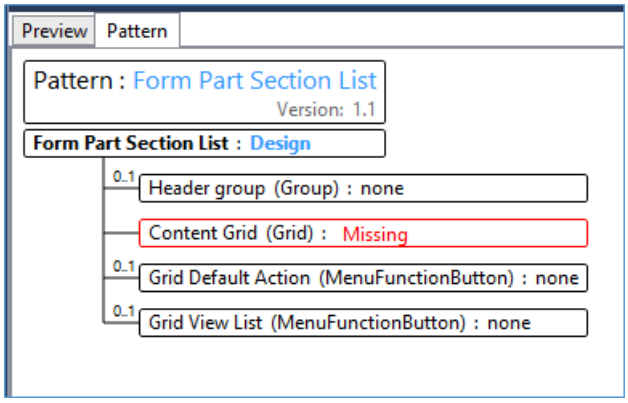
   a. Right-click **Data Sources**, and then click **New Data Source**.

   b. Click the new data source node. In the **Properties** window, set the following properties.

PROPERTY	VALUE
Table	FMTVehicleModel
Name	FMTVehicleModel **Note:** Be sure to specify the value for the **Table** property first. This property is automatically updated so that it uses the same value.
Join Source	FMTVehicle
Link Type	Inner Join

7. Notice the **Pattern: <select>** notation next to **Form Design**. This indicates the required pattern for this node. Right-click **Design**, point to **Apply pattern**, and then click **Form Part Section List**. This form pattern is typically used by workspace lists.

8. Click the **Pattern** tab to see the expected content for this pattern. This information will help guide you as you create content for the form. **Note:** In the future, we plan to provide a mechanism for automatically creating a form structure, based on a selected form pattern.



In particular, this pattern looks for the following elements:

- An optional header group that contains any filters and actions that are required for this workspace list.

- A required grid. As the red border in the preceding illustration indicates, the patterns engine can't currently find this element.
- An optional default action, which can provide navigation to a backing form for an individual record in the grid.
- An optional button that takes the user to a backing form that shows the full list of items in this section.

9. Right-click **Design**, point to **New**, and then click **Grid**.
10. In the **Properties** window, set the following properties. **Note:** The grid that we are building will contain cards and will be two cards wide.

PROPERTY	VALUE
ExtendedStyle	cardList
Multi Select	No
Style	List
Data Source	FMTVehicle
Name	VehicleList
Visible Columns Mode	Fixed
Visible Columns	2

11. Right-click the **VehicleList** grid, point to **New**, and then click **Group**.
12. In the **Properties** window, set the **Name** property to **VehicleCard**.
13. Right-click the **VehicleCard** group, point to **Apply pattern**, and then click **Business Card – Three Fields**.
14. Right-click the **VehicleCard** group, point to **New**, and then click **Image**.
15. In the **Properties** window, set the following properties.

PROPERTY	VALUE
Data Source	FMTVehicleModel
Data Method	vehicleImage

16. Expand **Data Sources** > **FMTVehicle** > **Fields**.
17. Drag the **VehicleId** and **DisplayRelationType** fields into the **VehicleCard** group.
18. Right-click **Design**, point to **New**, and then click **Group**. Update the new group's **Name** property to **HeaderGroup**.
19. The new HeaderGroup element requires a subpattern, because there are two variants for the arrangement of filters and actions in these sections. Right-click **HeaderGroup**, point to **Apply pattern**, and then click **Filters and Toolbar – Inline**.
20. Right-click **HeaderGroup**, point to **New**, and then click **Group**. Update the new group's **Name** property to **FilterGroup**.

21. Right-click **FilterGroup**, point to **New**, and then click **QuickFilter**.
22. In the **Properties** window, set the following properties.

PROPERTY	VALUE
Name	VehicleQuickFilter
Target Control	VehicleList

23. Press **Ctrl+S** to save.

Add a new query that limits the data to available vehicles

1. In Solution Explorer, in the **FMTutorial** project, right-click the **Queries** folder, point to **Add**, and then click **New item**.
2. Click **Dynamics 365 Items** > **Data Model** > **Query**. Set the **Name** property to **FMTAvailableVehicles**.
3. Click **Add**.
4. If the new **FMTAvailableVehicles** query isn't already open in the designer, double-click it in Solution Explorer.
5. In the designer, right-click **Data Sources**, and then click **New Data Source**.
6. In the **Properties** window, set the following properties.

PROPERTY	VALUE
Table	FMTVehicle
Dynamics Fields	Yes

7. Right-click **Ranges**, and then click **New Range**.
8. In the **Properties** window, set the following properties.

PROPERTY	VALUE
Field	Status
Value	Available

9. Press **Ctrl+S** to save.

Add a new menu item that references the new form

1. In Solution Explorer, in the **FMTutorial** project, right-click the **Menu items** folder, point to **Add**, and then click **New item**.
2. Click **Dynamics 365 Items** > **User Interface** > **Display menu item**. Set the **Name** property to **FMTAvailableVehicles**.
3. Click **Add**.
4. If the new **FMTAvailableVehicles** menu item isn't already open in the designer, double-click it in Solution Explorer.

5. In the **Properties** window, set the following properties.

PROPERTY	VALUE
Object	FMTAvailableVehicles
Query	FMTAvailableVehicles

6. Press **Ctrl+S** to save.

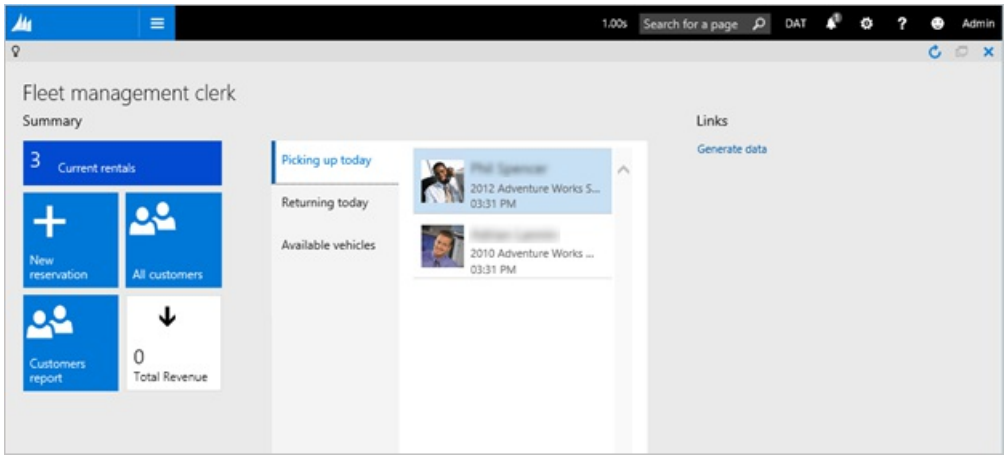
Link the new list to the workspace

1. In Solution Explorer, double-click the **FmtClerkWorkspace** form to open it in the designer.
2. Click **Design** > **PanoramaBody** > **TabbedListSection** > **TabbedLists** > **AvailableVehiclesContainer** > **AvailableVehiclesPart**.
3. In the **Properties** window, set the **Menu item name** property to **FmtAvailableVehicles**.

View the new menu item

Use Visual Studio to build and run the updated **FmtClerkWorkspace** form.

1. In Solution Explorer, right-click the **FmtClerkWorkspace** form, and then click **Set as Startup Object**.
2. Press **Ctrl+F5** to build and run the form. The form opens in Internet Explorer.



3. Click the **Available vehicles** tab to see the new list.
4. Click in the QuickFilter, type **Lit**, and then press **Enter** to filter down to Litware model vehicles that are available.

Exercise 4: Create a backing data cache for a list

For lists that have expensive queries, or lists where multiple users might use and work from the same workspace, consider caching the list to improve performance. In this section, you will add the necessary artifacts to create a data cache for a list. These artifacts include a query that maps fields to the cache, a table that holds the cache, and a class that provides the mapping between the query and the table. You will then uptake this cache on one of the tabbed lists in the workspace.

Add a query that maps fields to the data cache

The first step is to build a query that will be used to populate the cache table. This query should include all the tables that you want to get your cache data from, and it should limit the results to those records/columns that you want cached.

1. In Solution Explorer, in the **FMTutorial** project, right-click the **Queries** folder, point to **Add**, and then click **New item**.

2. Click **Dynamics 365 Items** > **Data Model** > **Query**. Set the **Name** property to **FMTPickupAndReturnQuery**.
3. Click **Add**.
4. If the new **FMTPickupAndReturnQuery** query isn't already open in the designer, double-click it in Solution Explorer.
5. In the designer, right-click **Data Sources**, and then click **New Data Source**.
6. In the **Properties** window, set the following properties.

PROPERTY	VALUE
Table	FMTRental
Dynamics Fields	No

7. Add the following five fields to the FMTRental data source. For each field, right-click **Fields**, point to **New**, and then click **Field**. In the **Properties** window, set the **Field** property as appropriate.
   - StartDate
   - EndDate
   - Vehicle
   - State
   - RentalId
8. Right-click **Ranges**, and then click **New Range**.
9. In the **Properties** window, set the following properties.

PROPERTY	VALUE
Field	State
Value	1..2 **Note:** This value will cache rentals that are Ready for Pickup or In Progress.

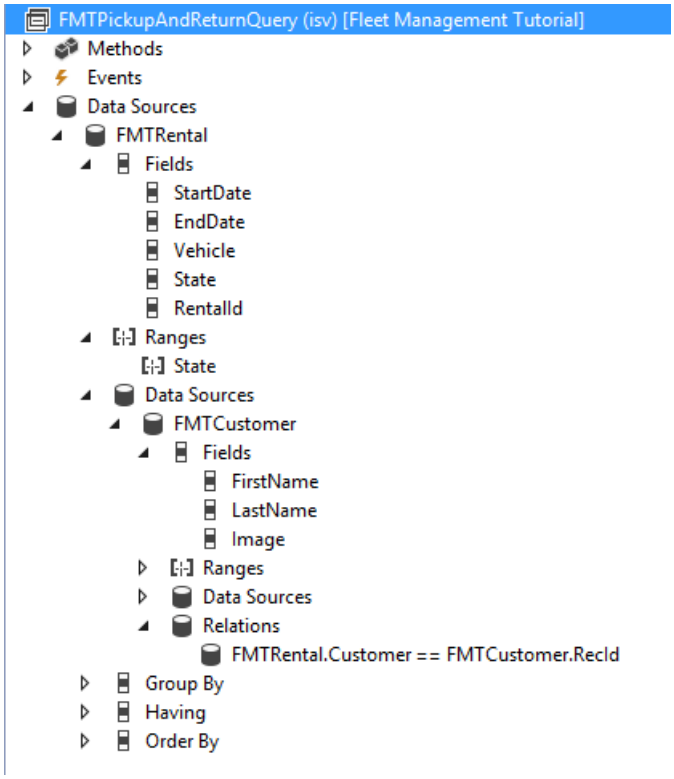
10. Under **FMTRental**, right-click **Data Sources**, and then click **New Data Source**.
11. In the **Properties** window, set the following properties.

PROPERTY	VALUE
Table	FMTCustomer
Dynamics Fields	No

12. Add the following three fields to the FMTCustomer data source. For each field, right-click **Fields**, point to **New**, and then click **Field**. In the **Properties** window, set the **Field** property as appropriate.
    - FirstName
    - LastName
    - Image
13. Right-click **Relations**, and then click **New Relation**.
14. In the **Properties** window, set the following properties.

PROPERTY	VALUE
Join Data Source	FMTRental
Field	Customer
Related Field	RecID

The query that you've constructed should match the following illustration.



15. Press **Ctrl+S** to save.

Add a cache table

The second step is to define a table that has the fields that are returned from the cache query. You must also add a **SysDataCacheContextId** field that will be used to map the cache row to the base framework cache tables. Additionally, you should also define any required relations between this table and other tables, and also any data methods that you require that involve the cached fields.

1. In Solution Explorer, in the **FMTutorial** project, right-click the **Tables** folder, point to **Add**, and then click **New item**.
2. Click **Dynamics 365 Items** > **Data Model** > **Table**. Set the **Name** property to **FMTPickupAndReturnTableCache**.
3. Click **Add**.
4. If the new **FMTPickupAndReturnTableCache** table isn't already open in the designer, double-click it in Solution Explorer.
5. In the designer, right-click **Fields**, and then add the following fields. For each field, the following table shows the data type and the extended data type (EDT) or enum type.

FIELD TYPE	FIELD NAME	EDT/ENUM TYPE
String	FirstName	FirstName (EDT)
String	LastName	LastName (EDT)
Container	Image	Bitmap (EDT)
Int64	Vehicle	FMTVehicleRecId (EDT)
Utc Date Time	StartDate	StartDateTime (EDT)
Utc Date Time	EndDate	EndDateTime (EDT)
Int64	SysDataCacheContextId	SysDataCacheContextId (EDT)
Enum	State	FMTReservationState (Enum)
String	RentalId	FMTRentalId (EDT)

6. In the designer, right-click **Relations**, point to **New**, and then click **Relation**.
7. In the **Properties** window, set the following properties.

PROPERTY	VALUE
Name	FMTRental
Related Table	FMTRental

8. Right-click the **FMTRental** relation, point to **New**, and then click **Normal**.
9. In the **Properties** window, set the following properties.

PROPERTY	VALUE
Field	RentalId
Related Field	RentalId

10. In the designer, right-click **Mappings**, and then click **New**.
11. In the **Properties** window, set the **Map** property to **SysDataSetCacheTableMap**.
12. Click **Id**. In the **Properties** window, set the **Map Field To** property to **RecId**.
13. Click **SysDataCacheContextId**. In the **Properties** window, set the **Map Field To** property to **SysDataCacheContextId**. **Note:** If the field doesn't appear in the list, you might have to save the table first, by pressing **Ctrl+S**.
14. Press **F7** to view the table's code. Alternatively, right-click **FMTReturnAndPickupTableCache**, and then click **View Code**.
15. Add the following display methods to the table. The form will use these methods later.

```
public display FMTName fullName()
{
    return this.FirstName + ' ' + this.LastName;
}
public display container customerImage()
{
    ImageReference imgRef;
    container imgContainer = this.Image;
    if(imgContainer == connull())
        {
            imgRef = ImageReference::constructForSymbol("Person");
            imgContainer = imgRef.pack();
        }
        return imgContainer;
}
public display str rentalVehicle()
{
    FMTVehicle vehicle;
    str value;
    if(this.Vehicle == 0)
    {
        value = "No vehicle assigned";
    }
    else
    {
        select vehicle where vehicle.RecId == this.Vehicle;
        value = vehicle.Description;
    }
    return value;
}
```

16. Press **Ctrl+S** to save.

Adding a cache class that links the query and table

The third step is to create a class that defines the relationship between the cache query and the catch table.

1. In Solution Explorer, in the **FMTutorial** project, right-click the **Classes** folder, point to **Add**, and then click **New item**.
2. Click **Dynamics 365 Items** > **Code** > **Class**. Set the **Name** property to **FMTPickupAndReturnClass**.
3. Click **Add**.
4. If the new **FMTPickupAndReturnClass** class isn't already open in the designer, double-click it in Solution Explorer.
5. Add the following code to the class.

```
[SysDataSetExtension(classStr(FMTPickupAndReturnClass)), // The name of this class
SysDataSetCacheTableExtension(tableStr(FMTPickupAndReturnTableCache))] // The name of the cache table
class FMTPickupAndReturnClass extends SysDataSetQuery implements SysIDataSet
{
    public SysDataCacheRefreshFrequency parmRefreshFrequency()
    {
        return 600; // Cache refresh frequency, in seconds.
    }
    public SysQueryableIdentifier parmQueryableIdentifier()
    {
        return queryStr(FMTPickupAndReturnQuery); // The name of the query.
    }
    public SysDataCacheTypeId parmCacheTypeId()
    {
        return tableNum(FMTPickupAndReturnTableCache); // The name of the table.
    }
    public static FMTPickupAndReturnClass construct()
    {
        return new FMTPickupAndReturnClass();
    }
}
```

6. Press **Ctrl+S** to save.

Uptake the data cache on a list in the workspace and table

After you've set up the data cache, you can start to use the cache in your forms. In this section, you will update one of the workspace lists so that it uses the data cache.

1. In Solution Explorer, double-click the **FMTReturningTodayPart** form to open it in the designer.
2. Expand the **Data Sources** node.
3. Delete the **FMTCustomer** data source.
4. Click the **FMTRental** data source. In the **Properties** window, set the **Table** property to **FMTPickupAndReturnTableCache**.
5. Click **Design** > **ReturningTodayGrid**. In the **Properties** window, set the **Data Source** property to **FMTPickupAndReturnTableCache**.
6. Inside **ReturningTodayGrid**, click **CustomerImage**. Update the **Data Source** property to **FMTPickupAndReturnTableCache** and the **Data Method** property to **customerImage**.
7. Inside **ReturningTodayGrid**, click **FirstNameCopy1**. Update the **Data Source** property to **FMTPickupAndReturnTableCache** and the **Data Method** property to **fullName**.
8. Press **F7** to view code for the form.
9. Instrument the form so that it can react to data caching, as shown in the following code.

```
[Form]
public class FMTReturningTodayPart extends FormRun implements SysIDataSetConsumerForm
{
    public void registerDatasourceOnQueryingEvent()
    {
        FMTPickupAndReturnTableCache_DS.OnQueryExecuting +=
            eventhandler(this.parmDataSetFormQueryEventHandler().prepareDataSet);
    }
}
```

10. Press **Ctrl+S** to save.

Update the action above the list so that it works with the cache table

Actions that are performed from the workspace might expect records from the base tables. Therefore, these actions might have to be updated so that they work with the cache table. In this example, the **FmtCompleteRecord** form currently expects a **FMTRental** record as context. Therefore, this form must be updated so that it works correctly with either a base rental record or a cache table record as context.

1. In Solution Explorer, double-click the **FmtCompleteRental** form to open it in the designer.
2. Press **F7** to view the form's code.

```
public void init()
{
    //If this form was opened with a Rental as context
    if(element.args() != null && element.args().record() != null && element.args().record().TableId
== tablenum(FMTRental))
    {
        //Get the Rental context
        rentalDS = FormDataUtil::getFormDataSource(element.args().record());
        rental = element.args().record();
        if(rental != null)
        {
            select firstonly forupdate vehicle where vehicle.RecId == rental.Vehicle;
        }
    }
    super();
}
```

3. Update the **init()** method so that it matches the following code.

```
public void init()
{
    //If this form was opened with a record context
    if(element.args() != null && element.args().record() != null))
    {
        //Get that context
        rentalDS = FormDataUtil::getFormDataSource(element.args().record());
        if(element.args().record().TableId == tableNum(FMTPickupAndReturnTableCache))
        {
            FMTPickupAndReturnTableCache cacheRecord = element.args().record();
            select firstonly forupdate rental where rental.RentalId == cacheRecord.RentalId;
        }
        else if(element.args().record().TableId == tableNum(FMTRental))
        {
            rental = element.args().record();
        }
        if(rental != null)
        {
            select firstonly forupdate vehicle where vehicle.RecId == rental.Vehicle;
        }
    }
    super();
}
```

4. Press **Ctrl+S** to save.

Update the Returning today query so that it works with the cache table

1. In Solution Explorer, double-click the **FmtRental_ReturningToday** query to open it in the designer.
2. Expand the **Data Sources** node, and then click **FMTRental**.
3. In the **Properties** window, update the **Table** property to **FMTPickupAndReturnTableCache**. **Note:** The **Name** property should be updated automatically to the same value.
4. Press **Ctrl+S** to save.

View the updated query

Use Visual Studio to build and run the updated **FmtClerkWorkspace** form.

1. In Solution Explorer, right-click the **FmtClerkWorkspace** form, and then click **Set as Startup Object**.
2. Press **Ctrl+F5** to build and run the form. The form opens in Internet Explorer.
3. Click the **Returning today** vertical tab.
4. Click **Complete rental** for the second record in the list.
5. Set **End Mileage** to **200**, and then click **OK**. Notice that the rental that you just returned still appears in the list.

Make sure that your workspace is responsive

You must make sure that your lists remain up to date after a user performs an action that should remove a record from the list (for example, the user completes a rental). In this section, we will instrument that action to help guarantee that the workspace reacts appropriately.

1. In Solution Explorer, double-click the **FmtCompleteRental** form to open it in the designer.
2. Press **F7** to view the code for the form.
3. Locate the **clicked()** code for **OKButton**. Near the end of this method is a research call on the calling form's data source. Just before that line of code, add the following **if** statement to delete the processed rental from the cache table.

```
. . .
if(rentalDS.table() == tableNum(FMTPickupAndReturnTableCache))
{
    //Delete updated record from backing cache
    FMTPickupAndReturnTableCache cacheRecord = element.args().record();
    cacheRecord.delete();
}
rentalDS.research(true);
}
```

4. Press **Ctrl+S** to save.

View the updated form

Use Visual Studio to build and run the updated **FmtClerkWorkspace** form.

1. In Solution Explorer, right-click the **FmtClerkWorkspace** form, and then click **Set as Startup Object**.
2. Press **Ctrl+F5** to build and run the form. The form opens in Internet Explorer.
3. Click the **Returning today** vertical tab.
4. Click **Complete rental** for the second record in the list.
5. Set **End Mileage** to **100**, and then click **OK**. Notice that the rental that you just returned no longer appears in the list.

Related tutorials

- Build the Customer form – See this tutorial if you want more exposure to form patterns. This tutorial will walk through the process of applying the Details Master pattern to a form.
- Build navigation – See this tutorial if you want instructions for adding your workspace to the menu structure.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Navigation concepts

2/18/2021 • 7 minutes to read • Edit Online

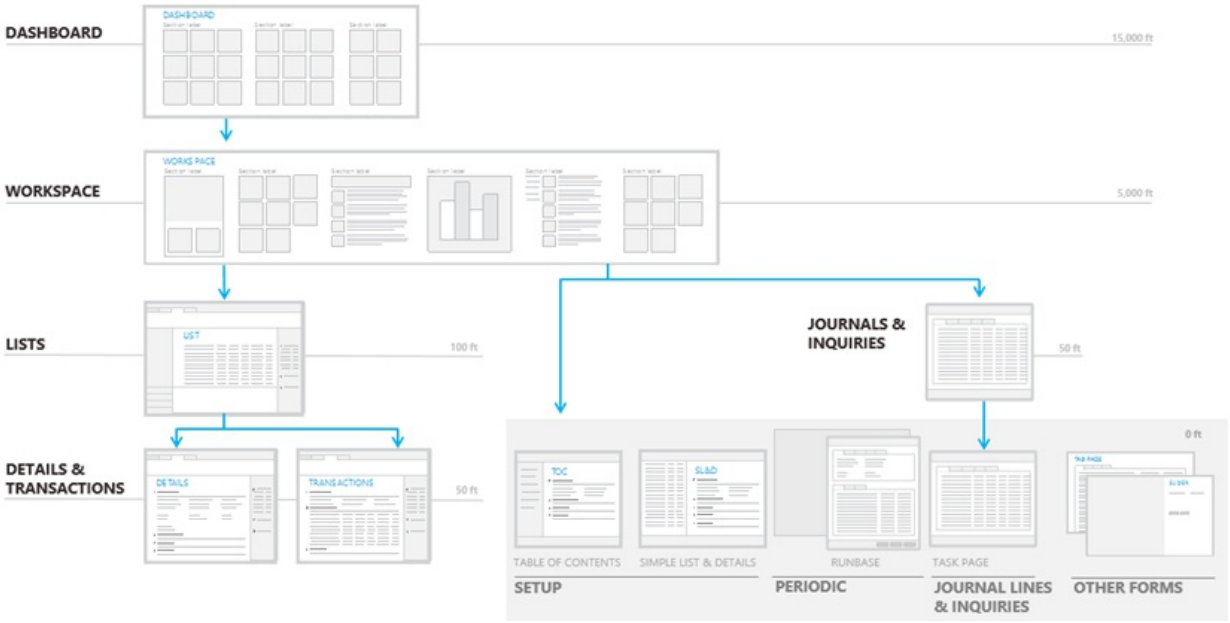
This articles describes the primary navigation concepts including the dashboard, the new navigation search feature, the navigation pane, workspaces, and tiles.

Navigation concepts

The primary navigation concepts are:

- Dashboard
- Navigation pane
- Workspaces
- Tiles
- Navigation search

The dashboard is a new concept, whereas the navigation pane and workspaces are updates to existing concepts. To implement the navigation concepts, the user interface model uses several standard page types. When you create an application, you should follow the conventions for these pages to present a consistent experience for the user. The following diagram shows an overview of the standard page types and how they fit together.



The following sections provide more detail about the pages that underlie these concepts. They include information about the modeling of these and other types of pages.

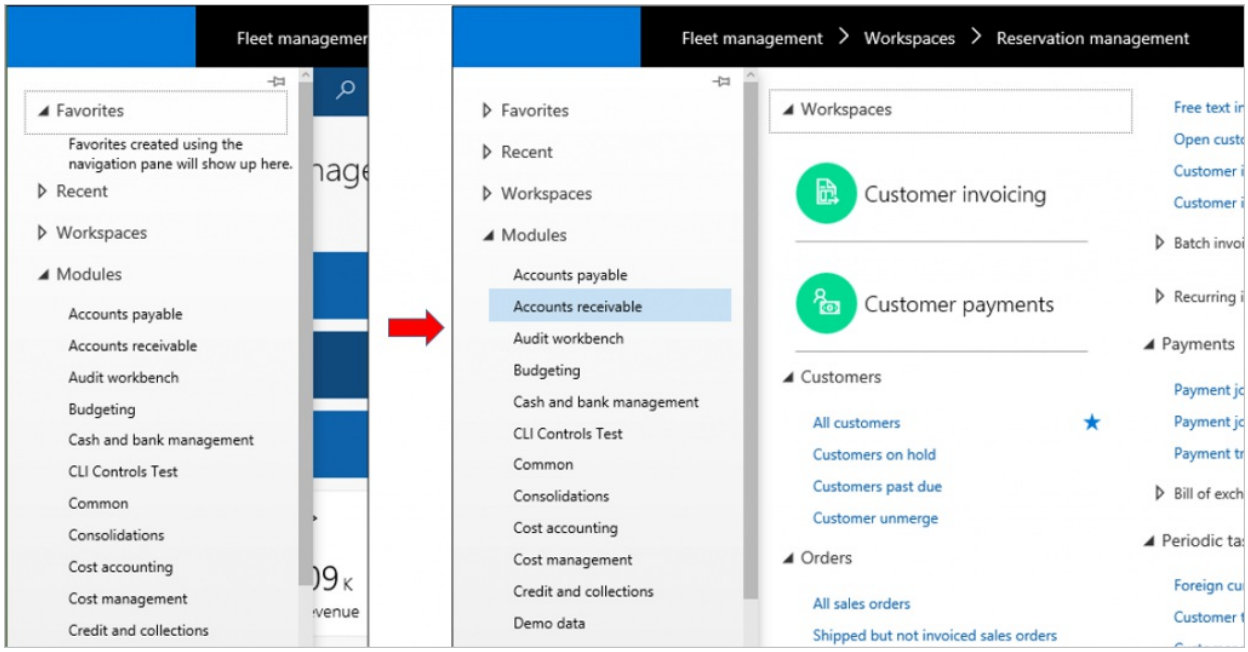
Dashboard

The dashboard is the first page that users see when they access the client. The dashboard contains tiles that show important details from the system. Content that was previously displayed in Cues on Role Center pages in Microsoft Dynamics AX 2012 is now available on the dashboard. You can return to the dashboard at any time by clicking **Dynamics 365** on the navigation bar at the top of the application frame.

The dashboard primarily consists of a large section of workspace tiles. There might also be a Getting Started tool, which isn't shown in the preceding screenshot. The dashboard's workspace tile section is built from a menu structure that has its root in the **NavPaneMenu** menu. The menu is modified by a set of menu extensions, and those extensions contain one or more tile references that correspond to the tiles that users see in that section.

Navigation pane

The navigation pane provides access to workspaces, main menu elements, recently opened forms, and user-defined favorites. The user can open the navigation pane by clicking the **Show navigation pane** button under the navigation bar. The navigation pane consists of four collapsible sections. The **Favorites** section provides quick access to the list of forms the user has explicitly marked as a favorite. Marking a form as a favorite is accomplished by clicking the star icon next to the form in the navigation pane. The **Recent** section lists the forms the user has most recently visited. The set of workspaces a user has access to is conveniently shown in the **Workspaces** section. Finally, the **Modules** section provides the full list of modules. Clicking on a module will open the right side of the navigation pane, where the user can navigate to the desired page in that module.
Note: In this screenshot, **All customers** has been marked as a favorite, and therefore it will appear in the **Favorites** list.



Like the workspace tiles on the dashboard, the elements that are listed in the navigation pane are generated at runtime, based on a menu structure. The same root menu (**NavPaneMenu**) that defines the set of workspaces on the dashboard also defines the navigation pane. Here's an example of the logical structure for the navigation pane:

- NavPaneMenu (menu)
  - NavPaneMenuFleet (menu extension)
    - "Reservation management" (tile reference)

Workspaces

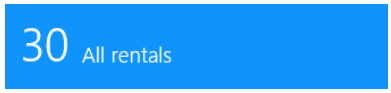
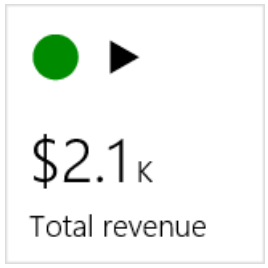
Workspaces are activity-oriented pages that are designed to increase a user's productivity by providing information that answers the targeted user's most pressing activity-related questions and allows the user to initiate their more frequent tasks. Access to the various workspaces depends on the roles that users have in the organization. Much of the list and business intelligence (BI) content from the old Role Center pages is exposed on workspaces. To navigate to a workspace, you can click a tile on the dashboard, click a link in the navigation pane, or find the workspace using the navigation search feature (see the next section).

Workspaces are simply a different type of modeled forms. The system differentiates workspaces by **Form.Design.Style**=**Workspace** (this value is defined as part of the Workspace form pattern). A workspace consists of a caption (which is defined on **Form.Design.Caption**) and a panorama (that is, a tab control). A panorama contains sections of content that are relevant to the task for which the workspace is intended. These sections are modeled as tab pages. The first section will generally be a set of tiles that users can click to begin

new tasks or access lists of items. The second section contains a set of relevant lists for the activity. The last section contains a number of links to pages that are important but not frequently used for this activity. In between the list and links section are a few optional sections that might contain charts and graphs. One important distinction of workspaces is that they do not have a data source. If the content (such as a list or chart) requires a data source, you must model that content on an independent form of type **Form part**, and then reference that form part on the workspace. Form parts can be hosted on other forms, and each can have its own data source.

Tiles

Windows 8 introduced the concept of tiles, and you will see them used in the client. A tile is a rectangular button that behaves like a menu item button. It is used to navigate to or open pages. In addition, tiles can display relevant data, such as counts or key performance indicators (KPIs). A tile can include images that provide the user with additional visual context. You can create the following types of tiles.

TYPE	EXAMPLE	DESCRIPTION
Standard		This type of tile does not show any business data.
Count		This type of tile shows a count of the items in the referenced form's query. Note that the tile in this example uses the **ShortWide** size.
KPI		This type of tile shows a summary of data from a KPI.
Link		This type of tile points to a URL. The tile has the same appearance as a tile of the **Standard** type.

For modeling, you first create a tile element, which is an abstract element. The tile element is then referenced on a form by a tile button element. Multiple tile button elements can refer to a single tile. Tiles are defined first by the **Type** property (which has valid values of **Standard**, **Count**, **KPI**, and **Link**). After you specify the type, you specify the following minimum properties for each type:

- **Standard** and **Count**:
  - Label
  - Menu Item Name

    - Menu Item Type
- **KPI**:
    - Label
    - KPI
- **Link**:
    - Label
    - URL

After these properties are defined, your tile is complete. To make the user experience richer, you can optionally extend tiles of the **Standard** and **Link** types so that they include images. Use the following properties to add images.

PROPERTY	DESCRIPTION
Image Location	See the information about basic button behavior in Actions.
Normal Image	See the information about basic button behavior in Actions.
Tile Display	Define how the image appears on the tile.

The following are the valid options and corresponding behaviors for the **Tile Display** property.

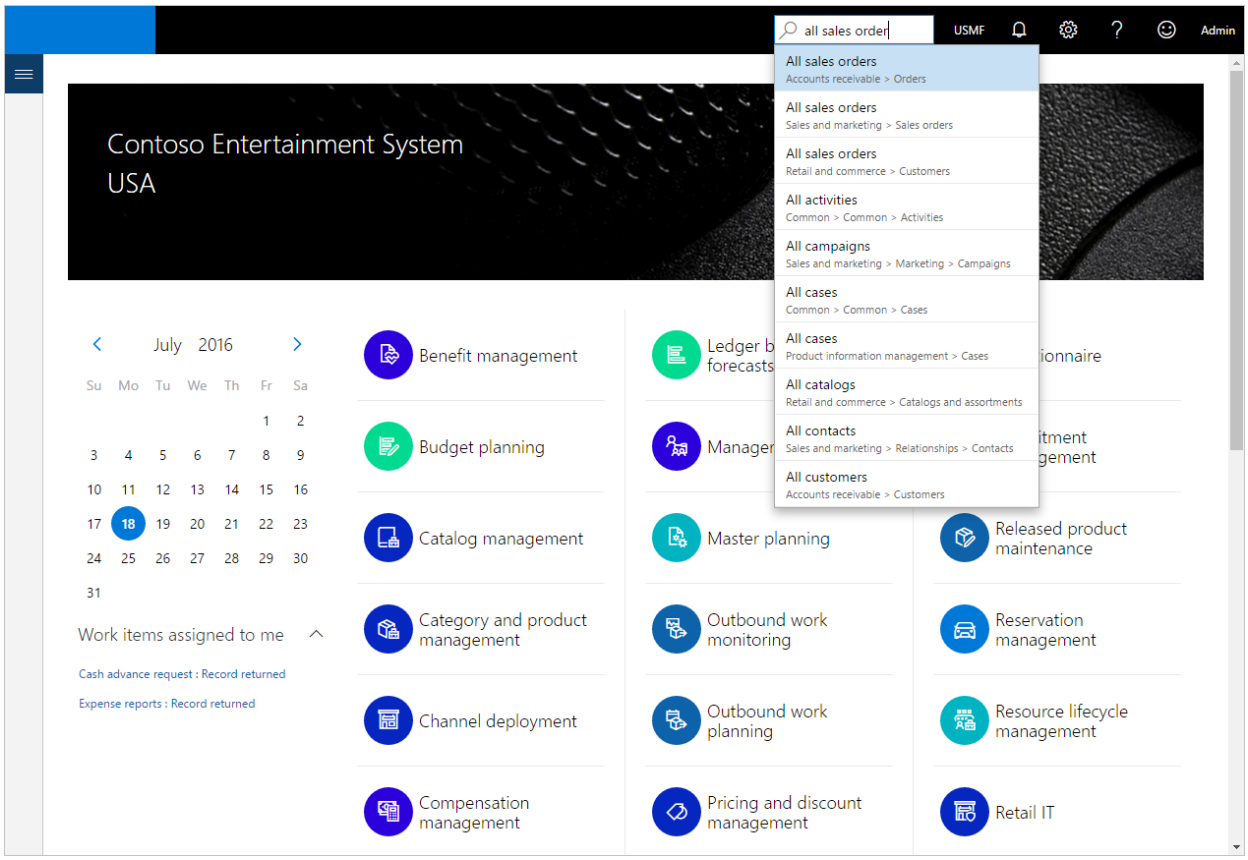
VALUE	DESCRIPTION
Auto	The behavior is the same as the behavior for **TextAndImage**.
TextAndImage	The tile shows the specified label and the image in a small container above it. For example, the **App links** tile uses this value.
TextOnly	The other two image-related properties are ignored, and only the label is shown.
ImageOnly	The label is ignored, and only the image is shown.
BackgroundImage	The specified image is expanded to fill the tile from edge to edge, and the label is overlaid on the image in the same location. Shading is applied behind the text to ensure that it remains legible.

Note: The following limitations apply when the **BackgroundImage** value is used:

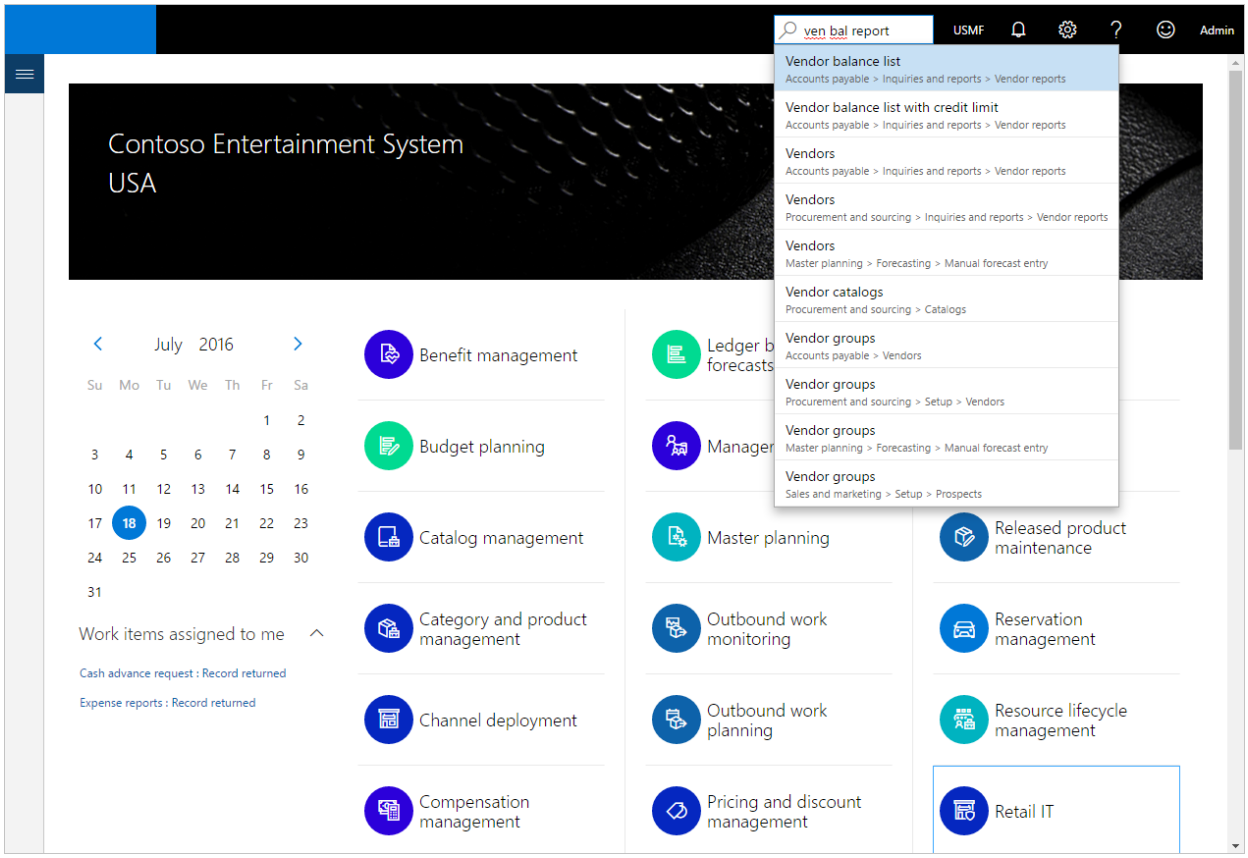
- If a symbol is used, it isn't displayed.
- The image that you specify isn't resized. Therefore, you must create an image of the appropriate size to guarantee that it fills the tile correctly. Currently, a standard-sized tile is a square that is 130 pixels on each side.

Navigation search

There is a convenient search mechanism for finding and navigating to forms and workspaces that appear in the navigation pane and on the dashboard. For example, a search on the keywords "all sales order" returns a list of navigation elements that match those keywords.



The search keywords are matched not only to the caption of the navigation elements but also to the corresponding path. For example, a search on the keywords "ven bal report" returns results that match "vendor balance" in the caption and "report" in the path.



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Page layout in the web client

2/18/2021 • 7 minutes to read • Edit Online

This topic discusses layout in the web client. Layout is a design process that specifies how controls appear on a page.

Introduction

Layout is a design process that specifies how the controls on a page appear in the web client. Layout occurs within container controls. The following table lists the container controls.

CONTAINER	DESCRIPTION
Form.Design	The root of the page. It functions as a special kind of container.
Group	The general-purpose container control. Group controls can be nested as required.
Tab	A control that contains TabPage controls and has many possible **Tab.Style** values, such as **Tab**, **FastTab**, **Vertical Tab**, and **Panorama**.
TabPage	The appearance of each TabPage control depends on its **Tab.Style** value.
ButtonGroup	A special type of Group control that contains buttons.

A grid is a special type of control that has some container behaviors, such as flexible sizing (**SizeToAvailable**). However, a grid has special visualizations and isn't a general-purpose container control.

Layout: Dynamics AX 2012 vs. Finance and Operations apps

Layout in Dynamics AX 2012

In Microsoft Dynamics AX 2012, the arrangement of controls in containers is almost always vertical, and columns are manually set to provide some horizontal spread.

Examples

Columns=1 1 2 3 **Columns=2** 1 4 2 5 3 In Dynamics AX 2012, sizing is achieved via the **Height** and **Width** properties. If **Height** and **Width** are set to **Auto**, the size is as large as the child controls require. If **Height** and **Width** are set to **Column**, the container is as large as it can be within the parent container. By default, **Height** and **Width** are set to **Auto** for every container.

Layout in Finance and Operations

In Finance and Operations, layout is controlled by the same basic properties that control layout in Dynamics AX 2012. However, additional options have been added to support a more responsive layout. In particular, the layout of a page is based on the following factors:

- The arrangement method that is specified by the **ArrangeMethod** property.
- The columns that are specified by the **Columns** property.
- The sizing that is specified by the **HeightMode**, **WidthMode**, **Height**, and **Width** properties.

ArrangeMethod property

The **ArrangeMethod** property specifies a base arrangement method for a container. For this property, Finance and Operations apps have all the options from AX 2012. However, they also have a **HorizontalWrap** option that is intended for tile layouts in panoramas. The following table describes the various options for the **ArrangeMethod** property.

OPTION	DESCRIPTION
Vertical	Controls are arranged vertically. If columns are also used, controls are arranged vertically inside the generated columns. This option is the default value for Groups and for TabPages where **Tab.Style** is set to a value other than **Panorama**.
HorizontalLeft	Controls are arranged horizontally, and they are left-aligned and bottom-aligned inside the parent container.
HorizontalRight	Controls are arranged horizontally, and they are right-aligned and bottom-aligned inside the parent container.
HorizontalWrap	Controls are arranged inside columns of fixed width that wrap horizontally. This option is typically used for tile layouts in panorama sections. It's the default value for TabPages where **Tab.Style** is set to **Panorama**.

ColumnsMode property

For the **ColumnsMode** property, Finance and Operations apps have a **Fill** option to support responsive layouts. When the property is set to this value, columns automatically flow as required. The following table describes the various options for the **ColumnsMode** property.

OPTION	DESCRIPTION
Fill	Columns are generated to fill the available horizontal space or vertical space, depending on the container type. If the container is a Panorama-style tab, this option generates columns to fill it along the vertical axis. For all other containers (Groups, Tab-style Tabs, and all other styles of Tabs), this option generates columns to fill the container along the horizontal axis.
Fixed	Specify the number of columns that the **Columns** property should generate. Controls are evenly distributed among the columns, and their order is maintained. If the controls can't be distributed evenly among the columns, the leftmost columns receive extra controls first. This option is the default value for all controls.

HeightMode/WidthMode properties

In Finance and Operations apps, sizing is done via two pairs of size properties: **WidthMode** and **Width**, and **HeightMode** and **Height**. The following table describes the various options for these properties.

OPTION	DESCRIPTION
SizeToAvailable	Fill the available space along the vertical (or horizontal) axis inside the parent container. If the parent container has **SizeToContent** height (or width), the child's height (or width) is also **SizeToContent**, unless there is a sibling in the container that can provide a height (or width). This option is the default value for Grids and Tabs (of all styles).
SizeToContent	The height (or width) of the container should be the height (or width) of its contents. This option is the default value for Groups and all other controls except Tabs. FastTabs that aren't always expanded also have **SizeToContent** height.
Manual	The height (or width) is manually sized. Set **HeightMode** (or **WidthMode**) to **Manual**, and then set **Height** (or **Width**) to a fixed number of pixels. **Note:** Microsoft doesn't recommend that you use manual heights and widths, because they don't adapt to changes in form density.

Note that if a value of **Auto** is used for these properties, the behavior is automatically determined at runtime. Typically, a value of **Auto** for these properties causes the same behavior as a value of **SizeToContent**, as in AX 2012.

Interactions between the ArrangeMethod and Columns properties

If **ArrangeMethod**=**HorizontalLeft** or **HorizontalRight**, the **Columns** property has no effect, because items are laid out in strict horizontal arrangement and no wrapping is used. If **ArrangeMethod**=**Vertical**, columns are arranged vertically, and the controls are either distributed evenly among the columns (**Fixed**), or distributed to fill the available horizontal or vertical space (**Fill**). If **ArrangeMethod**=**HorizontalWrap**, columns are arranged, and horizontal wrapping is used at a fixed column width of 280 px. Typically, this option is used to wrap tile layouts.

Examples

ArrangeMethod=HorizontalWrap

ArrangeMethod=**HorizontalWrap** and **Columns**=**1**

1 2
3 4
5 6
7 8
9

ArrangeMethod=**HorizontalWrap** and **Columns**=**2**

1 2	6 7
3 4	8 9

5	

ArrangeMethod=HorizontalWrap and **Columns=Fill**

For this example, we assume that only three lines of items can fit in the container height.

1 2	7 8	13 14
3 4	9 10	15
5 6	11 12	

ArrangeMethod=Vertical

ArrangeMethod=Vertical and **Columns=1**

1
2
3
...

ArrangeMethod=Vertical and **Columns=2**

1	5
2	6
3	7
4	8

ArrangeMethod=Vertical and **Columns=Fill**

For this example, we assume that only three lines of items can fit in the container height.

1	4	7
2	5	8
3	6	

ArrangeMethod=Vertical and **Columns=Fill** on a FastTab

For this example, we assume that the width of the FastTab can fit four columns.

1	3	5	7
2	4	6	8

Breakable groups

When you set **ColumnsMode** to **Fill** to dynamically create columns, based on the amount of available space, groups of fields can be split into multiple columns. The **Breakable** property on Group controls lets developers ensure that controls in a group aren't distributed across columns. The default value for this property is **Yes**, which indicates that the contents of the group can be split between groups. To keep a group together all the time, set **Breakable** to **No**. Note that **Breakable** applies only to the first level in nested groups.

Guidelines for using layout properties

ColumnsMode=Fill

- Don't nest containers that have **ColumnsMode** set to **Fill**. Set **ColumnsMode** to **Fill** only on the direct parent container of the controls/fields that you want to responsively fill the available space.
- Don't set **HeightMode** to **SizeToAvailable** on any child controls of a container that has **ColumnsMode** set to **Fill**. **ColumnsMode=Fill** tries to calculate an average height of all controls across columns to balance them as much as possible. However, our layout CSS and calculations can't handle **SizeToAvailable** children, and this setting doesn't necessarily make sense.
- Don't set **WidthMode** to **SizeToAvailable** on any child controls of a container that has **ColumnsMode** set to **Fill**. Otherwise, the child controls will take up all the available width, and all controls will appear in one column.
- Container that have **ColumnsMode=Fill** should have **WidthMode** set to **SizeToAvailable** (if you're using fill-width containers such as Tabs and Groups) or **HeightMode** set to **SizeToAvailable** (if you're using fill-height containers such as Panorama sections).

HeightMode/WidthMode=SizeToAvailable

- If you use **WidthMode=SizeToAvailable**, make sure that parent containers in the form have **WidthMode** set to **SizeToAvailable**, not **SizeToContent**. **SizeToAvailable** containers inside **SizeToContent** containers are overridden and become **SizeToContent** containers.

Additional resources

User interface development home page

Dynamics Symbol font

2/18/2021 • 2 minutes to read • Edit Online

The Dynamics Symbol font defines the set of out-of-box symbols that are available in the product. These symbols are primarily used for buttons, tiles, and image controls. In every release, there might be updates to this font. For example, symbols might be added or removed.

To access the list of available symbols (the name and an image) for every release that updated the Symbol font, visit the Dynamics Symbol Font page. A description of the various locations where symbols are used in the product and usage guidelines for each location is also included.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Saved views

2/18/2021 • 20 minutes to read • Edit Online

IMPORTANT

Some or all of the functionality noted in this topic is available as part of a preview release. The content and the functionality are subject to change. For more information about preview releases, see Service update availability.

Introduction

Personalization plays an important role in allowing users and organizations to optimize the user experience to meet their needs. For more details on personalization, see Personalize the user experience.

Traditional personalization let users have only one set of personalizations per page. The **Saved views** feature expands on personalization in several important ways:

- Views permit users to have multiple named sets of personalizations per form, which they can quickly switch between as needed. This allows a user to create multiple optimized views of a page, where each view has been tailored to fit the needs of performing a particular business task.
- Views created for particular page types can also include user-added filters or sorts, which allows users to quickly return to commonly filtered datasets. See the What pages support views section for more details.
- Views can be published to users in specific security roles and specific legal entities. Therefore, any user who has a specified role and access to a specified legal entity can access and use that view, even if that user doesn't not have permission to personalize. This publish capability lets organizations define corporate, standard views that are optimized for their business. For more information, see the Managing personalizations at an organizational level with views section.
- Unlike traditional personalization, views aren't automatically saved when a user performs personalizations or filters a list. Explicit saves are required to give users the flexibility to create a view before or after the changes that are associated with that view have been made. This requirement also ensures that view definitions aren't unintentionally changed by filters or personalizations that aren't intended for long-term use. Items that the system automatically stores as part of typical page usage (for example, column widths, or the expanded or collapsed state of sections) will be saved per view.
- Views can be added to workspaces as tiles, lists, or links. Therefore, a filtered data set can be surfaced in a workspace, and users can associate a set of personalizations that is relevant to that data set with a tile or link.

Switching between views

After views have been made available for an environment, the top of any page that supports views will include a collapsed view selector control that shows the name of the current view.

There are two size variations to the view selector:

- **Large view selectors** – Pages that prominently feature a list will have a larger view selector for a few reasons. Most importantly, the larger view selector indicates the pages where the view can include user-defined filters. Because filters are included in the views, the larger selector size is also warranted as the view names will often be the best description of the data shown on the screen and the expectation is that users will switch between views more often on these page types.
- **Small view selectors** – All other full-screen pages (except workspaces and the dashboard) have a smaller view selector that appears next to the page caption. Views on these pages include only personalizations, not

user-defined filters. On these pages, the caption or record title is often the most important information at the top of the page. The smaller size of the view selector also reflects the lower frequency of view switching that is expected on these pages.

If you select the view name, the view selector is opened and shows the list of available views for the page.

- **Standard view** – The **Standard** view is the out-of-box view of the page, where no personalizations are applied.
- **Personal views** – The views without padlocks represent your personal views. These are views that either you have created or that an administrator has given to you.
- **Locked views** – Some views (such as the **Standard** view and any views that are published to your role) have a padlock symbol next to them in the view selector. This symbol indicates that you can't edit those views. However, changes that reflect page usage are automatically saved. These changes include changes to the width of a grid column, and changes to the expanded or collapsed state of a FastTab. Nevertheless, if you have personalization privileges, you can use the **Save as** action to make a personal view that is based on a locked view.
- **New views** – Published views that haven't yet been opened have a spark symbol to the left of the view name.

To switch to a different view, first open the view selector and then select the view that you want to load.

Creating and modifying views

Unlike traditional personalization, views aren't automatically saved when a user personalizes the page, or when a user applies a filter to a list or sorts it. An explicit action is required to save these changes to a view. This requirement gives users the flexibility to create a view before or after the changes that are associated with that view have been made. It also ensures that view definitions aren't unintentionally changed by one-time filters or personalizations. Note that typical page usage items (for example, column widths, or the expanded or collapsed state of sections) are automatically saved to the current view, even for locked views.

To ensure that the current state of the view is known, when you start to change a view by personalizing or filtering it, an asterisk (*) appears next to the current view name. This symbol indicates that you're looking at an unsaved, modified version of that view.

If you want to save those changes, follow these steps.

1. Select the view name to open the view selector.
2. To modify the existing view, select **Save**. Note that this action isn't available for locked views.
3. To create a new view:
    a. Select **Save as**.
    b. Enter a view name and (optionally) a description.
    c. Select **Save**.

Changing the default view

The default view is the view that the system tries to open when you first open the page. You should set the default view to the view that you expect to use most often.

> **NOTE**
> There is a single, global default view across companies. If you change the default view, that view will be opened by default, regardless of the legal entity that you're currently in.

To change the default view for a page, follow these steps:

1. Switch to the view that you use as the default.
2. Select the view name to open the view selector.
3. Select **More** and then **Pin as default**.

Alternatively, when you create a new view (by using the **Save as** action), you can make that new view the default view by setting the **Pin as default** option before you save the view.

Note that, in some cases, the query that is associated with the default view isn't run when you first open a page. For example, if you open the page through a tile, the tile's query will be run, regardless of the query that is associated with the default view. Additionally, if you open a page that has a **Standard** view that already has a defined query, the original query will be run instead of the default view's query. In this case, you will receive an informational message when the view is loaded. If you switch views after the page has been loaded, the view query should be able to be run as expected. In version 10.0.10 and later, the informational message that you receive will have an embedded action that lets you load the default view's query directly.

Managing personal views

The **Manage my views** dialog box gives you basic maintenance capabilities over your personal views and the order of views in the view selector. To open this page, select the view name to open the view selector drop-down menu, select **More**, and then select **Manage my views**.

For a list of available views for that page, the following set of actions are available.

- **Change the default view** – Use the **Pin as default** action to make the currently selected view the default view for this page.
- **Reorder your views** – Use the **Move up** and **Move down** actions to rearrange your views in a specific order.
- **Rename a view** – Use the **Rename** action to change the name of the currently selected personal view. This action is turned off for locked views.
- **Delete a view** – Use the **Delete** action to permanently delete the currently selected view from the page. There is no way to recover a view after you remove it.

Any changes made in this dialog box will take effect after you select the **Save** button.

Managing personalizations at an organizational level with views

To help you understand how saved views help improve management of personalizations at an organizational level, this section describes some differences in personalization management with and without the **Saved views** feature.

Without views, administrators would apply a set of personalizations for a page to a user or a group of users via the Personalization page. If those users had personalization rights, the personalizations would be applied to that page. However, there was no ability to prevent users from further personalizing the page, which meant the organization could not ensure that its users had a consistent user interface. If any of those users didn't have personalization rights, the personalizations given to them by an administrator were not loaded. Further, if new users were hired into an organization, administrators needed to manually load a set of personalizations for the user. There was no automatic mechanism for specifying that a certain set of personalizations should be available for users in that role.

The **Saved views** feature makes organizational management of personalizations much easier, primarily because views can be published to groups of users. After a view has been published, any user who has one of the defined security roles and access to one the specified legal entities can see and use the view, even if that user doesn't have access to personalization. Although every user has a copy of the published view, where page usage

items are automatically applied, no user can save personalizations or query updates to a published view. In other words, published views are locked. Additionally, if new users are assigned to roles in legal entities that views were published to, they will automatically see the views that are associated with their roles and legal entities. No additional action is required by the admin. Likewise, if users change roles in an organization or are given access to different legal entities, they might no longer be able to access the views that were previously published to them. Again, no additional action is required by the admin.

Updates to a published view can easily be distributed to users by republishing the view to the appropriate security roles and legal entities.

The publish capability allows organizations to define corporate standard views that are optimized for their business, targeted at users in specific security roles.

Publishing views

During the publishing process, views can be assigned to one or more security roles for one or more legal entities. Therefore, any user who has access to a legal entity and is assigned to one of those roles can access and use the views. However, the user can't edit the views. By default, system admins have access to the **Publish** action in the view selector drop-down menu. However, other trusted users in your organization can also be given access to view publishing via the new **Saved views administrator** role.

To publish a view, follow these steps:

1. Create and save a personal copy of the view that you want to publish.
2. With that view currently loaded, select the view name to open the view selector drop-down menu.
3. Select the **More** button and then select **Publish**. The Publish dialog box will open.
4. Enter a name for the view. The name that you enter is the name that users who receive this view will see in their view selectors. The names of published views for a page must be unique. No duplicate names are allowed, even if the list of roles or legal entities that the views are applied to differ.
5. **Update 10.0.17 or later:** If the **(Preview) Translation support for organization views** feature is turned on, you can add translations for your view name in as many languages as your organization requires by selecting the **Translations** button next to the **Name** field. The view name will then be shown to users in their current language. You can also set the default language to specify the translation that will be shown to users who are running languages that no translation is defined for.
6. Optional: Enter a description for the view, so that users who receive this view can better understand its purpose.
7. Determine whether the view should be published as the default view for the selected users. When a view is made the default view, users will see it the next time that they open the target page. The single, global default view of every targeted user will be changed. However, users can still change their default view after publishing has occurred.
8. Add the security roles that correspond to the users who are being targeted by this view.
9. Determine whether you want to publish the view to the child roles of each security role that is selected. If you do, select the **Include child roles** check box in the row for the appropriate security roles. Note that this check box isn't available for roles that don't have child roles.
10. Add the legal entities that this view should be available for.
11. Select **Publish**.

Note that in some environments, it may take some time (up to an hour) before users see the published view.

NOTE

Be aware of the following expectations when you publish a view to a legal entity, or when you publish a view as the default view.

- If you publish a view as the default view to all or some legal entities, you change the single, global default view of every targeted user. If a user has roles where multiple views are published as the default view, the last view that was published will be used as the user's default view.
- If you publish a view to a legal entity, but you don't publish it as the default view, users will initially see the view in the view selector only for the specified legal entities. However, after the view is loaded for the first time, it will always be in the user's view selector for that page, regardless of the legal entity.

Modifying a published view

After you publish a view, you might find that you want to change it. Although you can't make live changes to a published view, because these views are locked for editing for all users (including publishers), you can republish a view to update it.

If the changes that you want to make to a published view only involve the publish parameters (the name and description of the view, or the security roles the view is published to), do the following:

1. Switch to the published view for the parameters that you want to update.
2. On the view selector drop-down menu, select **Republish**. If you're using version 10.0.12 or earlier, you must select **Publish** and then **Yes** to update the existing view.
3. Update the name, description, security roles, and legal entities for the view.
4. Select **Publish**. If you originally selected this published view as the default view, it will be the default view for users again after you republish it.

If the changes to the published view involve modifications of the personalizations or filters that are associated with the view, follow these steps.

1. Load the published view that you want to change.
2. Make the required changes to the local draft.
3. On the view selector drop-down menu, select **Republish**.
4. Select **Yes** to indicate that you want to publish the view together with its unsaved changes.
5. Adjust any publishing parameters that require adjustment, and then select **Publish**.

Managing published views

Like managing personal views, the **Manage my views** dialog box gives users with publish privileges basic maintenance capabilities over that page's published views (in addition to their own personal views). To open this page, select the view name to open the view selector drop-down menu, select **More**, and then select **Manage my views**.

Although all users have a **My views** tab that show their personal views, users who have publish privileges also have an **Organization views** tab that shows all the published and unpublished views for that page. Because several users might be publishing views, it's important that you be able to manage the full list of published views, even if you aren't the user who published a given view.

For the list of all published views for the page, the following set of actions are available.

- **Republish** – Use the **Republish** action to republish a view after publishing parameters (name, description, security roles, or legal entities) are changed.
- **Publish** – Use the **Publish** action to publish a view that is currently unpublished.

- **Unpublish** – Use the **Unpublish** action to make a view inactive. The view will still be available in the system, but users won't see it in the view selector until the view is published again.
- **Save as personal** – Use the **Save as personal** action to create a personal draft copy of the published view. This capability can help you understand the contents of a view that wasn't published to you or that hasn't yet been published. You can also use it to edit and then republish a view.
- **Delete** – Use the **Delete** action to permanently delete a published or unpublished view. This action also removes the view for all users in the system. The removal of published views takes effect after the **Save** button is selected. After a view is deleted, it can't be recovered.

Managing views globally

Although some management capabilities are surfaced on every page, as indicated in this topic, **system administrators** and **saved view administrators** can manage views more holistically for the system via the **Personalization** page. In particular, this page has the following sections and capabilities:

- **Published views** – This section lists all views that have been published for your organization. From here, you can republish a view after you adjust the security roles or legal entities that the view targets. You can also export, delete, or unpublish views. You can use the **Save as personal** action to create a personal copy of a view, so that you can update the view or gain a better understanding of its contents.
- **Unpublished views** – This section lists all the organization views in your system that aren't currently published. These views most often come into the system through the import capability. You can publish, export, or delete these views. The **Quick publish** action that was added in version 10.0.12 enables multiple views from this section to be published in one action, by using the existing security role and legal entity configurations. You can use the **Save as personal** action to create personal copies of these views, so that you can gain a better understand their contents.
- **Personal views** – This section lists all views that have been created by users in the system. From here, you can publish a personal view to the organization, or copy one or more of these views to other users. You can also export or delete these views as required.
- **User settings** – Select a user to view, or adjust the user's ability to use personalization either for the whole system or for specific pages that the user has visited. You can view and interact with the user's personalizations in the system. You can also delete all personalizations for that user or reset feature callouts for the user. If feature callouts are reset, any pop-up windows that introduced new features and that the user previously dismissed will appear again the next time that the user encounters those features.
- **System settings** – You can temporarily turn off personalization for all users in the system. In this case, no personalizations are applied for any user, and all pages are reset to their default state. If you turn personalization back on later, all personalizations are reapplied. You can also permanently delete all personalizations for all users in the system. Personalizations that have been deleted can't be recovered. Therefore, before you perform this task, be sure to export any personalizations that you might want later.

Users who have access to the **Personalization** page can also import personal or organization views by using the **Import views** button on the Action Pane. For organization views, you can select **Publish immediately** to make the views available to users without an additional explicit publish.

Known issues

For a list of known issues with saved views, please see Build forms that fully utilize saved views.

Frequently asked questions

How do I enable saved views in my environment?

NOTE

The **Saved views** feature requires the Personalization system in Finance and Operations to be enabled. If personalization is turned off for the entire environment, views will be disabled even if you follow steps below.

You can turn the **Saved views** feature on and off through Feature management in any environment. After it's turned on, saved views will be enabled in all subsequent user sessions.

What happens to existing personalizations when views are enabled?

When views are enabled, any existing personalizations for a user and form are saved into a new view called **My view** that is automatically set as the default view. This is meant to ensure that there is a consistent user experience before and after views are enabled, except for the view selector control appearing on forms.

What pages support views?

Views are available on most, but not all pages. Specifically, views are currently available on all full-screen pages except for dashboards and workspaces. Non-full-screen pages, which include dialog boxes, drop-down dialogs, lookups, enhanced previews, currently do not support views. View support for additional page types, such as workspaces and dialog boxes, may be considered for a future update.

Who is allowed to publish views?

Only system admins and users who have been assigned to the **Saved views administrator** role have the rights to publish views.

Why am I not able to save filters with this view?

There are a few reasons why a filter may not appear to save with a view:

- The page may not support saving filters as part of the view definition. Note that only pages with large view selectors allow personalizations and query modifications to be saved as a view. See the **Switching views** section for more information.
- The page in question may not properly support views, as it may ignore the view query completely or may operate on a temporary table whose data is not persistent.

What data will I see when I visit a page?

For pages that have small view selectors (only personalizations can be saved to the view), you will see the same data as you always have when you visit the page.

For pages that have large view selectors (both personalizations and queries can be saved to the view), you will typically see the data that is linked to the query that is associated with your default view. There are two main exceptions:

- If you navigate to a page from a tile, the tile query will execute regardless of the query associated with the default view. If you created that tile after views have been enabled, selecting a tile will open the page with the view associated with that tile.
- If you navigate to a page and that entry point includes a query, the original query will execute originally in place of the default view's query. You should be alerted when this occurs via an informational message when the view is loading. You can also confirm by switching to this view after the page loads, as that should allow the view query to execute regardless.

Build forms that fully utilize saved views

2/18/2021 • 9 minutes to read • Edit Online

Saved views are an important expansion of personalization capabilities Finance and Operations applications. While the Saved views topic provides general details about this feature, this topic focuses on the more technical elements of saved views as well as aspects of form development that may be impacted by views.

"User-perceived" pages

Traditionally, a set of personalizations has a 1:1 link to a modeled form. For many pages, this makes sense to the user, as the user's perception of the page matches the way in which the form is modeled. However, in some cases the 1:1 link of a modeled form to a set of personalizations is not intuitive or obvious because users do not see or care about the boundaries between modeled forms.

Saved views try to eliminate this confusion by letting users create views on "user-perceived" pages. Therefore, users don't have to understand how forms are modeled to understand how and when personalizations are applied. Consider the following two scenarios:

- **More than one "user-perceived" page in a single modeled form:** The standard modeling of Master Details and Transaction Details forms (for example, the **CustTable** and **PurchTable** forms, respectively) consists of more than one "user-perceived" page: a grid page and a details page.

  Because users are not aware when this transition from list to details crosses a form boundary (nor do they need to know this), view support in Details forms is handled differently to allow views to be defined separately for the grid and details portions. This means that the view selector for the "grid" and "details" can show different sets of available views. The special casing of view support on these forms also allows the "grid" views to allow filters in their view definitions, whereas the "details" view only need personalizations.

- **More than one modeled form in a single "user-perceived" page:** The ability to embed subforms into modeled forms (via FactBoxes or form parts) leads to situations where more than one modeled form corresponds to a single "user-perceived" page. For example, consider the details portion of the **All customers** page, which has a number of FactBoxes and two FastTabs whose contents come from form parts. With traditional personalization, contrary to a user's expectations, exporting the personalizations for the **CustTable** form would not include personalizations on any FactBox or any personalizations done inside the **Addresses** or **Contact information** FastTabs. Equally unexpected for users, any personalizations done on these subforms would also be reflected in other parts of the application where these form parts are used. For example, changes to the **Addresses** FastTab in the **All customers** page would also result in the same changes appearing in the **Addresses** FastTab in the **All vendors** page.

  With views, the personalization scope of form parts has been modified to match user expectations. In particular, subform personalizations are now tied to the base form where they are made. This means that exporting a view on the **All customers** details page will include personalizations from the base **CustTable** form as well as any personalizations in FactBoxes or form parts modeled in that portion of the form. Similarly, any personalizations done on the **Addresses** FastTab (form part) on the **All customers** page will not be reflected in other forms where that form part is used.

These are highly technical modifications to the personalization subsystem that are only available when view is enabled. These modifications are important for ensuring that users have a predictable and understandable experience with views.

View support

The style of a form determines the level of support for views.

- Views include queries for forms, such as:
  - List pages
  - Simple lists
  - Grid portions of Master Details and Transaction Details forms
- Views do not include queries, such as:
  - Any other full-page form
  - Details portions of Master Details and Transaction Details forms
- Views that are not currently supported include:
  - Dashboards
  - Workspaces
  - Dialogs
  - Secondary forms like drop-down dialogs, lookups, and enhanced previews

Modifying forms to fully utilize views

While most forms will work well with saved views, there are some areas that may require changes to form logic so that views work as expected on these forms without causing confusion. Here are some key items to keep in mind during development of new forms.

- X++ code late in the form startup cycle can interfere with views working as users expect. In particular, be aware of the following items:
  - Modifications of the query after **super()** of **executeQuery()** or after **super()** of **run()** can cause the query aspect of a view to be ignored.
  - Form changes after **super()** of **run()** can cause some user personalizations to be incorrectly applied to the default view.
- Extra work may be required to ensure the values of custom filters always align to the current view or query.
  - To make sure that custom filters work correctly with saved views, the platform still has additional work to better support these controls. Once that support is available, uptake will be required by any form that has custom filters. More information will be provided when the recommended approach has been finalized.
- Looking forward, in the long-term, views are meant to replace modeled secondary list pages.
  - Typically, secondary list pages, such as **Customers on hold**, are menu items that point to the same form but have a different query. Because menu items that pass in queries override any query that is defined on the default view, these entry points can cause confusion for users. The current long-term plan is to make secondary list pages obsolete (deprecated) and move them to views. However, that effort hasn't yet been started.
  - To avoid user confusion between form caption (such as "All customers") and view name (such as "My customers"), consider renaming form captions to be the name of the corresponding entity. For example, instead of a form caption of "All customers" or "All sales orders", the form caption would be modified to "Customers" and "Sales orders".

Known issues

This section provides a list of known issues for saved views while the feature is in a preview state.

Open issues

- A view does not get marked as having unsaved changes after using custom filters, which are the filters above a grid excluding the QuickFilter. If custom filter conditions have been saved to a view, the custom filter controls may not correctly reflect the current query.
- View support for workspaces, dashboards, and dialog boxes.
- [KB 4553227] After adding (reference group) fields via personalization, the fields remain blank.
- If the Filter pane is open when switching to a different view, the Filter pane will not update to reflect the filters on the target view.
- Cannot move a view with a QuickFilter condition saved to it to another environment. The fix in release 10.0.13 more gracefully handles the situation, but does not allow these conditions to move between environments.

Fixed in release 10.0.16

- [KB 4590240] Grid resize does not work properly when switching views with the old grid
- [KB 4600209] Personalizations of form parts are not reflected when switching views
- [KB 4590224] Focus can start on the wrong control when saved views is enabled
- [KB 4562254] Table permission error after accessing a shared custom workspace
- [KB 4600210] Unexpected client error when switching to the Hide tool
- [KB 4599871] Workspaces do not open if personalization is turned off for user / Unbound controls cannot be set as mandatory via personalization
- [KB 4594453] Duplicate key exception for forms opening as full-page forms and dialogs

Fixed in release 10.0.15

- (Quality update) [KB 4599871] Workspaces do not open if personalization is turned off for user / Unbound controls cannot be set as mandatory via personalization
- (Quality update) [KB 4600209] Personalizations of form parts are not reflected when switching views
- [KB 4594452] Duplicate record error when interacting with some subforms (form parts)
- [KB 4586310] Attachments page loses context after switching views
- [Bug 494204] Error when deleting/clearing personalizations from User options > Personalization

Fixed in release 10.0.14

- (Quality update) [KB 4594452] Duplicate record error when interacting with some subforms (form parts)
- (Quality update) [KB 4600209] Personalizations of form parts are not reflected when switching views
- (Quality update) [KB 4584077] Error when exporting multiple views
- (Quality update) [KB 4584775] Record position lost when switching between list and details
- [Bug 481290] Error when trying to re-import personalizations to a set of users
- [KB 4582745] Error triggered when importing user views from one environment to another

Fixed in release 10.0.13

- (Quality update) [KB 4594452] Duplicate record error when interacting with some subforms (form parts)
- (Quality update) [KB 4600209] Personalizations of form parts are not reflected when switching views
- (Quality update) [KB 4584077] Error when exporting multiple views
- (Quality update) [KB 4582719 and KB 4578126] When multiple personalization records exist for a form, the wrong one can be selected and loaded
- [Bug 481283] Error opening a form after moving a view with a QuickFilter condition between environments
- [Bug 481608] Database sync fails because of a unique index violation on the FormRunConfigurationPublishedView table
- [Bug 474817] User options > Personalization doesn't list all personalizations for the user

- [KB 4574781] Duplicate record exception on saving a view
- [KB 4575278] Tiles, lists, and links lose their link to the published view if the view is republished

> **NOTE**
> Because additional information is needed to restore the link, re-linking will not occur for any pinned elements from published views prior to 10.0.13. To mitigate, you will need to re-publish your views after updating to 10.0.13 and re-pin the elements to your workspace.

- [KB 4575285] Publishing to an existing view name overwrites configuration changes already made
- [KB 4574778] Pin and publish as default do not respect companies that the view was published to
- [KB 4568154] View import flow doesn't surface if views apply to the grid or details aspect of Details pages
- [KB 4568152] Users are able to export the Standard view
- [KB 4568151] Published views recipients are not updated after republishing from a different legal entity
- [KB 4562137] Views published to a parent security role are not applied to child roles
- [KB 4564528] QuickFilter default field personalization isn't working as expected with views

Fixed in release 10.0.12

- (Quality update) [KB 4582719] When multiple personalization records exist for a form, the wrong one can be selected and loaded
- [Bug 486275] Strange tooltip behavior for saved views
- [KB 4568122] Unexpected queries applied after enabling views
- [KB 4562152] Migration of personalizations after enabling saved views throws exception in some cases
- [KB 4568121] Default view personalizations not applied for users without personalization rights
- [KB 4568119] Error may occur when importing a workspace or adding a tile or list to a workspace with views enabled
- [KB 4568118] Error when trying to open the view selector when user has a large number of views
- [KB 4568148] Old custom user workspaces aren't shown in the Personalization form after enabling views

Fixed in release 10.0.11

- (Quality update) [KB 4562147] Importing personalizations to a large number of users is timing out
- [KB 4549735] Personalization form missing from security role
- [KB 4568116] Views are not marked as having unsaved changes after using Advanced filter or sort
- [KB 4568117] Crash when attempting to import old personalization formats
- [KB 4568115] Views can be published with no name
- [KB 4564908] Unsaved filters and personalizations reflected in some views

Fixed in release 10.0.10/Platform update 34

- (Quality update) [KB 4560406] Importing personalizations to a large number of users is timing out
- (Quality update) [KB 4564906] Personalization form doesn't load/Loading a published view for the first time takes a long time
- [KB 4568114] Views can be published to a blank legal entity
- [kB 4568113] "View query cannot be applied" message shown when loading a view that modifies existing filters

Test forms that use custom patterns

2/18/2021 • 3 minutes to read • Edit Online

This topic how to test forms using custom patterns.

Introduction

By adhering to form patterns, you gain various benefits. For example, form patterns correctly set layout properties so that forms are laid out responsively. However, when form pattern coverage is lacking (for example, there currently isn't support for many extensible controls), or when a form or container has unique requirements/uses that don't fit any pattern, developers can set the pattern to Custom. The developer then becomes responsible for ensuring a correct and responsive form layout.

Forms that use custom patterns

You can find the forms that use custom patterns by using the **Form Patterns** report. For information on running the report, see Form pattern add-ins. After running the report, filter the **Percent covered controls** column to show forms that have less than 100-percent coverage. For forms that have a top-level Custom pattern, **Custom** will appear in the **Patterns** column.

Testing configurations

Key resolution

- 1366 × 768 is a typical resolution on screen sizes that are between 12 and 23 inches. Therefore, this resolution provides a good baseline for testing.
- It's also a good idea to test on a higher resolution, such as 1920 × 1080.

Parameters

- Browser
  - Internet Explorer 11
  - Google Chrome
  - Microsoft Edge
  - Apple Safari (on iPad) – You'll have to point Safari to a cloud URL.
    - Landscape and portrait modes
- Density
  - Low density
  - High density
- Viewport size
  - Maximized window
  - Snap view (half the screen) to simulate portrait mode on a tablet
- Zoom
  - 50 to 200 percent

Steps

Follow these steps. At each step, examine your form for layout issues. As part of your examination, look at all tab pages, and any groups that can expand/collapse content.

1. Open a browser window at full-screen size, and navigate to your form/control (in low density). It's a good

idea to starting your testing at a resolution of 1366 × 768.

2. Press the Windows logo key+Left Arrow to switch the browser window to Snap view (half the screen).
3. Slowly resize the browser horizontally back to full width. Stop at intervals, and evaluate the form layout.
4. Slowly resize the browser vertically from full height to half-screen height. Stop at intervals, and evaluate the form layout.
5. Maximize the browser window to full-screen size. Adjust the zoom levels (50 percent, 75 percent, 125 percent, 150 percent, and 200 percent), and evaluate the form layout.
6. Do a sanity check in high density.
7. Do a sanity check in other browsers.

Visual issues that you might encounter

Questions to answer while you're testing

- Is the form usable?
- Can I reach everything on the form? (You might have to move multiple scrollbars in smaller viewports.)

Issues that might suggest problems with the layout metadata

- Form content isn't being adjusted based on the available space.
- Grids and other "fill" controls aren't taking up the full height/width.
- Containers aren't showing up at all (there are no scrollbars that you can use to get to parts of the form).
  - This issue can occur with **SizeToAvailable** containers when there is no available height/width.
- There are extra (unnecessary) scrollbars.
  - You should expect more scrollbars in smaller viewports, especially because some containers (for example, grids and tab pages) have a minimum height.
  - This issue can occur with **SizeToContent** containers that should be **SizeToAvailable**.

Other known/expected issues

- Horizontal scrollbars appear on Toolbars.
  - Where there isn't enough width to show all the buttons on a Toolbar, a horizontal scrollbar will appear. This issue will be resolved soon by a framework deliverable that will implement overflow behavior on Toolbars.
- Random input control borders are missing at various zoom levels.
  - This is a known issue and is tracked by 1721990.
- Grids receive extra scrollbars, because the space that is available for the grid is less than the grid's minimum height (200 px).
  - We have a future work item to investigate reducing/removing this minimum height.
- When StaticText controls that have **SizeToAvailable** width are inside a group that also has **SizeToAvailable** width, horizontal scrollbars appear at some widths (Internet Explorer only).
  - The browser has a calculation error that sometimes causes scrollbars to appear in this scenario.

Appendix

Information/guidelines about layout

For information about the layout properties, and for guidelines about scenarios that you should avoid, see Page layout.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Create shareable, secured URLs (deep links)

2/18/2021 • 3 minutes to read • Edit Online

Learn how to create shareable, secured URLs to forms and records.

Overview

The URL Generator enables developers to create shareable and secured URLs (also known as deep links) to specific forms that are root navigable. An optional data context can be passed to the form to display filtered or specific data when the form is opened. The URL Generator enables scenarios such as embedding links in reports, email, and external applications, enabling users to quickly and easily locate the specified forms or data by simply navigating using the generated link.

Purpose

- Empower developers to generate URLs that can be used to navigate to root navigable forms in a specified instance.
- Empower developers to optionally specify a data context that should be displayed when navigating to the specified form.
- Empower users to share, save, and access the generated URLs from any browser with Internet access.
- Secure the URLs to prevent unauthorized access to the system, forms, or data.
- Secure the URLs to prevent exposure of sensitive data or tampering.

Security

Site access

Access to the domain/client is controlled through the existing login and SSL mechanism.

Form access

Access to forms is controlled through Menu items, as Menu items are the entry points where security is enforced. If a user navigates using a URL that contains a Menu item that the user does not have access to, then the Menu item security will prevent the form from opening. The user will receive a message indicating that they do not have the necessary permissions to open the form. Note that deep links will only work for Menu items that allow root navigation.

Data access

Access to data is controlled through the existing form-level queries. When a form is opened with a generated URL, the form will run its existing form-level queries, which restrict the user's access to data. The data context that is specified in the generated URL is consumed after these form-level queries are applied, and results only in further filtering of the data displayed to the user. In short, a generated URL can, at most, open a form and display all of the data that a form would display to the user based on the form-level queries. A generated URL cannot grant a user access to data that is otherwise inaccessible on the form when not using the generated URL.

Usage

The URL Generator is a .NET library that is accessible from X++, under the following namespace.

```
Microsoft.Dynamics.AX.Framework.Utilities.UrlHelper.UrlGenerator
```

Requirements

The URL Generator must be used from code running on the AOS, in an active user session or batch process. This requirement ensures that the URL can be secured through encryption specific to the instance that generates the URL. At a minimum, the following information must be specified and passed to the URL Generator in order to generate a working URL.

- **Host URL**
  - The URL of the web root for the instance. For example: `https://ax.dynamics.contoso.com/`
- AOT name of the **Menu Item Display**
  - The menu item display to be used to open the form.
- **Partition**
  - The partition to use for the request.
- **Company**
  - The company to use for the request.

Example

```
// gets the generator instance
var generator     = new Microsoft.Dynamics.AX.Framework.Utilities.UrlHelper.UrlGenerator();
var currentHost   = new System.Uri(UrlUtility::getUrl());
generator.HostUrl = currentHost.GetLeftPart(System.UriPartial::Authority);
generator.Company = curext();
generator.MenuItemName = <menu item name>;
generator.Partition = getCurrentPartition();

// repeat this segment for each datasource to filter
var requestQueryParameterCollection = generator.RequestQueryParameterCollection;
requestQueryParameterCollection.AddRequestQueryParameter(
    <datasource name>,
    <field1>, <value1>,
    <field2>, <value2>,
    <field3>, <value3>,
    <field4>, <value4>,
    <field5>, <value5>
);

System.Uri fullURI = generator.GenerateFullUrl();

// to get the encoded URI, use the following code
fullURI.AbsoluteUri
```

Accessibility in forms, products, and controls

2/18/2021 • 8 minutes to read • Edit Online

This topic describes best practices for enabling accessibility in your form, product, or control. An accessibility checklist is also included.

Accessibility is about inclusion, that is, a person with a disability can perform the same task as a person without that disability. Making an accessible control or form should be as fundamental as making it secure, high-performing, or easy-to-understand.

Keyboard

The bedrock of accessibility is keyboard-only access. When you can use the keyboard to perform all the actions of a form, then it can be utilized by a non-sighted person or a person with restricted or limited use of their hands. This means that all controls can be reached via the tab sequence, direct action (such as **Ctrl+S** for Save), or through some other shortcut key that enables the user to move to a control, such as Navigation Pane, App Bar, Message Center, or Message Bar. A simple test is to simply disconnect your mouse and complete all core and secondary scenarios using only the keyboard.

Color

The use of color is encouraged and is a common way to express state or status of a record or other piece of information. However, color cannot be the only way that state or status is communicated. An accompanying symbol, help text, or additional column should include a textual description of the state or status. A simple test is to identify all use of color in your system and ensure that color isn't being used to express a state or status. A common example is to use the color red to indicate "needs attention," or the color green to mean an "OK" status.

Images

When showing an image there should be a label that describes the image. If the image expresses state or status of a record, then accompanying help text or an additional column should include a textual description of the state or status. If the image is symbolic, like a logo, then it doesn't require a textual description. If you have an image on a form or grid to convey a status, such as "in progress", ensure that the image has a tooltip that can then be read to someone who is utilizing a screen reader.

```
public display container statusImageDataMethod()
{
    ImageReference statusImage;
    if (this.Status == NoYes::Yes)
    {
        statusImage = ImageReference::constructForSymbol(ImageReferenceSymbol::Accept, "Accept");
        // a product ready example would use a label in place of an embedded string
    }
    else
    {
        statusImage = ImageReference::constructForSymbol(ImageReferenceSymbol::Cancel, "Cancel");
        // a product ready example would use a label in place of an embedded string
    }
    return statusImage.pack();
}
```

Extensible controls

Extensible controls are simply an extension of the controls that are provided by the framework, but require the same usability and accessibility. In addition, widgets or other visually rich controls, such as segmented entry controls or charts, should provide an Accessible Rich Internet Applications (ARIA) tag that is hidden from the sighted user, but provides an introduction to the blind user through a screen reader that describes the use of the control. Every action in an extensible control must be possible with a keyboard.

Layout and dynamic management of forms

A visually impaired or blind user cannot be surprised by significant or unexpected re-laying out of a form based on an action such as selecting a value or entering input.

Accessibility checklist

MEASURE	DESCRIPTION	SUCCESS CRITERIA	VALIDATED (X)
Keyboard-only access to all functions	Drill downs, lookups, hyperlinks, data input actions, and shortcut keys.	All action may be completed without the use of a mouse.	
Visible focus indicator		It's always apparent which control has focus.	
Focus order	Cannot "jump to" a non-logical location.	Tabbing doesn't jump to a non-logical portion or unexpected portion of the form.	
Focus trapping	Cannot be "trapped" in a control, focus must be able to move out of control using the keyboard.	Tabbing into a control should allow tabbing out or a keyboard equivalent that lets them escape.	
Images of text	Cannot use an image to display text.	For example, you cannot have a button label that is an image of text. Logos are exempt.	
Using color	Color alone cannot be used to convey status or state.	All status indicators on a form or grid must have unique shape or texture for each color. See the "Color" section in this document for more information.	
Text contrast	Minimum 4.5:1 ratio.	Extensible controls should use framework color theming to ensure compliance.	
User input instructions	Widgets or other multi-step controls must have a usability overview label or help text (can be embedded in ARIA tag).	When using a screen reader, the control must be introduced and its label or purpose described.	

MEASURE	DESCRIPTION	SUCCESS CRITERIA	VALIDATED (X)
Change of context	Changing the setting of, or entering data into any UI component must not automatically cause an unexpected change of context that might disorient the user without first notifying the user that the change of context will occur.	For example, selecting a value in a drop-down box or clicking a check box shouldn't change the layout of the form unexpectedly or in a non-standard way.	
Consistent UI	Widgets or other elements must work consistently across the product. There cannot be two similar looking or modeled elements that behave differently.	For example, selecting a check box shouldn't open a window.	
Fine motor control	Cannot require use of the mouse. Must support "sticky keys."	User cannot be forced to click a moving target (pull right) or any other action that requires a mouse. The same action must also be possible with the keyboard.	
Use of video	Video (with audio) must have accompanied text.	When showing pre-recorded content, the deaf person must have supporting text to consume the content.	
Use of images		All images must be accompanied by text that describes the specific content. For example, a "tooltip" or secondary column in a grid that describes and matches the state or status.	
Communication	Use of VoIP or other telecommunication device or software, such as Skype, must be accessible and usable from the keyboard.	Any use of VoIP or telecommunication is a legal obligation.	
Navigation		Navigation controls must communicate that navigation will occur if executed.	

Extensible controls – insights for accessibility

An extensible control is simply an extension of the framework. The interaction of the extensible control should be considered no different than any other framework control. When the control is a visually-rich widget that offers mouse actions, the author needs to ensure that equivalent, keyboard access functionality is available. A widget may not run in an "accessible mode" that is different than the standard presentation. Instead the widget

must offer similar functionality without the need to activate or toggle state of the control. All uses of color should be based on themed colors so that when the mode is "High Contrast" your color scheme matches the theme change. The World Wide Web Consortium (WC3) website provides guidance for Supported States and Properties. This site is a helpful resource for ARIA tags and provides the definitions that you see below.

Controls should do this

Introduce itself

Importantly, your control should not only identify itself by name, but (using a label or ARIA tag) give a brief introduction on how it works. Make sure your descriptive text is supplied via a framework label for localization.

- *aria-describedby* - Identifies the element (or elements) that describes the object.

Introduce itself - example

```
<http://www.w3.org/TR/WCAG20-TECHS/ARIA1.html>
<button aria-label="Close" aria-describedby="descriptionClose" onclick="myDialog.close()"></button>
<div id="descriptionClose">Closing this window will discard any information entered and return you to the
main page</div>
```

Indicate when it is busy

It may not always be clear to the visually-impaired user why the control isn't responsive. Providing a "busy" message helps in these cases.

- *aria-busy (state)* - Indicates whether an element, and its subtree, are currently being updated.

Indicate when it is busy - example

```
<p aria-live=”polite” aria-busy=”true”></p>
```

Indicate that the contents have been validated and are invalid

The async nature will result in a dynamic field state change. The message bar will introduce itself to the visually-impaired user, and the control itself should express an invalid state.

- *aria-invalid (state)* - Indicates the entered value does not conform to the format expected by the application.

Indicate when a field is readonly

- *aria-readonly* - Indicates that the element is not editable, but is otherwise operable. See related *aria-disabled*.

Indicate that the field requires input (mandatory)

The sighted user understands that a field is mandatory through a visual symbol. The non-sighted user will need an identifying tag.

- *aria-required* - Indicates that user input is required on the element before a form may be submitted.

Describe the state of a toggled value

A toggle control has a toggled state. This tag will express that state.

- *aria-pressed (state)* - Indicates the current "pressed" state of toggle buttons. See related *aria-checked* and *aria-selected*.
- *aria-valuenow* - Defines the current value for a range widget. See related *aria-valuetext*.
- *aria-valuetext* - Defines the human readable text alternative of *aria-valuenow* for a range widget.

Controls could do this

Indicate an expanded state

Complex interactions can be learned, but current state isn't always easy to determine without experimentation. When using an *aria-expanded* tag, the control describes its current state. An example is tabbing to a tab or FastTab section of a control.

- *aria-expanded (state)* - Indicates whether the element, or another grouping element it controls, is currently expanded or collapsed.

Describe applicable context menu

Finance and Operations apps provide a context menu. When the application author has provided functionality to the current control or context, you can announce that functionality.

- *aria-haspopup* - Indicates that the element has a pop-up context menu or sub-level menu.

Other miscellaneous controls

- *aria-live* - Indicates that an element will be updated, and describes the types of updates the user agents, assistive technologies, and user can expect from the live region.
- *aria-multiline* - Indicates whether a text box accepts multiple lines of input or only a single line.
- *aria-multiselectable* - Indicates that the user may select more than one item from the current selectable descendants.
- *aria-orientation* - Indicates whether the element and orientation is horizontal or vertical.
- *aria-setsize* - Defines the number of items in the current set of listitems or treeitems. Not required if all elements in the set are present in the DOM. See related *aria-posinset*.
- *aria-sort* - Indicates if items in a table or grid are sorted in ascending or descending order.
- *aria-valuemax* - Defines the maximum allowed value for a range widget.
- *aria-valuemin* - Defines the minimum allowed value for a range widget.

Customize field descriptions

2/18/2021 • 2 minutes to read • Edit Online

This article describes how you can customize existing field descriptions and add your own descriptions.

There are descriptions for some of the more complex fields. These descriptions appear when you hover over a field. You can customize these descriptions if, for example, you want to add company-specific information. You can also add descriptions for additional fields. You create field descriptions by using the **HelpText** property for field controls. The **HelpText** property is no longer specified for table fields and data types, as it was in previous versions. Additionally, the inheritance of the **HelpText** property from data types and table fields to form controls is obsolete. Field descriptions are intended to be specific to an individual field, in the context of the other controls and information that are available on the page. To add and customize field descriptions, you must have access to the development environment. Like other metadata changes, new descriptions should be added in a new model to prevent them from being overwritten when a new version of Operations is released. For more information, see Customize through extension and overlayering.

Customize a field description or add a new description

The same procedure is used to customize existing field descriptions and to add new field descriptions. However, when you customize an existing description, you replace the existing label reference.

1. In Application Explorer, find the relevant page (form), and add it to your project.
2. In the node for the page, find the relevant field control. Make a note of the name, so that you can use it as part of the label ID.
3. Add a new label for your description. You can follow the conventions that Microsoft uses to name the label files for field descriptions and to create the label file IDs. For more information, see the next section.
4. In the **HelpText** property of the field control, add a reference to the label.

Label file names and label IDs

The field descriptions that are provided by Microsoft are stored in separate label files. There is one label file per module per model. The pattern for the label file names is FieldDescriptions_*ModuleName*_*ModelName*.*CountryCode*.label.txt. Here are some examples:

- FieldDescriptions_AccountsPayable_ApplicationFoundation.en-US.label.txt
- FieldDescriptions_SystemAdministration_ApplicationFoundation_en-US.txt

The pattern for label IDs is @FieldDescriptions_*ModuleName:PageName*_*ControlName*. Here are some examples:

- @FieldDescriptions_AccountsPayable:CustSettlement_CustSettlement_OffsetCompany is the ID for the label for the **Company accounts** field (CustSettlement_OffsetCompany) on the **Customer settlement** page (CustSettlement).
- @FieldDescriptions_ProcurementAndSourcing:PurchLineBackOrder_LinkViewCheckBox is the ID for the label for the **Link on change view** option (LinkViewCheckBox) on the **Backorder purchase lines** page (PurchLineBackOrder).

Additional resources

Create localizable labels

View and export field descriptions

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Action controls

2/18/2021 • 11 minutes to read • Edit Online

Actions are an essential component of any enterprise resource planning (ERP) system, and are triggered by mouse click, keyboard, or touch.

Introduction

Actions are an essential component of any enterprise resource planning (ERP) system. Actions can be accessed through various mechanisms:

- Buttons on standard Action Panes
- Buttons on Toolbars
- Buttons directly on the form canvas
- Right-click context menus
- Keyboard shortcuts
- The new action search feature

Note that, in general, actions that are triggered by right-click context menus or keyboard shortcuts are meant to have a corresponding button available elsewhere in the user interface. Action controls can be triggered by using touch or a mouse click. Many system-provided actions can also be triggered by using the keyboard. In the future, we plan to provide functionality so that developers and end users can also define their own keyboard shortcuts.

Buttons

Buttons are the foundation of action controls. They can be modeled inside of standard Action Panes or in Toolbars, which are discussed later in the article. They can also be added as stand-alone buttons on the page (for example, the **OK** and **Cancel** buttons at the bottom of a dialog box, or buttons for actions that are specific to an individual field). The following button types continue to be available:

- A **button** is a basic button, for which the entire functionality must be implemented in code.
- A **command button** specifies a command or task to run.
- A **menu item button** specifies a menu item to navigate to or run.
- A **drop dialog button** opens a flyout dialog box, the contents of which are retrieved via a menu item.
- A **menu button** is a button container that opens a flyout that contains a list of other buttons.

In general, buttons continue to take advantage of the same properties as the buttons in previous versions. The following sections discuss a few properties that are related to button visualization, with particular focus on changes from previous versions.

Button Display

The **Button Display** property controls what information (including the button label and/or image) appears on the button. The allowed values for this property depend on where the button is located (for example, inside an Action Pane). Here are some of the location restrictions on the **Button Display** property:

- Buttons inside Menu Buttons must be set to **Text Only** or **Auto** (which is interpreted as **Text Only** in this case).
- Buttons inside Action Pane tabs on Standard Action Panes must be set to **Text Only** or **Auto** (which is interpreted as **Text Only** in this case).

- **Image Only** buttons should be used only for on-canvas buttons that are inline with a field.

For more details about how to use the **Button Display** property in various form locations, see the "Button image guidelines" section of the General form guidelines article. The following table shows the values for the **Button Display** property.

BUTTON DISPLAY VALUE	DESCRIPTION
Auto	The button has the content (text and/or image) that is defined (but only content that is allowed for the button's location).
Text Only	Only text appears on the button.
Image Only	Only an image appears on the button.
Text with Image Above	Both an image and text appear on the button. The text appears above the image.

Note that the other values of **Button Display** from Microsoft Dynamics AX 2012, such as **Background Image** and other relative positioning of text and image, are no longer supported.

Button Images

In previous versions, images or icons were often shown on buttons to help users recognize those buttons. In the current version, the number of images that are used for this purpose has been drastically reduced. Fewer images produce a cleaner, more modern user interface. Additionally, there was a desire to indicate processes and tasks by using more common symbols instead of multiple subtly different images. For more details about how images are used on buttons, see the "Button image guidelines" section of the General form guidelines article.

Two metadata properties are used to define an image for a button: **Image Location** and **Normal Image**. The allowed values for the **Normal Image** property depend on the value of the **Image Location** property. In previous versions, Embedded Resources (kernel resources) were heavily used to specify button images or icons. However, with the shift to the web, this image format option is no longer available. Instead, a new image format (Symbol font) has been added, and the expectation is that all buttons that require images will use this format (**Image Location** = **Symbol**). The primary reason for this change is that a symbol font is the best performing and most scalable image format. For a list of the full set of symbols that are supported, see Dynamics Symbol font article. The following table shows the recommended and preferred method for assigning images to buttons.

PROPERTY	VALUE
Image Location	Symbol
Normal Image	The name of the symbol font glyph

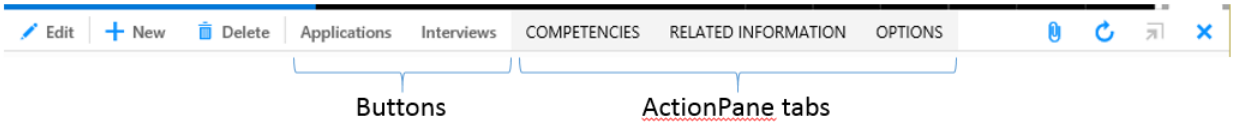
Button Style

In general, the **Button Style** property defines how a button is shown in the user interface. The exceptions are buttons that are modeled inside an Action Pane or Toolbar, because the **Button Style** property is disregarded in those cases. Instead, those buttons are rendered by using the style specially designated for buttons in those types of containers. For buttons that are modeled directly on the form canvas (outside Action Panes), the following button styles are available.

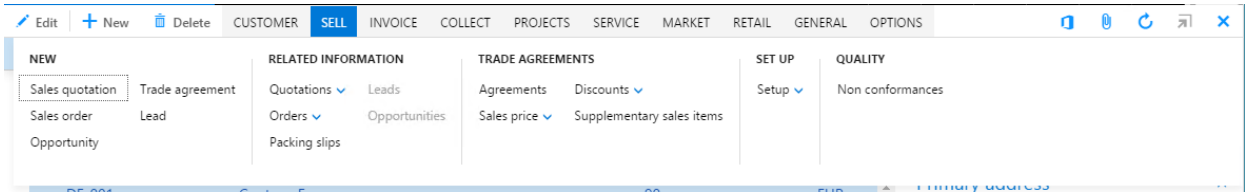
STYLE	EXAMPLE	DESCRIPTION
Standard (Auto)	Show more fields	The traditional button appearance
Command link	Vendors Customers	A combination of a small square block or image on the left, and a label on the right
Link	Current rentals	A button that has the appearance of a hyperlink

Standard Action Panes

The standard Action Pane is the primary location for page-level actions. It consists of both system-defined actions (actions that aren't explicitly modeled but are automatically added by the framework) and developer-defined actions (actions that are explicitly modeled in either Action Pane tabs or Button Groups). Developers can promote the most frequently used actions directly to the standard Action Pane by modeling Button Groups directly under the Action Pane. However, Action Pane tabs can still be used to group actions and provide access via a flyout. The following illustration shows a standard Action Pane that includes system-defined buttons, two promoted developer-defined actions, and a set of Action Pane tabs.



The following illustration shows the flyout that appears to show additional commands when an Action Pane tab is clicked.



System-defined buttons

Several system-defined buttons are added automatically to pages. The following table shows the list of system-defined buttons that are added to the Action Pane. For more information about how these buttons behave and how to manage them, see the System-defined buttons article.

BUTTON	NAME	COMMENTS
New	New	Create a new record for the first master data source.
Delete	Delete	Delete the currently selected record for the first master data source.
Edit	Edit	Switch to **Edit** mode.
	Show filters	Open the **Filter** pane.

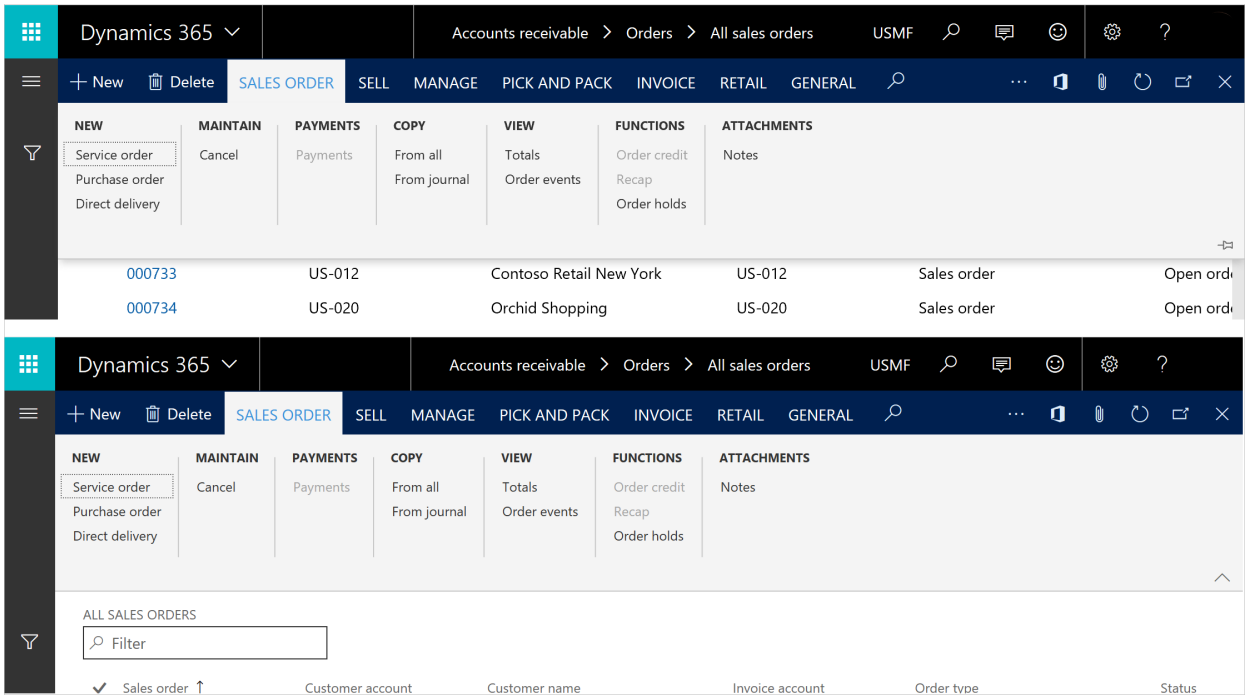
BUTTON	NAME	COMMENTS
	Show list	Toggle visibility of the navigation list on the details pages.
	Attach	Attach a document.
	Refresh	Update all the data on the page.
	Close	Close the page (equivalent to clicking the browser's **Back** button).
	Open in Microsoft Office	Open or export to Microsoft Excel. More Office integration is planned.
	Popout	Pop out the current form into a new dynalinked window.

Pinning the Action Pane

The standard Action Pane supports the ability for the user to "pin" or "unpin" the Action Pane as desired.

When the Action Pane is pinned open, an Action Pane tab is expanded and pushes the form content below it (e.g. it does not overlap anything on the form). In this mode, there is a chevron button in the lower right corner of the expanded Action Pane tab to "unpin" the Action Pane.

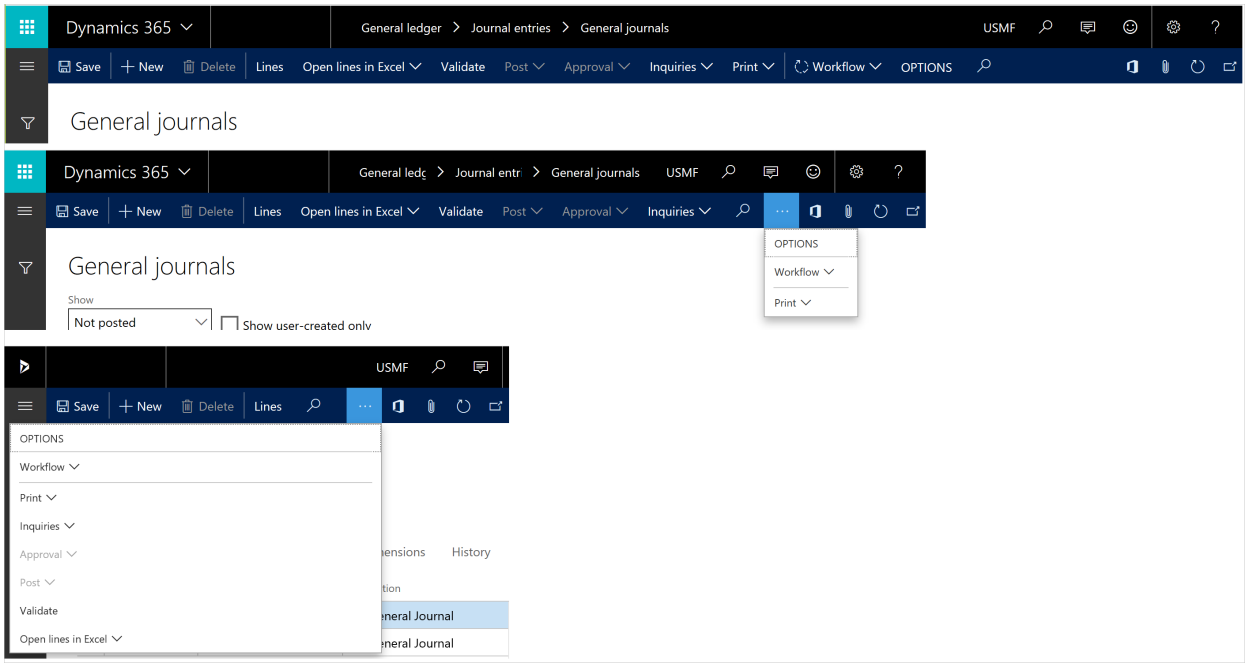
When the Action Pane is not pinned open, clicking on an Action Pane tab opens it as a flyout on top of the form content. The lower right corner of the Action Pane tab flyout has a pushpin button that can be clicked to pin the Action Pane open.



Overflow behavior in the Action Pane

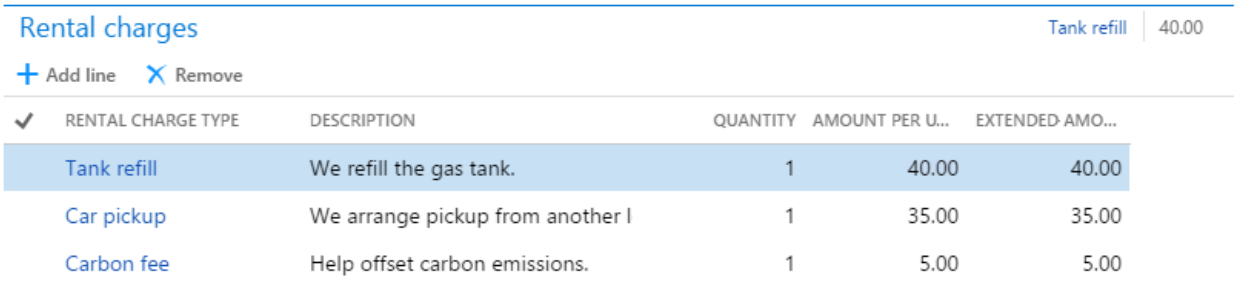
Standard Action Panes include an overflow feature that adds a responsive element to forms and eliminates the need for a horizontal scrollbar in the Action Pane. When the browser width is insufficient to show the entire Action Pane contents, an overflow menu automatically appears in the Action Pane and includes any buttons and

Action Pane tabs that did not fit given the browser width. Items are added one-by-one starting with the rightmost actions from the Action Pane. Note the system actions on the right side of the Action Pane do not participate in the overflow behavior.



Toolbars

Toolbars (previously called Action Pane strips) are Actions Panes that have the **Style** property set to **Strip**. They are used for actions that have a specific context and aren't page-level actions. They are primarily used for actions that are specific to a FastTab, tab, or grid. The actions in a Strip-styled Action Pane are shown horizontally in a Toolbar. The following illustration shows a Toolbar that has two buttons for adding and removing lines from this **TransactionDetails** form.



Overflow behavior in Toolbars

Toolbars have the same overflow feature as standard Action Panes. See the section above for more details.

Toolbars allow developers to provide a cleaner action story by designating certain buttons to always render in overflow. This makes it easier to differentiate actions that users will commonly use versus those that are infrequently or rarely used. This behavior is controlled by a new metadata property called **AlwaysInOverflow** that exists on Button groups inside Toolbars.

Right-click context menus

Some actions can also be accessed via shortcut menus (right-click context menus). Depending on the context of the right-click, you see either the browser's default context menu or the application context menu, which shows both system-defined actions and developer-defined actions.

- If you right-click on an image, in an editable field, or if text is selected, then the browser's context menu will appear. This is to provide access to browser functionality like **Cut**, **Copy**, and **Paste**. Actions cannot be embedded into context menus because browsers do not allow programmatic access to the system

clipboard for security reasons.

- Other right-click targets (for example, on a field label or on the value of a read-only control) should trigger the context menu.

> **NOTE**
> Context menus are intended to provide an alternate route to a command, and should not be only way to execute a command. Therefore, any action that is added to a control's context menu should also have a corresponding action that is available outside the context menu.

The programming model for modifying context menus differs from the model used in previous releases. In Dynamics AX 2012, the **PopupMenu** class was used. This class relies on Microsoft Windows application programming interfaces (APIs). However, because these APIs aren't available on the web, replacement APIs have been created to provide similar functionality. For more information, see Code migration - Context menu code.

Keyboard shortcuts

Keyboard shortcuts are another mechanism for triggering some actions. Many actions that had shortcuts in Dynamics AX 2012 continue to have shortcuts in Operations. However, because of browser restrictions, the key combination that is used to trigger a particular action might differ.

The following table shows some important keyboard shortcuts that are available. For the full list of current keyboard shortcuts, see the Keyboard shortcuts article. In the future, we plan to provide mechanisms so that developers and end users can define shortcuts for other actions.

KEY COMBINATION	ACTION	COMMENTS
Alt+N	Create a new record	The keyboard shortcut for creating a new record (unlike the system-defined **New** button on the Action Pane) is contextual. It creates a new record that is based on the data source of the control that currently has the focus.
Alt+Del (or Alt+F9)	Delete record	The keyboard shortcut for deleting a record (unlike the system-defined **Delete** button on the Action Pane) is contextual. It deletes based on the data source of the control that currently has the focus.
Alt+S (or Ctrl+S)	Save a record	
F2	Toggle edit mode	Switch the form between **View** mode and **Edit **mode.

Action search

Dynamics AX 2012 included a Key Tips feature that let users run any command in an action pane by pressing Alt and then a series of letters. The action search feature has been implemented as the replacement for the old Key Tips functionality. Action search can be accessed through a search field that is located in the standard action pane at the top of the form. Currently, it's represented by a magnifying glass icon in the action pane (note this is different from the navigation search feature in the navigation bar). In the action search field, you can type the beginning of the name of the action that you want to perform in the field (typically, only two to four characters

are required). This search mechanism then finds all buttons in visible action panes on the form that match the search string. You can then use click the button in the result list to run the command. For productivity, focus then returns to your last position in the form after the button has been triggered. You can also initiate action search by pressing Ctrl+' or Alt+Q. Pressing the keyboard shortcut again will return focus to your last position in the form.

Input controls and grid column sizes

2/18/2021 • 2 minutes to read • Edit Online

This topic describes how to create a consistent look and feel for forms by controlling the size of controls and grids.

Overview

Many frameworks offer complete freedom over the width of input controls. However, that level of freedom can lead to inconsistent presentation of data and a non-uniform layout for similar forms. For example, the customer name in one form might show 10 characters of information, whereas the customer name in another form might show 20 characters of information. To provide streamlined, easy-to-read interfaces, discrete sizing helps guarantee consistent presentation of data. The introduction of discrete sizing is a significant change to the basic input controls. The control framework attempts to provide a fresh, clean user experience that provides simplicity and consistency. As part of an attempt to provide consistent and uniform layout of forms, each input control is sized to one of four sizes: extra-small (XS), small (S), medium (M), or large (L). These sizes are determined by inspecting the explicitly specified width in the **DisplayLength** property of the control or the corresponding extended data type (EDT).

Sizing grid columns

Column sizing in a grid control differs slightly from the legacy algorithm, which tried to provide an initial appealing presentation for each grid, based partly on the contents of the first page of data. The new approach disregards the contents of the first page of data. Instead, the end user decides which columns are most important to view and the ideal viewing width for each of those columns. By using personalization, each user can change the width of grid columns, and the client keeps track of that user's preference. In the new, simplified approach, the default column sizing is based on 75 percent of the base input control sizing. In most cases, we recommended that developers not override the default sizing. However, if you must resize the column width programmatically or in the model, see the next two sections of this article for guidance.

Discrete sizing

SIZE	CHARACTER RANGE	SIZE IN PIXELS (PX)
XS	0–5	60
S	6–15	120
M	16–30	180
L	>30	240

Note: The explicit number of pixels will likely vary over time as the user interface evolves. Developers should not rely on explicit pixel sizing.

Forcing a desired discrete size

As the previous table shows, if you change the width of a grid-hosted control from 6 characters to 15 characters, you don't affect the width of the column. The layout engine gives the same width to all control widths that are in

the 6-to-15-character range, because controls in this range are considered small (S). If you want to extend the control to medium (M) size, the width value must be set to a value that is more than 16 characters and less than 31 characters. For guidance about how to size input controls, see the following table. In some cases, the use of a form pattern will override or hide developer-defined sizing.

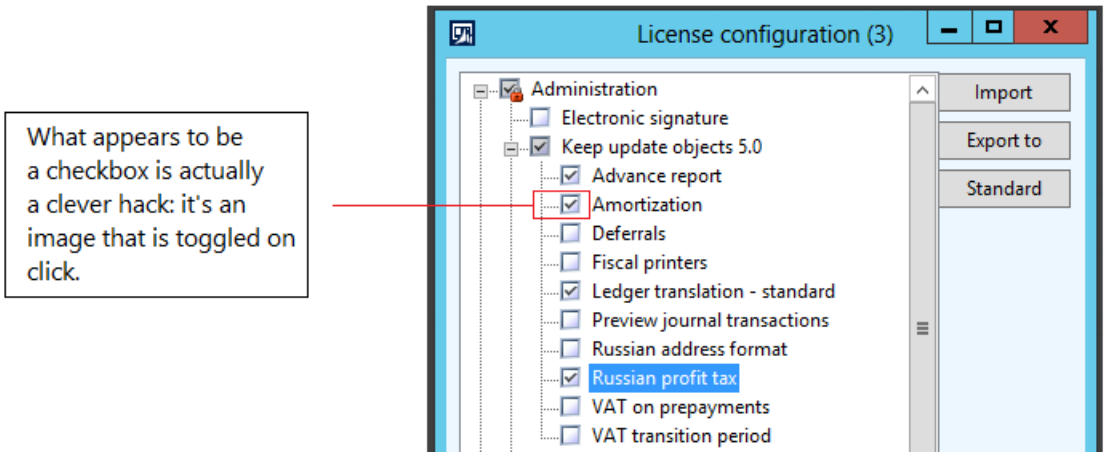
PROPERTY	VALUE	DESCRIPTION
DisplayLengthMode	Auto	Use the kernel's default value.
DisplayLengthMode	Fixed	Use the developer-defined **DisplayLength** value.
WidthMode	Auto	Use the kernel's default sizing behavior.
WidthMode	Fixed	Use the developer-defined **Width** value.
WidthMode	Column Width	Use size-to-parent behavior.
DisplayLength	N	The developer-defined control width, which is specified in characters. (Values are mapped to the sizing groups that are listed in the previous table.)
Width	N	The developer-defined control width, which is specified in pixels.

Check box support in tree controls

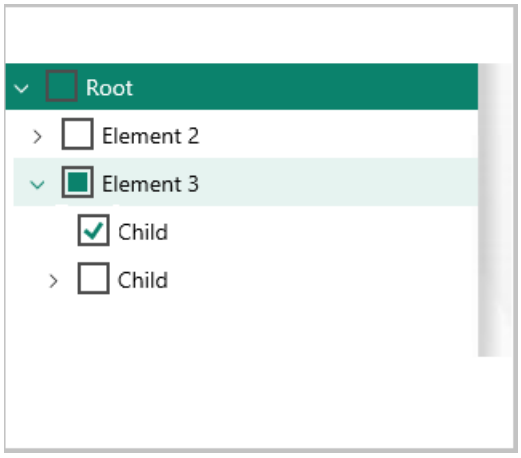
2/18/2021 • 4 minutes to read • Edit Online

This article is intended as a primer for using check box controls in the tree control. It's not a general "how to" for using tree controls.

Microsoft Dynamics AX 2012 includes several examples of tree controls that were enhanced so that they both show data in a tree hierarchy and let the user select one or more nodes by using check boxes. In Dynamics AX 2012, the tree control had no built-in support for check box controls. Instead, an image of a check box was added for each node in the tree control. The image state for each node was then toggled as the user clicked the check box.



The current version has greatly simplified the experience for the developer. Check box support is now built into the tree control.

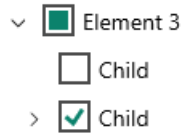


You no longer have to use images to include a check box, and you also don't have to explicitly set the state of the check box state when it's selected. The control doesn't use images, and the check box state is managed in the way that you would expect for a tri-state check box. Examples of tri-state check boxes can be found in most installation scenarios. When tri-state check boxes are used, if the user selects a parent node, all children of that parent also become selected. The check box interaction is independent of the node's expand/collapse functionality. When the parent node is collapsed (no children are visible), a check mark on the parent node indicates that all children are also selected. However, if one child of a parent that has multiple children isn't selected, the appearance of the parent node changes. The check box no longer contains a check mark but is filled in. This state is considered a partial check. Therefore, a parent node has three states:

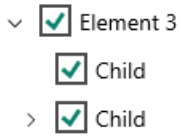
- Checked
- Unchecked
- Partial

If the user clicks the check box on a parent node that is in a partial state, the state of the parent and all its children changes to checked. (The parent node and all its child nodes are now selected.)

Parent node in a partial state

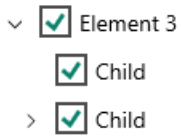


Parent node and all child nodes in a checked state after the parent node is selected

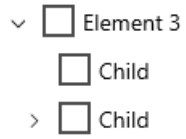


If the user clicks the check box on a parent node that is in a checked state, the state of the parent and all its children changes to unchecked. (The parent node and all its child nodes are now cleared.)

Parent node in a checked state

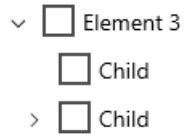


Parent node and all child nodes in an unchecked state after the parent node is cleared

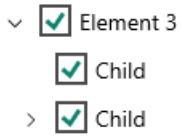


If the user clicks the check box on a parent node that is in an unchecked state, the state of the parent and all its children changes to checked. (The parent node and all its child nodes are now selected.)

Parent node in an unchecked state



Parent node and all child nodes in a checked state after the parent node is selected



A child node that has no children (in other words, a child node that isn't a parent itself) has only two states: checked and unchecked. A child node that is the only child in a checked state affects the state of its parent. If a child node is selected, the state of its parent changes to partial. **Note:** A single node in a tree also has a

“selected” state to indicate that it's the current node. This state differs from the checked state.

Tree controls that contain check boxes in Dynamics AX 2012

The following example is from SysConfiguration.

1. The program checks for the **mouseDown** event.
2. When the **mouseDown** event is detected, the program determines whether the user clicked the node or the image.
3. If the user clicked the image, the program toggles the image state.

None of this code is required for the current version.

```
int mouseDown(int x, int y, int button, boolean ctrl, boolean shift)
{
    int idx,f;
    FormTreeItem        parentNode, node;
    int                 parentMode;
    boolean             enabled;
    #FormTreeControl;
    [idx,f] = this.hitTest(x,y);
    parentNode  = this.getItem(this.getParent(idx));
    node        = this.getItem(idx);
    if (node)
    {
        if(parentNode)
        {
            if (element.enabled(parentNode.data()))
            parentMode = true;
        }
        else
            parentMode  = true;
        if ((f & #FTCHT_ONITEMICON) && parentMode)
        {
            if (!node.overlayImage())
            {
                enabled = (element.enabled(this.getItem(idx).data()) ? false : true);
                element.enabled(this.getItem(idx).data(), enabled);
                element.drawTree();
            }
            return 1;
        }
    }
    return super(x, y, button, ctrl, shift);
}
```

In the current version, you still set the selected state for scenarios where the user is presented with preselected nodes. Additionally, the developer can still set the state explicitly when the FormTreeItem is created. However, instead of specifying the current image, the developer now sets the **stateChecked** property on the FormTreeItem. If developers must know when the state of a check box changes, they can override the **checkedStateChanged()** method.

Basic check box use for tree controls in the current version

Make sure that the **Check Box** property on the modeled tree control is set to **Yes**. To explicitly set the state on a node, use the following code.

```
formTreeItem.stateChecked(FormTreeCheckedState::Checked);
formTreeControl.setItem(formTreeItem);
```

To interrogate a node for its current state, use the following code.

```
FormTreeItem formTreeItem = formTreeControl.getItem(formTreeControl.getSelection());
FormTreeCheckedState currentState;
if (formTreeItem != null)
{
    currentState = formTreeItem.stateChecked();
    switch (currentState)
    {
        case FormTreeCheckedState::Unchecked:
            /* unchecked */
            break;
        case FormTreeCheckedState::Checked:
            /*checked */
            break;
        case FormTreeCheckedState::Partial:
            /* parent has children checked */
            break;
        default:
            /* shouldn't get here */
            break;
    }
}
```

To react to or track the checked state of a node (**idx** is the node index), use the following code.

```
public void checkedStateChanged(int _Idx, FormTreeCheckedState _newState)
{
    super(_Idx, _newState);
}
```

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Filtering options

2/18/2021 • 8 minutes to read • Edit Online

This topic explains the filtering options that are available.

Introduction

Microsoft Dynamics AX 2012 offers the following filtering options.

FILTER OPTION	DESCRIPTION
Filter by grid	The user defines filter conditions in input fields below the grid column headers.
Filter by selection (filter by field)	The user selects a field value and uses that value as a filter condition.
Advanced filter	The user opens a dialog box that contains advanced filtering options (filter on columns, not on the form; join additional data sources; sort by multiple columns; and so on).

Finance and Operations offers the following filtering options.

FILTER OPTION	DESCRIPTION
Filter Pane	An inline pane that slides in from the left, and that contains multiple filter criteria that can be applied to the targeted content.
QuickFilter	A framework-provided filtering mechanism that can appear above any list or grid, and that provides fast single-column filtering.
Grid column filtering	The user can define filter conditions and perform single-column sorting by using a drop dialog that is opened from the grid column header.
Advanced filter/sort	For most advanced filtering scenarios, the migrated **Advanced filter** form from Dynamics AX 2012 is still available.

Filter expressions

One important difference between filtering in Finance and Operations apps and filtering in Dynamics AX 2012 is related to the way that query symbols are used when filter values are defined (for example, "*" to match 0 or more characters, or ".." to specify a range of values to match). In Dynamics AX 2012, these symbols are highly visible during the filtering experience. For example, for the filter by grid option, if a user selects the **contains** operator on a field, the system translates that operator by adding wildcard characters (*) to each end of the current expression. In the current version, the query symbols are implied by the selected operator and aren't injected into the user interface. This makes filtering more intuitive and simpler for users. For users who want to specify additional filter conditions by using specific query symbols, or users who must enter more complex

conditions, the **matches** operator is provided for each data type. For all other operators, the query symbols are interpreted as literals. For example, the filter condition "First name MATCHES A" finds all records where the first name starts with the letter A. However, the filter condition "First Name IS A*" finds records where the first name is literally equal to "A*." The following table shows how the client translates between Finance and Operations apps filter operators and Dynamics AX 2012 query syntax.

FILTER OPERATOR	FINANCE AND OPERATIONS APPS QUERY SYNTAX
Is “circle” / Is equal to “circle”	“circle”
Is not “circle” / Is not equal to “circle”	“!circle”
Is one of “circle”, “square”, “circlesquare”	“circle,square,circlesquare”
Contains “circle”	“*circle*”
Does not contain “circle”	“!*circle*”
Begins with “circle”	“circle*”
After “circle” / Greater than “circle”	“>circle”
Greater than or equal “circle”	“circle..”
Before “circle” / Less than “circle”	“<circle”
Less than or equal “circle”	“..circle”
Between “square” and “circle”	“square..circle”

Any query syntax that doesn't match the preceding templates is interpreted as the **matches** operator.

Note that the syntax for looking for blank values in a column remains the same as AX2012. With either the **matches** operator or the **is equal to** operator, you can type **""** to retrive rows with blank values for the current column. For example, **First Name IS ""** will finds all records where the first name is blank.

Filter Pane

The Filter Pane provides an easy-to-use interface for filtering full page lists. The Filter Pane is an inline pane that slides in from the left side of the screen and pushes the page content to the right, so that users can see the data that they want to filter. Users open this filter mechanism by clicking the system-defined **Show filters** button on the left side of the page. After it has been opened, the Filter Pane remains visible until the user goes to a new page, or until the user closes the Filter Pane by using the **Hide filters** button.

When is the Filter Pane available?

Currently, the Filter Pane is available for all forms except the following forms:

- Drop dialogs
- Dialogs
- Enhanced previews
- Lookups
- Form parts
- Parts

- Table of contents form type
- Forms that have no data sources

Note: The availability of the Filter Pane on particular forms and form types is evolving, so this list might change.

What data does the Filter Pane work on?

Because the Filter Pane is targeted at full page lists, it works only on the tables and fields that are directly joined (by inner/outer joins) to the first master data source on the form. This filtering mechanism isn't intended for filtering on secondary collections, or for filtering on other root data sources and their directly joined data sources. Other filtering mechanisms (QuickFilter, grid column filtering, and so on) are available to meet these other requirements.

What fields are initially shown in the Filter Pane?

Here is how the fields that are initially shown in the Filter Pane are selected:

1. All ranges/filters that currently exist on the query (only non-hidden filters/ranges are shown) are used.
2. If no ranges filters currently exist on the query, the fields from the primary index from the first master data source are used.
3. If there are no fields from the primary index from the first master data source, the TitleFields that are defined directly on the first master data source are used. If no TitleFields are defined, no default fields are shown. (Currently, if the first master data source extends another table (for example, table B), we don't show the TitleFields from table B. However, we plan to add that check in the future.)

Can I control the default fields that appear in the Filter Pane?

Developers can make sure that a particular field appears in the Filter Pane by adding an empty filter for that field to the query. For an example, see the **FmCustomer** form, which adds the filters post super() in form init(). Note that after an empty field has been added to guarantee that it appears in the Filter Pane, the fields in the Filter Pane will always be those that are explicitly on the query, and will never be the TitleFields or fields from the primary index on the first master data source.

I don't want users to be able to filter on a specific field or modify an existing filter. How do I accomplish this?

Developers can affect whether users can modify/add filters on certain fields by changing the status of the filters. The allowed values are in the **RangeStatus** enum:

1. **Open (default)** – The user can see and modify this filter.
2. **Locked** – The user can see the filter value but can't modify it. The user also can't add another filter on this column.
3. **Hidden** – The user can't see that there is a filter on this column. The user also can't add another filter on this column.

Can I control the fields that appear in the Add a filter field list in the Filter Pane?

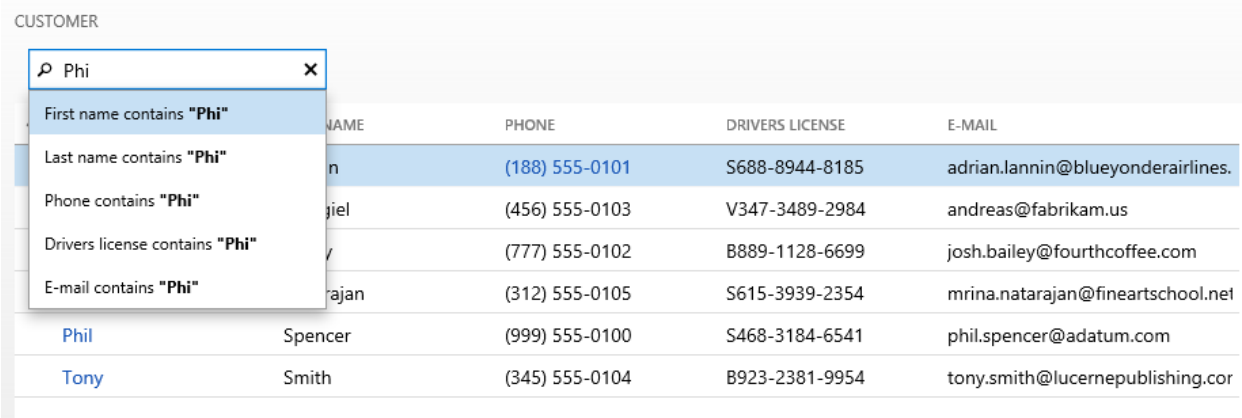
The fields that appear in the **Add a filter field** list are all the filterable fields from the query that involves the first master data source on the form. Therefore, developers can't control the fields that appear in this list. Usually, if you see unexpected fields or can't find the fields that you want to filter on, the fields that you're expecting are either on a different master data source (not the first) or on a child collection.

How is the Filter Pane used?

The Filter Pane is simple and straightforward to use. First, select a filtering operator in the list that is associated with each filter field. Note that the set of operators that appears depends on the data type of the field. Then enter an appropriate value for the filter condition, and click **Apply**. The form is updated based on the filter criteria that you specified.

QuickFilter

In Dynamics AX 2012, the QuickFilter was a framework control that was automatically added only to list pages. In Finance and Operations apps, the QuickFilter is now a modeled control that can be associated with any grid in the system. As the user starts to type, a column selector drop-down appears to guide the user toward the column that the filter will be applied to. The developer can also specify the default column for the QuickFilter. If no column is specified by the developer, the default column is the first field that can be filtered in the grid.



Why don't I have a column selector in my QuickFilter?

Column selectors are shown only for QuickFilters that are attached to grids. If you don't see a column selector, the most likely reason is that the **TargetControl** property on the QuickFilter is blank. This property must point to the grid that it should operate on. If the **TargetControl** property is set correctly, but you don't see a column selector, you might not have any filterable columns in your grid. In addition to non-text controls (such as images), controls that are bound to data methods aren't filterable.

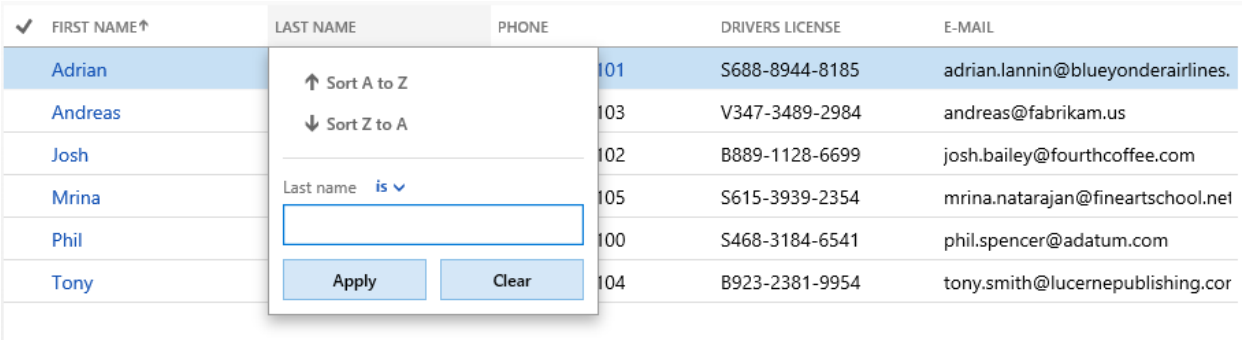
Can I use the QuickFilter to filter other collection controls (such as trees)?

Yes, you can use the QuickFilter to filter other collection controls, but you must manually wire up the filtering. Here are the general steps:

- Leave the **TargetControl** property blank.
- Override the **applyFilter()** method on the QuickFilter.
- Write code in that method to perform the desired filtering.

Grid column header filtering/sorting

In Finance and Operations apps, the grid filtering experience is more closely aligned with the experience in Microsoft Excel. When the user clicks a column header (for columns that can be filtered), a drop dialog appears, and the user can use it to filter the column. The filtering experience here mimics the filtering experience in the Filter Pane. Additionally, there are options to sort the grid based on the column that is currently selected.



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

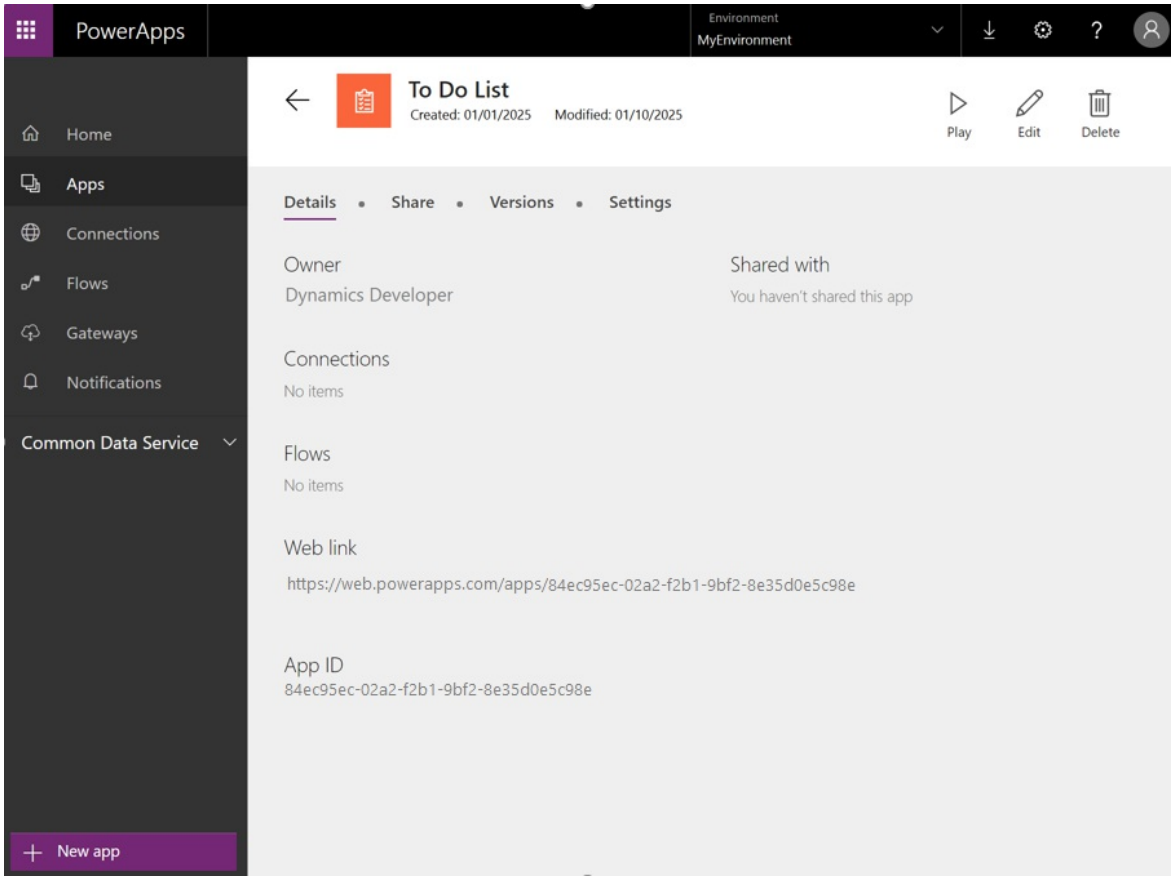
Power Apps Host control

2/18/2021 • 2 minutes to read • Edit Online

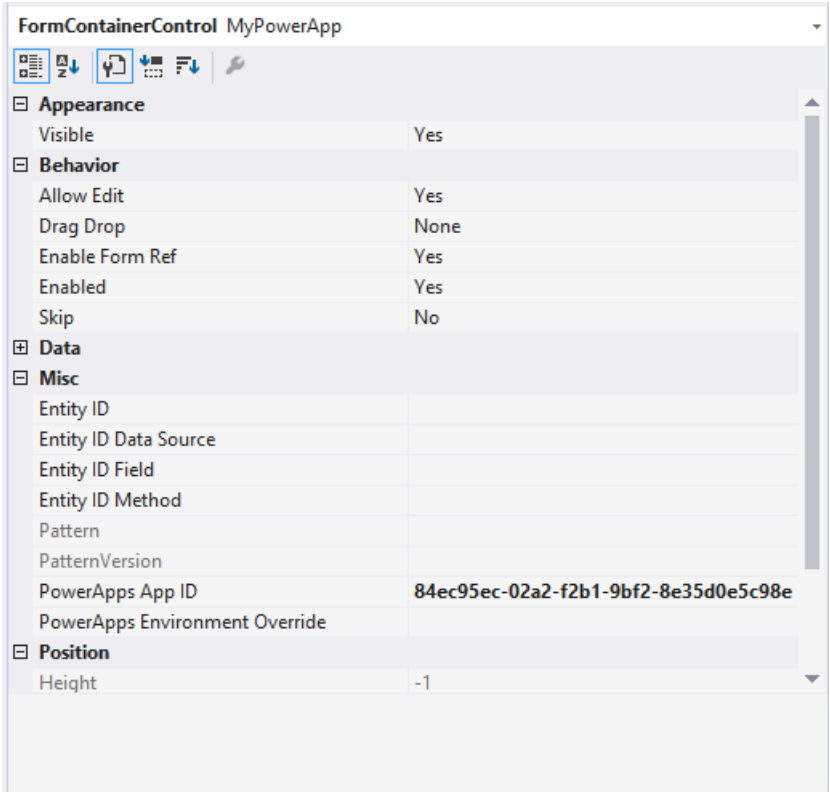
In Microsoft Power Apps, you can manage organizational data through apps that you created, or apps that someone else created and shared with you. Apps run on mobile devices such as phones or in a browser. Apps can also be embedded in Finance and Operations apps by developers using the Microsoft Visual Studio developer experience. To learn more about Power Apps, see https://powerapps.microsoft.com.

Host an app from Power Apps on a page

1. In Power Apps, find the web-based app that you want to host, and record or copy the **App ID** value.



2. In Visual Studio, open your project, and then, in the form designer, add an instance of a Power Apps Host control to your page.
3. In the **Properties** pane, enter the **App ID** value.
4. If your app shares or is linked to the current data source on your page, you can pass the ID of the primary or linked key field for the data that you want your app to show. In this case, provide the ID as the value of the **Entity ID**, **Entity ID Data Source/Field**, or **DataMethod** property. This value will then be passed to your app as a parmeter value, and your app must use that value to obtain the linked data.



5. In some cases, your app might be hosted in a development or sandbox Power Apps environment that is provided by Microsoft. In this case, you must supply that override URL as the value of the **Power Apps Environment Override** property.

Sizing is determined by the container that you put your control in. If you put your control in a form pattern that has limited available space, and your app has been designed to be larger than the available space, your embedded app will have scroll bars.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Code migration - Context menu code

2/18/2021 • 3 minutes to read • Edit Online

A programming model is required for context menus (shortcut menus). This topic outlines the process for migrating context menu code from Microsoft Dynamics AX 2012 to Finance and Operations. It also includes user experience (UX) guidelines for context menus.

In Dynamics AX 2012 and earlier versions, developers modified right-click context menus (shortcut menus) by using the **PopupMenu** class. This class relied on Microsoft Windows application programming interfaces (APIs) that aren't available on the web. In Finance and Operations, the **ContextMenu** APIs have been created as replacements to provide similar functionality. Previously, the **context()** and **showContextMenu()** method overrides were the entry points for modifying context menus for specific controls. These overrides typically contained code to add options to the context menu, and also to process the user's selection. The code for processing the user's selection used a wait model. Because these overrides are being removed and the wait model is being eliminated, developers must now create two overrides: **getContextMenuOptions()** to add options to the context menu and **selectedMenuOption()** to process the user's selection.

Migrate context menu code

Migration from the **PopupMenu** APIs to the **ContextMenu** APIs can be broken down into three main steps.

Step 1. Add a constant for each menu option that must be added

The old **insertItem()** method in the **PopupMenu** class returned an identifier for the menu option that was being added. This identifier was saved into a variable for future reference. Because developers will define the menu identifier, it's a good idea to define constants for each option to help with code readability.

- At the form level, add a constant for each menu option that is being added to the context menu. The value must be unique within each context menu. Note that you must modify the old variable name if it conflicts with another variable on the form or control.

Before

```
public void context()
{
    ...
    int listCreateRoot = listMenu.insertItem("@SYS5480");
    ...
```

After

```
[Form]
public class MainAccount extends FormRun
{
    ...
    public const int listCreateRoot = 1;
    ...
```

Step 2. Build the context menu

Construct the list of submenus and menu options, and add it to the control's context menu.

1. Add the **getContextMenuOptions()** method override on the control.
2. Create a new context menu and a list to hold the options that you will add to the menu:

- ContextMenu menu = new ContextMenu();
- List menuOptions = new List(Types::Class);

3. Add menu options to the list:
   - ContextMenuOption option = ContextMenuOption::Create(label,identifier);
   - menuOptions.addEnd(option);
4. Add the list of options to the menu.
   - menu.ContextMenuOptions(menuOptions);
5. Modify the **return** statement.
   - return menu.Serialize();

Step 3. Process the user selection from the context menu

1. Add the **selectedMenuOption()** method override on the control.
2. Move the **switch()** statement for processing options into this override.

Code example

This section illustrates the migration of a context menu from Dynamics AX 2012 to Finance and Operations. The **MainAccount** form is used as an example.

Original code

```
public void context()
{
    PopupMenu  listMenu        = new PopupMenu(element.hWnd());
    int        listCreateRoot  = listMenu.insertItem("@SYS5480");
    int        selectedMenu;
    selectedMenu = listMenu.draw();
    switch (selectedMenu)
    {
        case -1:
            break;
        case listCreateRoot:
            mainAccount_ds.create();
            break;
        default:
            break;
    }
}
```

Migrated code

```
// Define new form-level constant for each context menu option
public const int listCreateRoot = 1;
// Define new override on the control for building the context menu
public str getContextMenuOptions()
{
    str ret;
    ContextMenu menu = new ContextMenu();
    ContextMenuOption option = ContextMenuOption::Create("@SYS5480", listCreateRoot);
    List menuOptions = new List(Types::Class);
    // Add label and ID of menu option
    menuOptions.addEnd(option);
    menu.ContextMenuOptions(menuOptions);
    return menu.Serialize();
}
// Define new override on the control for processing the user selection
public void selectedMenuOption(int selectedOption)
{
    switch (selectedOption)
    {
        case -1:
            break;
        case listCreateRoot:
            mainAccount_ds.create();
            break;
        default:
            break;
    }
}
```

UX guidelines for context menus

As you migrate context menus, consider the following guidelines:

- The most important commands should be at the top of the menu.
- Remove commands that don't apply to the current state of the element that is the target of the right-click.
- Right-click is a shortcut. Therefore, the commands on the context menu should **always** be available in other places on the page.
- Don't create submenus of context menus. Submenus are hard to use and aren't touch-friendly.
- Limit the number of menu items to five.

Code migration - Mouse double-click logic

2/18/2021 • 3 minutes to read • Edit Online

In Finance and Operations, the **mouseDblClick()** override has been deprecated, and you will need to move this logic to new controls.

In Microsoft Dynamics AX 2012, the mouse double-click event was used for various reasons. For example, it helped provide a better user experience and provided an alternative way to run certain scenarios. Here are some examples of common usage patterns:

- Moving elements between two lists or tree controls
- Opening a new form to get more details about the selected field
- Running complex business logic
- Selecting a field in a lookup

Strategy overview

Before you begin to use the form, it's a good idea to fix all best practice warning messages that state, "The mouseDblClick control method has been deprecated and should not be used." Otherwise, the form might be useless, or it might work only in limited ways.

Migrate code from mouseDblClick() methods

As we mentioned earlier, there were various reasons for using the **mouseDblClick()** method in Dynamics AX 2012. This section explains how to migrate some of the most common scenarios.

Moving items between two lists controls

In Dynamics AX 2012, a mouse double-click was often used in List Panel scenarios, where two list controls appeared side by side. Often, when a user double-clicked an item in one list control, that item was moved to the second list control. Migration of this **mouseDblClick()** scenario involves alignment to the List Panel pattern. You have two options for migrating this usage pattern:

- Use the **SysListPanel** class itself, which provides the logic and the buttons for moving items between the two list controls.
- If you can't use the **SysListPanel** class (because the lists aren't ListViews, or the class isn't appropriate for the given situation), you can manually model the controls by following the List Panel sub-pattern. This pattern includes buttons for moving items between lists, but the developer will have to add the correct logic to make these buttons work.

Opening a new form

In another common usage pattern in Dynamics AX 2012, the user double-clicked a field to open a new form that showed more detailed information about that field. You have several options for migrating this usage pattern:

- Use a single-click to open a backing form that shows more details about a field. This functionality is automatically implemented for many fields that are based on table relations, and you can implement it manually by overriding the **jumpRef()** method on a control. The preferred migration route is to move the code from **mouseDblClick()** into a **jumpRef()** override, so that the navigation will be aligned with other fields in the system.
- Model a new button on the form, and move the logic from the **mouseDblClick()** method into the button's **clicked()** method. You should use this approach only for non-input field controls (for example, a Tree

control) in which a **jumpRef()** override doesn't exist.

- Add a right-click context menu (shortcut menu) option. However, note that UX guidelines specify that the commands on context menus should **always** be available in other locations on the page. A **View details** command is automatically added to the right-click context menu for controls that have an overridden **jumpRef()** method. Therefore, this approach should be used only as an optional addition to the previous migration route (modeling a new button). For more information about how to add context menu options, see Code migration - Context menu code.

Moving logic to a button control

In another common usage pattern in Dynamics AX 2012, the double-click caused complex business logic to run. For this scenario, the preferred migration route is to model a new button on the form, and then move the logic from **mouseDblClick()** into the new button's **clicked()** method.

Selecting a field in a lookup

In some custom lookups in Dynamics AX 2012, code was added so that the user could double-click a row in a grid (or an element in a tree) to select the value and close the lookup. For this scenario, the recommended migration route is to add a **Select** button at the bottom of the lookup form to enable record selection.

UX guidelines

As you migrate mouse double-click methods, you should consider the following guidelines:

- To move items between controls, use the **SysListPanel** class or the ListPanel pattern whenever possible.
- When you add buttons to replace mouse double-click logic, put the button as close as possible (contextually) to the control.
- In some cases, you might have to redesign the form to accommodate the logic that was present in the **mouseDblClick()** method.

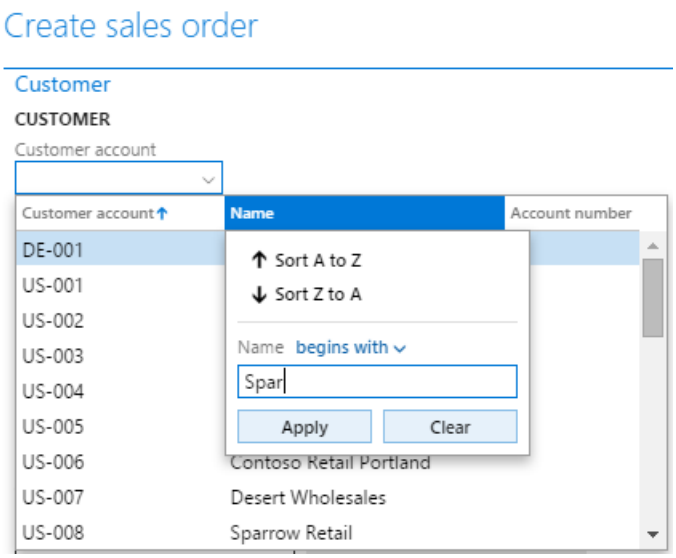
Contextual data entry for lookups

2/18/2021 • 14 minutes to read • Edit Online

In data entry scenarios, it is common for a user to attempt to identify an entity in terms of some more descriptive or natural language attribute if that entity is formally identified by a synthetic key, such as a number sequence. The contextual data entry feature allows users to type in either the synthetic key or a more descriptive attribute directly into a lookup field. This page explains how contextual data entry works and also provides implementation details and tips for developers who want their lookups to have this behavior.

Introduction

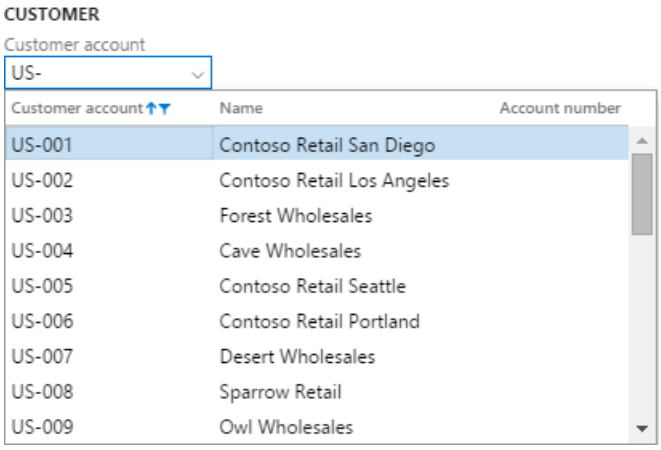
In data entry scenarios, it is common for a user to attempt to identify an entity in terms of some more descriptive or natural language attribute if that entity is formally identified by a synthetic key, such as a number sequence. A user will typically attempt to enter an **Account Name** instead of an **Account ID** for the **Customer Account** when creating a Sales Order. This is because most interaction with a customer is done using their actual name instead of some synthetic identifier. Unfortunately, any user's attempt to enter an **Account Name** will fail because the **Customer account** control's underlying foreign key relates to a field that is a synthetic key —a number sequence—and Dynamics AX 2012 (and older) will always attempt to validate the entered value directly. Therefore, if the **Account ID** was unknown to the user, the user would be forced to perform some type of searching step, such as opening the **Customer account** control's lookup and filtering on the **Account Name** column to identify the correct **Account ID** (see the image below).



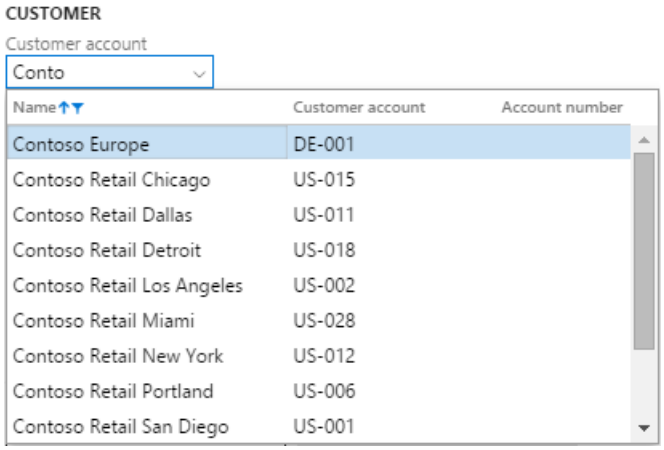
This user experience is not optimal and is being addressed by data entry efficiency and productivity. The platform adds initial support for contextual data entry, where the system automatically attempts to understand whether the user's entered data is in the context of the key field or some other more descriptive or well-understood field, and handle it appropriately. **For the remainder of this document, we'll generically refer to these types of fields as ID (synthetic) and NAME (descriptive) fields, respectively.**

Contextual lookup forms

Just like keyboard data entry, all system-generated lookup forms are also now contextual, meaning that filtering and sorting occur in the context of the data the user has entered. Using the create a Sales Order scenario as an example, the user will see the lookup shown below if an ID is entered.



If a NAME is entered, then the user will see the following lookup. Notice how the NAME column is moved first in the Grid, and how the lookup is sorted and filtered upon when the user's data is in the context of NAME.



Contextual data entry implementation details

Behavior

In the context of the Sales order create scenario mentioned above, the contextual data entry feature will allow the user to be able to freely type in either the ID or NAME without performing any laborious search process. In detail, the following behaviors will occur:

1. If the user enters a complete ID reference, the value will be taken directly.
2. If the user enters a complete and unique NAME reference, the value will be automatically translated into an ID and then processed.
3. If the user enters a non-complete ID or NAME reference (such as *Micro* instead of *Microsoft*), but it still uniquely matches either ID or NAME via a BEGINS WITH predicate, then the value will be translated into its complete ID and then processed.
4. If the user enters a non-complete ID or a non-unique NAME and there are multiple matches, then a *disambiguation* lookup will be presented to the user to select which value was actually intended.

See Appendix A for more detailed sample scenarios of contextual data entry.

Prerequisites

To maintain functional correctness and reasonable performance, the following constraints were added to the application of the behaviors described in the previous section:

1. **Title Field 2** is the NAME field**.
2. The NAME field must either be covered by an index ***OR*** belong to a Table whose *Cache Lookup* property is set to *EntireTable*. All contextual lookup behavior will be disabled if this requirement is not met for performance reasons. **NOTE: An index should only be added for NON TRANSACTIONAL tables**

because of index maintenance costs. Also note that you will **likely want to mark this index as non-unique** (Allow Duplicates = Yes).

3. If a control is using a custom lookup form (such as SysTableLookup; FormHelp on an EDT) then the disambiguation behavior described previously will not be turned on by default. This is because these custom lookup forms (and even surrounding modified and lookup method overrides) can and will do advanced things such as presenting a dialog, which are not desirable in the context of contextual lookups.

Handling custom lookup forms requires additional knowledge and will be covered in its own section.

Programming model additions

The behaviors and rules expressed in *Listings 1* and *2* are contained primarily by a new X++ class called *FormControlAmbiguousReferenceResolver*. *FormControlAmbiguousReferenceResolver* uptake in application code will be necessary in more advanced scenarios. Its use will be described later in the document. In addition to the *FormControlAmbiguousReferenceResolver* class, a new control override called *resolveAmbiguousReference* has been added. R*esolveAmbiguousReference* acts as a hook point in the system for translating what the user typed into a value that the system is expecting. The basic flow is as follows:

1. The user enters a value into a control and removes focus.
2. An interaction is sent from the client to the server, indicating that a new value has been entered. The appropriate command is executed on the server.
3. Before the command attempts to process the value entered by the user, it makes a call to *resolveAmbiguousReference* to give the system a chance to translate the value into the expected domain.
4. The super implementation of resolveAmbiguousReference creates an instance of *FormControlAmbiguousReferenceResolver* which executes the rules described above.

The value returned from *resolveAmbiguousReference* is used for the remainder of the command's execution. Validate() and modified() operate against the returned value.

Standard lookup uptake

Add an index that covers TitleField2

TitleField2 defines the default definition of NAME. In order to enable ID and NAME contextual data entry, *TitleField2* must be either indexed OR belong to a table with *CacheLookup* set to *EntireTable*. If the table containing *TitleField2* does not yet define an index covering *TitleField2* **and, importantly, the table does not have a high volume of CUD (Creates/Updates/Deletes*)**, then add a **non-unique** index (Allow Duplicates = Yes) covering *TitleField2*. This will cause the system to start executing the contextual data entry behavior, except for the custom lookup limitation described in the Prerequisites section. *Adding an index on high-volume transactional tables may incur a noticeable performance penalty due to index maintenance costs.

Enable disambiguation behavior for custom lookup scenarios

Custom lookup implementations can provide advanced or non-typical behaviors, such as presenting dialogs. Therefore, the system disables the default disambiguation behavior when a custom lookup scenario is detected. To opt into the default disambiguation behavior, override the *resolveAmbiguousReference* method (as shown below) **on the control hosting the lookup**. Note that the second parameter to the *resolveAmbiguousReferenceForControl* call is what overrides the default behavior of not performing disambiguation for custom lookup scenarios.

```
public str resolveAmbiguousReference()
{
    FormControlAmbiguousReferenceResolver::resolveAmbiguousReferenceForControl (this, true);
}
```

Make custom lookup forms contextual

As mentioned earlier, all system-generated lookup forms automatically consider the context of the data entered into their host control. This includes most lookup forms generated via *SysTableLookup*. Modeled custom lookup forms, by their nature, cannot be fully-handled by the system and must be modified to match the behavior and visuals of contextual lookups forms.

1. If the data contained by the host control is in the context of ID, then:
    a. Make the ID column first in the Grid.
    b. Sort and filter by ID.
2. If the data contained by the host control is in the context of NAME, then:
    a. Make the NAME column first in the Grid,
    b. Sort and filter by NAME,

The following scenarios illustrate some custom lookups, along with the recommendation for how to enable contextual data entry in these cases.

Scenario 1: Custom lookup defined via the FormHelp property on an EDT

Custom lookups defined via FormHelp (even though modeled) still go through normal kernel-based lookup generation routines. Therefore, the kernel still has hooks to make some changes to the lookup form. Specifically, the lookup system has enough information to apply the correct filters and sorts; however, it is NOT known which controls should be moved in the lookup's grid. (While an educated guess could be made based on bindings, that guess may be incorrect in more advanced lookup form designs.) If your custom lookup form is leveraging the `SysTableLookup::filterLookupPreRun` and `SysTableLookup::filterLookupPostRun` methods, then uptake the (new) optional parameters on `filterLookupPostRun` to have the NAME control moved automatically, as shown.

```
public class MyCustomLookupForm extends FormRun
{
    public void run()
    {
        FormStringControl lookupHostControl = SysTableLookup::getCallerStringControl(this.args());
        boolean isFiltered = SysTableLookup::filterLookupPreRun(lookupHostControl, ID_Control,
FormDataSourceToFilter);

        super();

        SysTableLookup::filterLookupPostRun(isFiltered, lookupHostControl.text(), ID_Control,
FormDataSourceToFilter,
            new FormControlAmbiguousReferenceResolver(callingControl), NAME_Control);
    }
}
```

If your lookup form isn't using the *SysTableLookup::filterLookup** methods, and you don't want to uptake those methods, then you can simply add a control move as shown below.

```
public class MyCustomLookupForm extends FormRun
{
    public void init()
    {
        super();
        this.applyControlOrdering();
    }

    private void applyControlOrdering()
    {
        FormControl callerControl = SysTableLookup::getCallerControl(this.args());
        if (FormControlAmbiguousReferenceResolver::isControlValueMappedToAlternativeField(callerControl))
        {
            Grid.moveControl(ID_Control.id(), NAME_control.id());
        }
        else
        {
            Grid.moveControl(NAME_Control.id(), ID_Control.id());
        }
    }
}
```

Scenario 2: Override of lookup method manually launching a form

Unlike Scenario 1, lookup forms launched by completely manual mechanisms, such as the class factory, have no kernel hooks. Therefore, it is the responsibility of the lookup form to adhere to the contextual data entry behaviors. The easiest way to do this is to leverage the SysTableLookup::filterLookup* methods (similar to Scenario 1) except include one additional parameter to indicate that sorting should also be maintained. An example is shown below.

```
public class MyCustomLookupForm extends FormRun
{
    public void run()
    {
        FormStringControl lookupHostControl = SysTableLookup::getCallerStringControl(this.args());
        boolean isFiltered = SysTableLookup::filterLookupPreRun(lookupHostControl, ID_Control,
FormDataSourceToFilter);

        super();

        SysTableLookup::filterLookupPostRun(isFiltered, lookupHostControl.text(), ID_Control,
FormDataSourceToFilter,
            new FormControlAmbiguousReferenceResolver(callingControl), NAME_Control, true);
    }
}
```

Advanced lookup uptake

Scenario 1: Overriding ID and NAME bindings

If you want to use a set of fields other than what is chosen by default, you must manually construct an instance of *FormControlAmbiguousReferenceResolver* and provide the optional parameters representing the custom bindings. This specialized instance must be used in an override of *resolveAmbiguousReference* and in a custom lookup form (including *SysTableLookup*, which also accepts an instance of *FormControlAmbiguousReferenceResolver*). A custom binding cannot currently be specified in kernel-generated lookups. Methods currently accepting custom ID and NAME bindings:

1. FormControlAmbiguousReferenceResolver
    - Constructor
    - resolveAmbiguousReferenceForControl

- surrogateFKHelperForAlternativeFieldMapping
- isControlValueMappedToAlternativeField

Here's an end-to-end example of how to provide custom bindings.

```
[Control Hosting Lookup]
public str resolveAmbiguousReference()
{
    return FormControlAmbiguousReferenceResolver::resolveAmbiguousReferenceForControl(
        this, true, AbsoluteFieldBinding::construct(IDField, Table),
        AbsoluteFieldBinding::construct(SomeOtherNAMEField, Table));
}

[Custom Lookup Form]
public class MyCustomLookupForm extends FormRun
{
    public void run()
    {
        FormStringControl lookupHostControl = SysTableLookup::getCallerStringControl(this.args());
        boolean isFiltered = SysTableLookup::filterLookupPreRun(lookupHostControl, ID_Control,
FormDataSourceToFilter);

        super();

        SysTableLookup::filterLookupPostRun(isFiltered, lookupHostControl.text(), ID_Control,
FormDataSourceToFilter,
            new FormControlAmbiguousReferenceResolver(callingControl,
AbsoluteFieldBinding::construct(IDField, Table),
            AbsoluteFieldBinding::construct(SomeOtherNAMEField, Table)), NAME_Control, true);
    }
}
```

Scenario 2: Custom resolution logic

It’s possible to use custom resolution logic by overriding resolveAmbiguousReference and leveraging something other than FormControlAmbiguousReferenceResolver. Note that this logic needs to be common to the hosted lookup form so that keyboard and lookup-based entry stay in sync.

```
public str resolveAmbiguousReference()
{
    // In this sample, allow “looser” data entry by simply picking the first record that matches, if any.
    CLI_Job _job;
    str mappedValue = this.text();
    if (strLen(mappedValue) > 0)
    {
        select firstonly _job order by _job.Title where _job.Title like mappedValue + “*”;
    }

    if (_job.RecId)
    {
        mappedValue = _job.Title;
    }

    return mappedValue;
}
```

Appendix Detailed usage scenarios for contextual data entry

For the scenarios, assume there is a table called "TableA" with PK field "ID" and index field "Name", with the FK we're trying to enter that is related to the ID (the user ultimately needs to pick an ID). Note that any algorithms that depend on like/begins with are assuming string fields. We won't be able to provide high fidelity resolution behavior on, for example, integral types.

Scenario 1: User enters a valid ID of "1234" The super() implementation of resolveReference first queries against TableA.ID with the appropriate predicate. The query finds a single record, and returns the user's entered value to be further processed by validate and modified. Validation passes and the user sees "1234" in the UI.

Scenario 2: User enters an invalid ID of "4321" The super() implementation of resolveReference first queries against TableA.ID. The query does not find any records, so a second query is performed against the Name field (SELECT TOP 2 FROM TableA WHERE TableA.Name LIKE "4321%"). Still, no record is found, so "4321" is passed through to validation, which fails. The user sees "4321" in the browser as well as a validation error.

Scenario 3: User enters a valid Name of "ACME" The super() implementation of resolveReference first queries against TableA.ID. The query does not find any records, so a second query is performed against the Name field (SELECT TOP 2 FROM TableA WHERE TableA.Name LIKE "ACME%"). This query does find a single record (unique reference), so the lookup automatically returns the corresponding TableA.ID. Validate and modified continue executing in the context of that value. Validation passes, and ultimately the user sees the ID value for ACME in the browser (for example, ACME would switch to 1234 in the browser).

Scenario 4: User enters an invalid Name of "ACNE" The super() implementation of resolveReference first queries against TableA.ID. The query does not find any records, so a second query is performed against the Name field (SELECT TOP 2 FROM TableA WHERE TableA.Name LIKE "ACNE%"). This query does not find any records, so the lookup passes ACNE through to validation, which fails. The user sees "ACNE" in the browser as well as a validation error.

Scenario 5: User enters an ambiguous Name of "ACME" In this case, assume there are two records in the database: one with Name "ACME W" and another with "ACME E". The super() implementation of resolveReference first queries against TableA.ID. The query does not find any records, so a second query is performed against the Name field (SELECT TOP 2 FROM TableA WHERE TableA.Name LIKE "ACME%"). This query finds two records, so it cannot make any further assumptions. A disambiguation lookup is presented to the user showing "ACME W" and "ACME E" as choices. The user picks "ACME E". resolveReference then takes the records selected by the user and redirects it to the ID of "ACME E". Validate and modified continue execution in the context of the ID of "ACME E". The browser ultimately displays the ID of "ACME E" (for example, 1234).

Scenario 6: User enters an ambiguous Name of "ACME" and doesn't make a choice in the disambiguation lookup In this case, assume there are two records in the database: one with Name "ACME W" and another with "ACME E". The super() implementation of resolveReference first queries against TableA.ID. The query does not find any records, so a second query is performed against the Name field (SELECT TOP 2 FROM TableA WHERE TableA.Name LIKE "ACME%"). This query finds two records, so it cannot make any further assumptions. A disambiguation lookup is presented to the user showing "ACME W" and "ACME E" as choices. The user doesn't make a selection from the lookup. Therefore "ACME" is passed through to validate and modified. Validation fails and the user is presented with a validation failure message. The browser still displays a value of "ACME".

Scenario 7: User enters a "valid" ID of "12" and presents the lookup form Prior to presenting the lookup, the system queries against TableA.ID (SELECT TOP 1 FROM TableA WHERE TableA.ID LIKE '12%'). The query finds a record and therefore assumes the user must be operating in the context of ID. It presents the lookup, filtering and sorting by ID.

Scenario 8: User enters an invalid ID of "4321" and presents the lookup form Prior to presenting the lookup, the system queries against TableA.ID (SELECT TOP 1 FROM TableA WHERE TableA.ID LIKE '4321%'). The query does not find a matching record and therefore assumes the user is entering a Name. The lookup is presented as filtered and sorted by Name (no records shown in this case).

Scenario 9: User enters a "valid" Name of "AC" and presents the lookup form Prior to presenting the lookup, the system queries against TableA.ID (SELECT TOP 1 FROM TableA WHERE TableA.ID LIKE 'AC%'). The query does not find a matching record and therefore assumes the user is entering a Name. The lookup is presented as filtered (those records matching "begins with AC") and sorted by Name in alphabetical order.

Scenario 10: User enters an invalid Name of "EM" and presents the lookup form Prior to presenting the lookup, the system queries against TableA.ID (SELECT TOP 1 FROM TableA WHERE TableA.ID LIKE 'EM%'). The query does not find a matching record and therefore assumes the user is entering a Name. The lookup is presented as filtered and sorted by Name. No records are found and therefore the user is presented with an empty lookup.

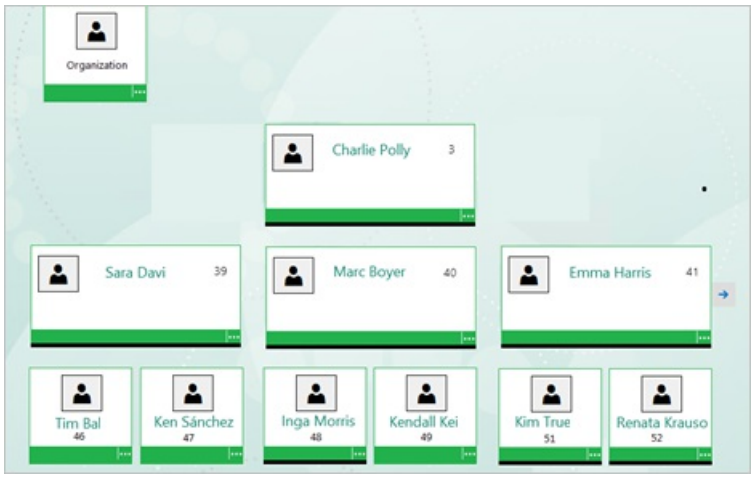
HierarchyViewer control

2/18/2021 • 3 minutes to read • Edit Online

This article provides information about the HierarchyViewer control, which lets you represent hierarchical relationships for people, products, or organizations.

Overview

The HierarchyViewer control lets you represent hierarchical relationships for people, products, or organizations. It's used primarily as a graphical means to help you understand hierarchical relationships in a traditional top-down manner, and as way to navigate to the entity that is represented by the focused node. The HierarchyViewer control lets you walk through deeply nested, multilevel content in a compact space. The control expands and collapses nodes to control the parts of the tree structure that are shown. Because it's an unbound control, the HierarchyViewer data is managed by an abstraction class and is used primarily as a way to visualize data in a simple tree relationship. For hierarchy data in a traditional tree, there is a standard tree control.



The HierarchyViewer control shows, at most, three levels on a single branch at any time. A breadcrumb trail shows the path of parents down the current tree branch. The top level shows the current top node and will always have one member. This member isn't necessarily the root. The second level, the child node, can have an indefinite number of member nodes. By default, three of these member nodes are shown on each page. The number of member nodes that is shown can be set by using the **Number of children** property. The control will page right and left through the members of the child level. The last level, the grandchild node, can have an indefinite number of member nodes. The number of visible member nodes is controlled by the **Number of Grandchildren** property. The HierarchyViewer control will page up and down through members of the grandchild level. The interactive display of nodes requires no business logic.

Business logic interaction

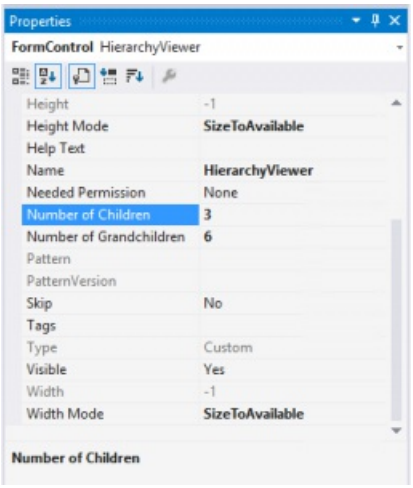
The HierarchyViewer control offers data visualization and navigation. The HierarchyViewer control is a read-only control. It can be used to select an entity (employee, product, or organization), and corresponding data can then be managed though other display and input fields on the form, outside the HierarchyViewer control. This is accomplished by a selection event that is raised on each user focus on each node.

```
public void init(){
    …
    // HierarchyViewer is the auto-declared name for the control.
    // handleNodeSelected is your event handler.
    HierarchyViewer.notfiyNodeSelected += eventhandler(element.handleNodeSelected);
}
public void handeNodeSelected(int _nodeId)
{
    // do something
}
```

Authoring a HierarchyViewer instance

To create a HierarchyViewer instance:

1. In the form designer, add an instance of HierarchyViewer to your form.
2. In the **Properties** pane, accept the default number of visible children and grandchildren, or set new values.



The HierarchyViewer control is primarily a visually interactive way of navigating or interrogating nodes in a static manner. The HierarchyViewer control isn't bound to a data source. Instead, the control is managed by a corresponding controller class that extends the base **HierarcyDesignerBase**. You initialize that class with data, and bind to the control instance and the visible fields of the HierarchyViewer node.

A typical use of the control is to initialize a server-side “in-memory” map of the hierarchy and then dynamically update the control as the user interactively explores the hierarchy by using load-on-demand semantics.

```
public void init()
{
    HcmPositionNode node;
    nodeMap = new Map(Types::Int64, Types::Class);
    hierarchyMap = new Map(Types::Int64, Types::Int64);
    firstNodeId = 0;
    // Initialize the organization node
    node = HcmPositionNode::newParameters(this.getNextNodeId(), HcmPositionNodeType::Enterprise, -1, 0,
"@SYS317690", "");
    rootNode = node;
    if (selectedNode == null)
    {
        selectedNode = rootNode;
    }
    this.insertNewNodeAndUpdateParent(node);
}
```

Override the **applyBuild** method of the control. This is where you will pass the instance of your controller class.

```
    public void applyBuild()
{
    super();
    YourControllerClass controller = new YourControllerClass();
    this.initControl(controller);
}
```

You don’t have to populate the whole node structure. Instead, you can populate nodes on demand.

```
public void initHcmPositionFromCurrentNode(HcmPosition _hcmPosition)
protected void insertNewNodeAndLoadDescendants(HcmPositionNode _node, int _depth, HcmPositionNode
_parentNode = null, Common _common = null)
protected void loadNodeDescendants(HcmPositionNode _node, int _depth, Common _common = null)
```

Changing node visuals

You can't change node visuals. The intended design is for the control to offer a set of unbound controls that have a default layout that you can manipulate by using an **ExtendedStyle** property to offer a predefined set of alternatives that the author can select.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Lookup controls

2/18/2021 • 3 minutes to read • Edit Online

This article discusses how to enable lookup behavior on controls. It also discusses how to create multi-select lookups and outlines lookup scenarios that are no longer supported.

Enabling lookup behavior in controls

Controls bound to an Extended Data Type

Controls with their Extended Data Type property set (no FormDataSource in play) will have a lookup under the following conditions:

1. If the EDT has its Table Relations or Table References node populated.
2. If the FormHelp property is set (custom lookup); doesn't require rule #1 to be true.
3. If the control has lookup or lookupReference overridden. Note, this rule also applies to fully unbound controls (no EDT, field, or data method). This includes overrides via registerOverrideMethod and others.

Controls bound to a form data source

Controls that are bound to a data source will have a lookup under the following conditions: **Field bound**

1. "lookup" or "lookupReference" (Reference Controls) methods are overridden.
    a. If the FormDataSource field has lookup or lookupReference overridden.
    b. If the control has lookup or lookupReference overridden.
        - This includes overrides via registerOverrideMethod and others.
2. If the field has an EDT, then rule #2 from the "Controls bound to an Extended Data Type" section applies.
3. If the bound field maps to a relation per DBFGetRef rules.
    a. High level rules:
        a. If there is an EDT relation backing the field, with the Table Relations node populated and Ignore EDT Relations is false on the field, the relation is used (has a lookup).
        b. If there is a relation mapping to the field and any fixed field link conditions are satisfied, the relation is used (has a lookup).
            a. Validate must be "Yes".
        c. Note the special case of migrated EDT relations which occur when:
            a. Field is backed by an EDT with the Relations node populated.
            b. Field is backed by a TABLE relation with the "EDTRelation" set to Yes.
            c. The table relation link has the SourceEDT set to the appropriate EDT.
        d. You can also have cases where IgnoreEDTRelation is set to true on a field, in which case a lookup will occur only if rule #3.1.2 of this section is true.

Data method bound

1. If the return type of the data method is an EDT, then rules #1 and #2 from the "Controls bound to an Extended Data Type" section apply.
2. If the control has lookup or lookupReference overridden.
    - This includes overrides by using registerOverrideMethod.

Multiselect lookups

Available system forms for building multi-select lookups

There are currently two system forms for creating multi-select lookups:

- SysLookup – Multiselect based on an Enum.
- SysLookupMultiselectGrid – Multiselect based on a collection of data.

What happened to the SysLookupMultiselect form?

SysLookupMultiselect was marked for deprecation in Microsoft Dynamics AX 2012 and has been removed. Any use of this form for multiselect lookup scenarios should be migrated to use SysLookupMultiselectGrid. For an example, see the form tutorial_LookupMultiSelectGrid.

Unsupported lookup scenarios

Creating multiple lookup forms when the lookup button is used

An error may occur if you create multiple lookup forms when the lookup button is used. For example, overriding the 'lookup' method and creating a new lookup form, but also calling 'super' (which will create another lookup form).

Using SelectedControl() to determine which control is hosting a lookup

Using SelectedControl() to determine which control is hosting a lookup is unsupported. While it may work in some cases, it will fail in others. For example, in disambiguation lookups, no control is selected on the parent form since the act of leaving the control is what triggers a disambiguation lookup. As an alternative to using SelectedControl(), there are a few other ways to retrieve the control that is hosting the lookup:

- Check the 'selectTarget' of the lookup form.

```
FormStringControl selectTarget = formRun.selectTarget();
```

- Check the 'callerFormControl' on the lookup form args. Note that SysTableLookup::getCallerControl(Args args) encapsulates that call.

```
FormStringControl argsCallerFormControl = args.callerFormControl();
```

Note that the selectTarget and callerFormControl will be set automatically if the lookup form instance is spun up automatically by the kernel. If the form instance is created in app code, these can be set manually as shown below.

```
public void lookup()
{
    Args args = new Args(formStr(<formName>));
    args.caller(element);
    args.callerFormControl(this);
    FormRun formRun = classfactory.formRunClass(args);
    formRun.init();
    this.performFormLookup(formRun);
}
```

Creating a slider dialog (instead of a lookup form) when the lookup button is used

Lookup controls should open lookup forms when the lookup button is used (not slider dialogs or other kinds of forms). The first reason for this is product consistency. The second and more important reason is that opening a slider dialog from a lookup is incompatible with the new type-ahead feature in lookups.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

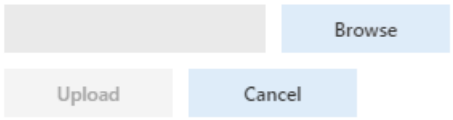
File upload control

2/18/2021 • 4 minutes to read • Edit Online

This topic provides information about the file upload control. This control lets users upload files.

Overview

The file upload control lets users upload a file. It also lets developers control the upload process and manage the file that is uploaded, based on their requirements.



The file upload control can have three styles. You control the style by using the **Style** property.

- The **Standard** style shows the file name field together with **Browse**, **Upload**, and **Cancel** buttons.
- The **Minimal** style shows only the **Browse** button.
- The **MinimalWithFileName** style shows the file name field and the **Browse** button.

The **FileTypesAccepted** property of the file upload control lets you limit the types of files that users can upload. The file types that users can upload are primarily controlled by the associated upload strategy. The **FileTypesAccepted** property on the file upload control should be used only if further restrictions are required. If the upload control tries to specify file types that are restricted by the upload strategy, the **Browse** button becomes unavailable.

ALLOWED FILE TYPES	ALLOWED FILE TYPES FROM THE UPLOAD STRATEGY	FINAL RESULT
".jpg,.png"	".jpg,.png,.gif,.txt"	".jpg,.png"
"image/png"	"image/*"	"image/png"
"image/*"	"image/png"	The **Browse** button is unavailable.
".jpg,.png,.gif,.txt"	".jpg,.png"	The **Browse** button is unavailable.

You can use the **OnBrowseButtonClicked**, **OnUploadAttemptStarted**, and **OnUploadCompleted** overrides to hook into the various stages of the file upload process. You can also create custom file upload strategies and associate them with a file upload control by using the **FileUpload Strategy Class** property.

Design classes

There are two base classes that developers can work with for the file upload control:

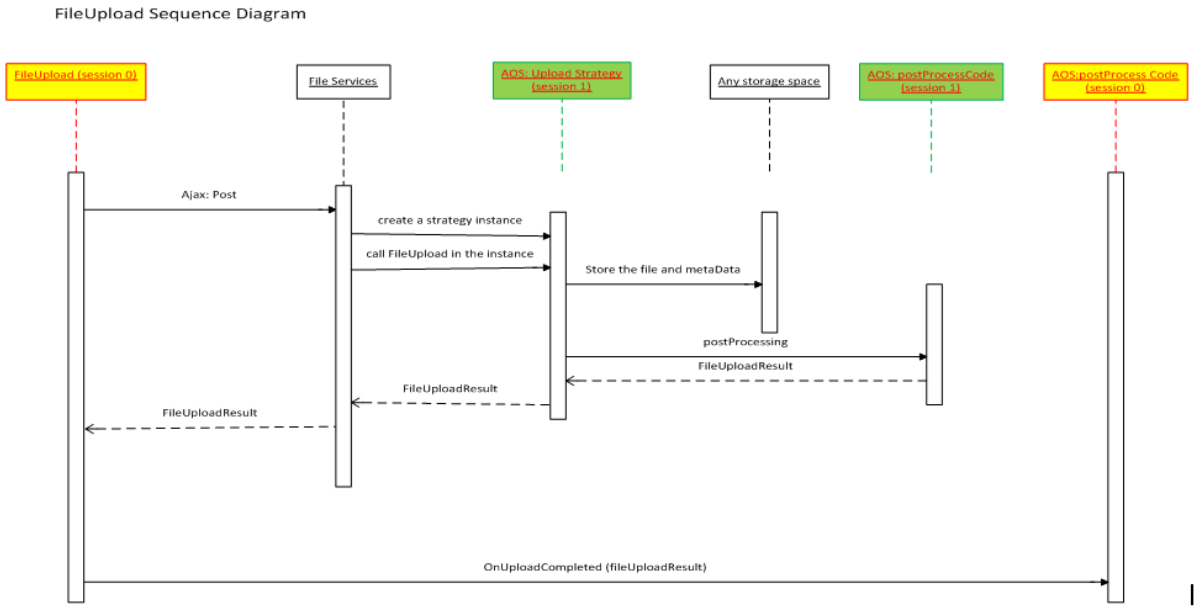
- **Upload strategy class** – This base class lets developers control various parameters that should be enforced for uploaded files, such as the types of files that a user can upload and the maximum size of a file. It also lets developers determine where and how the uploaded file should be stored. All derived classes used for upload strategies must inherit from the abstract **FileUploadStrategyBase** class.
- **Upload result class** – This base class lets developers access the details of a file that was uploaded by a user,

such as its name, content type, and upload status. It also lets developers open and delete the corresponding file. All derived classes used for specializing upload results must inherit from the abstract **FileUploadResultBase** class.

The framework provides a default upload strategy class that is named **FileUploadTemporaryStorageStrategy** and a default upload result class that is named **FileUploadTemporaryStorageResult**. This upload result class stores uploaded files to the temporary blob storage and provides a download URL. Developers can also implement their own custom upload strategy and upload result classes as required. For the upload strategy, two abstract methods from the **FileUploadStrategyBase** class must be implemented: **uploadFile** and **getResultClassName**. The **uploadFile** method handles where and how the file is stored. The **getResultClassName** method retrieves the upload result class that is used in this strategy. The **FileUploadResultBase** class has fields for the file name, the upload status, the content type of the file, and the log message. This class can be extended as required. All new properties should be able to be serialized and deserialized. The **openResult** method opens the file as a stream, and the **deleteResult** method deletes the file from the corresponding data storage.

Sequence diagram

The file upload control accepts the file and upload strategy in the client, and sends them to the file services. The file services start a new session, create an instance of a strategy class, and call the **uploadFile** method. When the **uploadFile** method has finished storing the file in the data source, a file upload result class returns to the file services. This class is sent back to the client, which might trigger the **OnUploadCompleted** event to deal with the post-process.



Scanning uploaded files for viruses and malicious code

Before you upload a file into the system, you might want to scan it for viruses or malicious code. Therefore, in version 10.0.12 and later, an extension point is available so that customers can integrate the file scanning software of their choice into the file upload process. Similar extension points are also available for scanning attachments. For more information about those extension points, see Configure document management.

IMPORTANT

Out of the box, Finance and Operations apps don't scan files for viruses and malicious code, and we don't recommend specific software for file scanning. Instead, customers are responsible for choosing their own file scanning software, and for adding the appropriate code to the delegate handlers so that they can use the software or service of their choice to scan files.

In particular, the **FileUploadResultBase** class exposes the **delegateScanStream()** delegate. This delegate applies to any file upload scenario where the **Upload strategy class** has been specialized. The upload process will fail if the scanning service determines that the file is malicious.

Implementation details

The following example of the **ScanDocuments** class shows boilerplate code for the handler. For general information about how to implement handlers for delegates, see EventHandlerResult classes in request or response scenarios.

```
    public final class ScanDocuments
    {

        [SubscribesTo(classStr(FileUploadResultBase), staticDelegateStr(FileUploadResultBase,
delegateScanStream))]
        public static void FileUploadResultBase_delegateScanStream(System.IO.Stream _stream,
EventHandlerRejectResult _validationResult)
        {
            if (!ScanDocuments::scanStream(_stream))
            {
                _validationResult.reject();
            }
        }

        private static boolean scanStream(System.IO.Stream _stream)
        {
            /*
            Custom implementation required for connecting to a scanning service
            If document scanning process found an issue, return false; otherwise, return true;
            */
            return true;
        }
    }
```

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

System-defined buttons

2/18/2021 • 10 minutes to read • Edit Online

This topic describes the system-defined buttons.

Overview

Several system-defined buttons are automatically present on the Action Pane. In general, these system-defined buttons should be applicable and should kept available to the end user. However, in rare cases (for example, if a more specialized control is required, or if a system-defined button isn't useful or applicable for a particular form), developers might have to explicitly suppress or override a system-defined button. For example, in some situations, a MenuButton that lets the user select from multiple "New" options might be preferable to the system-defined **New** button.

List of system-defined buttons

The following tables give the full list of system-defined buttons. The tables also provide information that will be useful if these buttons must be conditionally or completely suppressed or overridden.

Common buttons

BUTTON	BUTTON NAME MACRO*	COMMENTS
Export		Don't suppress this button.
Attach	#SystemDefinedAttachButton	Don't suppress this button, because we will suppress it on forms that aren't set up for attachments.
Show filters	#SystemDefinedShowFiltersButton	By default, **Visible**=**No** on TOC forms.

* System-defined button name macros are found in the SysSystemDefinedButtons macro file.

Buttons that are specific to Details forms

These buttons are an integral part of the Details form experience. Therefore, it's very unlikely that you'll have to suppress these buttons.

BUTTON	BUTTON NAME MACRO*	COMMENTS
Change view	#SystemDefinedShowMenuButton	This button exists under the **Options** Action Pane tab.
Grid view	#SystemDefinedGridViewButton	
Details view	#SystemDefinedDetailsViewButton	
Line details view	#SystemDefinedLineDetailsViewButton	
Header details view	#SystemDefinedHeaderDetailsViewButton	

BUTTON	BUTTON NAME MACRO*	COMMENTS
Show list	#SystemDefinedShowListButton	

* System-defined button name macros are found in the SysSystemDefinedButtons macro file.

Form styles that have no system-defined buttons

For several form styles, it doesn't make sense to add system-defined buttons, primarily because these forms don't have standard Action Panes. The following form styles never receive system-defined buttons:

- Dashboard
- Dialog
- DropDialog
- FormPart
- Lookup
- Sitemap
- Wizard

Suppressing most or all of the system-defined buttons

If you find that you're suppressing most or all of the system-defined buttons on a form, you should reexamine your form style (**Form.Design.Style**) or pattern, and reconsider the purpose of the form. Should the form be a dialog instead? (By default, dialogs don't receive any system-defined buttons.) Often, forms that are in this situation have **Style**=**Auto** and would be more appropriate as dialogs. If your form should not technically be a dialog, there is no currently no metadata or code that can automatically suppress all the system-defined buttons at the same time. Unless you switch the form to a form style that doesn't receive any system-defined buttons, you must to suppress/override each button individually (see the other sections in this article). This scenario should be extremely rare.

New and Delete system buttons

The **New** and **Delete** buttons are currently added by the kernel, and are controlled via special metadata properties. These buttons always work on the first master data source on the form.

When are these buttons available by default?

Forms usually have system-defined **New** and **Delete** buttons. These buttons appear on a form under the following conditions:

- The form has a style that allows for system-defined buttons.
- There is at least one data source on the form.

How do I affect the visibility of the New and Delete buttons on a form, but still allow the standard New and Delete tasks to fire via keyboard shortcuts?

Use the **ShowNewButton** and **ShowDeleteButton** properties on **Form.Design** to control the visibility of the **New** and **Delete** buttons on a form. Note that if the data sources still let the user create and delete records, the keyboard shortcuts will continue to work even if the system buttons aren't visible.

PROPERTY	VALUE	DESCRIPTION
Form.Design.ShowNewButton	No	Suppress the system-defined **New** button on the form.

PROPERTY	VALUE	DESCRIPTION
Form.Design.ShowDeleteButton	No	Suppress the system-defined **Delete** button on the form.

How do I affect the state (enabled/disabled) of the New and Delete buttons, and the associated task behavior on a form?

To affect both the state of the **New** and **Delete** buttons and the associated task behavior on a form, use the **AllowCreate** and **AllowDelete** properties on the first master data source. Additionally, the associated keyboard shortcuts will have no effect if you use this approach to disable the **New** and **Delete** buttons.

PROPERTY	VALUE	DESCRIPTION
Form.Datasources.<FirstMasterDatasource>.AllowCreate	No	The form doesn't allow record creation. The form has a disabled system-defined **New** button.
Form.Datasources.<FirstMasterDatasource>.AllowCreate	Yes	The form allows record creation. The form has an enabled system-defined **New** button (if it's visible).
Form.Datasources.<FirstMasterDatasource>.AllowDelete	No	The form doesn't allow record deletion. The form has a disabled system-defined **Delete** button.
Form.Datasources.<FirstMasterDatasource>.AllowDelete	Yes	The form allows record deletion. The form has an enabled system-defined **Delete** button (if it's visible).

> **NOTE**
> If the form has a New record action, that button control overrides the enabled state of the system-defined **New** button.

How do I change the behavior of the New task (either by clicking the button or by using the keyboard shortcut)?

There are three mechanisms for changing the behavior of the New task:

- Use the **New Record Action** property. This property is currently available only on **Form.Design**, and the referenced action will be triggered for **all** CommandButtons that call the **New** command on the form, even CommandButtons that are bound to secondary collections. In the future, the **New Record Action** property will be placed on all container controls to provide better control over the property's scope. Note that, currently, the **New Record Action** property also can't be defined as a Menu Button or Drop Dialog Button.

PROPERTY	VALUE	DESCRIPTION
Form.Design.NewRecordAction	The control name from the form	This property overrides the New task to perform the specified action.

- (Recommended) Use eventing. In particular, there are pre-events and post-events for record creation and deletion, where you can put code that is meant to run before and after these actions. See the following code example.

```
[Form]
public class Form1 extends FormRun
{
    public void init()
    {
        super();
        element.dataHelper().RecordCreating += eventhandler(this.PreRecordCreate);
        element.dataHelper().RecordCreated  += eventhandler(this.PostRecordCreate);
        // similar methods exist for deletion:
        // element.dataHelper().RecordDeleting, element.dataHelper().RecordDeleted
        //
        // explicit actions can be triggered via:
        // element.dataHelper().RecordCreate(), element.dataHelper().RecordDelete()
    }
    public void PreRecordCreate(FormRunServiceArgs _cancellableArgs)
    {
        //  I can add my logic here that gets invoked before the global creation task
        // if I need to, I can cancel the "super" of the task by doing:
        // _cancellableArgs.cancel()
    }
    public void PostRecordCreate()
    {
        //  I can add my logic here that gets invoked after the global creation task
    }
}
```

- Create an override on the **create()** or **delete()** method on the corresponding data source.

How do I change the behavior of the New button (but not the task)?

Imagine that you want to replace the system-defined **New** button with a menu button that lets the user select among several "New" options. Therefore, you complete the following tasks:

1. Hide the system-defined **New** button.
2. Model your own button for the New action.

However, at this point, the system-behavior for New still fires when the keyboard shortcut is invoked. If you want to make this modeled **New** button fire when the keyboard shortcut is pressed, you must set this button as the NewRecordAction on **Form.Design**. As we noted in the previous section, the action will currently be fired for all CommandButtons that call the New task on the form. Therefore, you should not use this approach on forms that have multiple data sources.

Edit, Done, Save, and Restore system buttons

The **Edit**, **Done**, **Save**, and **Restore** buttons let users switch the edit mode of the form as they require. Because this capability is critical, these buttons can't currently be suppressed. However, if a form should always be in **Edit** mode or should always be in **View** mode, you can use the **ViewEditMode** property to achieve that effect, and the buttons won't appear on the form. Note that most forms should be left as **ViewEditMode=Auto**. Don't set **ViewEditMode=View** or **ViewEditMode=Edit** for purely cosmetic reasons, but only if the form will always be read-only or will always be editable.

FORM.DESIGN.VIEWEDITMODE	FORM BEHAVIOR	SYSTEM-DEFINED BUTTON BEHAVIOR
View	The form is always in **View** mode.	None of these buttons (**Edit**, **Done**, **Save**, and **Restore**) are shown, because the framework assumes that they aren't needed.

FORM.DESIGN.VIEWEDITMODE	FORM BEHAVIOR	SYSTEM-DEFINED BUTTON BEHAVIOR
Edit	The form is always in **Edit** mode.	Only the **Save** and **Revert** buttons are shown, because the user can't exit **Edit** mode. The **Revert** button is on the framework-provided **Options** tab.
Auto	The form can be switched between **View** mode and **Edit** mode.	In **View** mode, the **Edit** button is shown on the Action Pane. In **Edit** mode, the **Save** button is shown on the Action Pane, and the **Read mode** and **Revert** buttons are shown on the framework-provided **Options** tab.

Can I conditionally suppress or show the Edit button?

If a form is editable at some points and read-only at other points, the logic that is used to determine whether the user can edit the form can be used to set the **ViewEditMode** property at run time. When the form should be read-only, set **ViewEditMode=View**. When the form can be either edited or viewed, set **ViewEditMode=Auto**.

How do I run additional code when the Edit button is clicked?

To have additional code run when the user clicks the **Edit** button, use eventing (see the example in the "How do I change the behavior of the New task (either by clicking the button or by using the keyboard shortcut)?" section earlier in this article). Specifically, use the following events:

- To subscribe to view/edit mode switching, use these events:
  - element.viewEditModeHelper().EditModeSwitching
  - element.viewEditModeHelper().EditModeSwitche
- To query the current view/edit mode, use these events:
  - element.viewEditModeHelper().isInEditMode()
  - element.viewEditModeHelper().IsInViewMode()
- To trigger view/edit mode switching, use these events:
  - element.viewEditModeHelper().setViewEditMode(ViewEditMode::Edit)

Refresh, Popout, and Close system buttons

The **Refresh**, **Popout**, and **Close** buttons are system buttons that are located on the right side of the Action Pane. The **Refresh** button is used to refresh all data on the form. The **Popout** button is used to move the current form into a separate window. The **Close** button closes the form (essentially, this button clicks the browser's **Back** button). These system buttons are integral components and can't currently be suppressed.

Other system-defined buttons

The remaining system-defined buttons are added during **Form.Init**. They are added only if a control of the same name doesn't already exist.

How do I run additional code together with a system-defined button or change the behavior of the button?

- For view switching buttons (for example, buttons that switch to the grid view, details view, header view, or lines view), you should override the **pageActivated()** method on the individual TabPage.
- For system-defined buttons that have pre-eventing/post-eventing (for example, **New**, **Delete**, and **Edit**), you can subscribe to the appropriate events. See the corresponding sections for **New**/**Delete** and **Edit** buttons for information about specific events for those buttons.

- For system-defined buttons that don't call a task directly (for example, **SystemDefinedShowMenuButton** and **SystemDefinedShowListButton**), you can register an override on the button through a **registerOverrideMethod()** call to have additional code run when the system-defined button is clicked.

```
public void overrideFunction(FormCommandButtonControl _command)
    {
        // Put any pre-super code here
        // This serves as the call to super()
        _command.clicked();
        // Put any post-super code here
    }
```

How do I suppress any of these system-defined buttons on a form, but without suppressing any corresponding task?

In general, we don't recommend that you suppress any of the system buttons. Their presence provides consistency to actions on forms. However, the buttons currently appear in some situations where they aren't as useful. We have future work items to address many of these situations. For example, have work items for the following tasks:

- Suppress the **Attach** button on forms that aren't configured to allow attachments.
- Suppress the **Export** button on forms that have no grids.
- Suppress the **Show filters** button on forms that don't have a main grid.

However, if you must suppress one of these buttons (strongly discouraged), you can find the control via code and set its visibility to **false**, as shown in the following code example. Use SysSystemDefinedButtons macros, where they are available, to reference the button names.

```
FormCommandButtonControl attachButton;

    public void init()
    {
        #SysSystemDefinedButtons

        super();

        attachButton= this.control(this.controlId(#SystemDefinedAttachButton)) as FormCommandButtonControl;
        attachButton.visible(false);
    }
```

Images on a page or in a grid

2/18/2021 • 13 minutes to read • Edit Online

This topic describes the steps for displaying images on a page or in a grid. The topic also provides background about some of the ways that images can be used, and the APIs that are used.

> **NOTE**
> For accessibility, when you use an image to indicate status or show data, the image must be accompanied by a tooltip, enhanced preview, label, or other textual representation that describes the value or status that the image represents.

Finance and Operation apps do not use embedded resources for images. Instead, it uses lightweight symbols. The coding pattern has changed slightly to support the new image control.

For ImageList uses, the runtime accepts the old **ImageID** value and maps it to a symbol, so that existing code continues to work.

> **NOTE**
> In some cases, there is no image even after runtime mapping, and this behavior is intentional.

AX 2012 displays images in a grid column to indicate status. These images were sometimes retrieved from embedded resources that are no longer available.

AX 2012 offers the following storage options for images:

- An embedded resource where images are offered as part of the kernel itself
- An Application Object Server (AOS) resource where developers or independent software vendors (ISVs) can add their own image resources
- A file location where developers or ISVs can load images at run time
- A database field that is stored as a bitmap

The following storage options are available for images:

- An AOS resource where developers or ISVs can add their own image resources
- A URL location where developers or ISVs can load images at run time
- A database field that is stored as a container.
- A symbol font, where images are rendered by name from the font

Images that are stored as AOS resources allow for the use of an image that isn't categorized as user data, and can be used with your application.

> **NOTE**
> If there are legacy embedded resource images that UX has approved for use, those embedded images can be manually transferred to an AOS resource and used.

A typical web application maintains a collection of images on an Internet Information Services (IIS) server and just provides a URL to the image. Although this approach is supported, we don't expect that it will be used very much. Instead, we expect that the symbol font will be used as an image source.

Of course, application logic will store an image in a database to allow for strong employee photos, product images, and so on, and this approach is a first-class experience.

A symbol font is the most performant and scalable image format. We expect that characters from the symbol font will be used for most application use cases (grid row by row status, button images, and so on).

For the list of symbols that are available in the symbol font, see Symbol font.

Image type: Symbol

PROS	CONS
• Usually, the symbol font is the smallest payload to send to the client. • You can easily customize the images by using Cascading Style Sheets (CSS). • The symbol font should already be cached on the user's computer. Therefore, no extra bandwidth is used, and there are no additional network requests that might slow down page loads. • Colors can be controlled by themes. • The images are automatically scaled on high-DPI displays.	A limited number of framework-defined symbols is available.

Design time

Image location: Symbol **Typical image:** "Person"

Run time

Sometimes, you don't have an image for a particular record in a grid, but you don't want an empty space where the image should be. The following example shows how you can use a display method to check for an image value, and then substitute a placeholder image instead.

```
public display container customerImage()
{
    ImageReference imgRef;
    container imgContainer = this.Image;
    if(imgContainer == connull())
    {
        // there is no image… the container is null
        // show a generic person outline image
        imgRef = ImageReference::constructForSymbol("Person");
        imgContainer = imgRef.pack();
    }
    return imgContainer;
}
```

Image type: AOT Resource

PROS	CONS
In addition to the pros of using URL images (see the next section), AOT resources are modeled and managed by the development tools.	A limited number of framework-defined images is available.

Design time

| You just create a new resource and then save the image into the Application Object Tree (AOT) resource. When

you model your image control on a page, you specify the resource name, not the image name. This approach is typically used for legacy images (icons) that don't have equivalents in the symbol font. **Image location:** AOTResource **Typical image:** "ResourceMicrosoft Dynamics AX" (a .jpg is added to resources) |

Run time

```
public display container imageDataMethod()
{
    ImageReference imgClass =
            ImageReference::constructForAotResource(
              "ResourceMicrosoft Dynamics AX");
    return imgClass.pack();
}
```

Image type: URL Image

PROS	CONS
• This approach provides an easy way to reference any image anywhere on web. • This approach supports full-color images. • The web browser can cache the image, based on the settings of the server that hosts the image.	• The transfer size isn't as small as it is for symbols, but it's reasonable. The URL is sent as a string for each control that uses the image. The browser then downloads the image from the URL, and from that point, standard browser caching rules apply. • You can't easily theme the images by using CSS. • Unless the URL points to a Scalable Vector Graphics (SVG) file, the image isn't automatically scaled on high-DPI displays.

Run time

The following example shows an image that uses a URL that is contained in a string.

```
public display container imageDataMethod()
{
ImageReference imgClass = ImageReference::constructForUrl(this.ImageURL);
return imgClass.pack();
}
```

This code sends a small JavaScript Object Notation (JSON) message to the control on the client. This message instructs the control to treat the image as a URL and let the browser do the work of downloading the image. No download occurs on the server. **Storing an image URL in a database table** You can also have a container field for the image column on your table. You can then use code that resembles the following example to store the **ImageReference** pack.

```
ImageReference imgClass;
CLIControls_ImageTable imgTable;
ttsbegin;
imgClass = ImageReference::constructForUrl(
    "http://dynamics/PublishingImages/ERPLogos/DynamicsLogo.jpg");
imgTable.ImageField = imgClass.pack();
imgTable.insert();
ttscommit;
```

This code causes the user's browser to download the image from the specified URL. The use of ImageReference involves some overhead, but this approach lets you use a single application programming interface (API) to handle images that are created from binary data, URLs, AOT resources, or symbols. You can even mix and match image types between rows of data.

Image type: Binary Image

PROS	CONS
Usually, this approach offers the easiest migration if the **Image** class in X++ was already used, or if binary images were previously stored in the database.	• This approach involves the largest transfer size, because the binary image is encoded as a string and sent to the client as part of the interaction. • The browser can't cache the images. • For a grid, the binary-encoded image is sent for every row, even if multiple rows use the same image. Therefore, this approach can lead to very large transfer sizes in the interactions. • You can't easily theme the images by using CSS. • The images aren't automatically scaled on high-DPI displays.

DESIGN TIME	RUN TIME
Using a database field This approach is typically used to display data, such as employee pictures and product images. You can bind directly to a field, or you can use a display method. Data Source Data Field Data Method	Typically, the images are loaded from database, and no additional code is required. For cases where the image is managed in a data method, see the data method examples.

Display methods and images (three return types)

When you use a display method for an image type to show an image in a grid, three return types are understood by the image control that works with the framework. All three return types can be used to display an image.

- Int (imagelist array index)
- Container (image instance)
- ResID (which is mapped to a symbol)

> **NOTE**
>
> ResID and Int are the same return types. If the **imageList** property of the image control instance has been assigned an instance value, the display method return value is considered an array index into the imagelist. If the **imageList** property is **null**, the return value is used to map a legacy ResID to a symbol.

Images in a grid and the legacy ImageList collection

In AX 2012 and earlier versions, a common use pattern for displaying images is to store an image as a resource or use a kernel-supplied image resource, and then at run time, extract that image and place it in a reusable collection that is known as an ImageList. The guidance is to use lighter-weight symbol images. You should rewrite all legacy code so that it uses symbols directly. You should also replace all code that uses the ImageList collection. If you don't make these changes, the legacy ImageList collection won't display images, because use of this collection relies on embedded (kernel) resources that no longer exist. Therefore, to support legacy code until it can be updated, the ImageList collection maps the ResID for an embedded resource to a new font-based symbol to help guarantee that any code that uses the ImageList collection will continue to run and provide an image.

Using the imageList property for backward compatibility

An image control has a property that is named **imageList**. You pass in an instance of the ImageList collection to this property. In this way, the image is an array of images that you select via the array number.

```
public void init()
{
    int imgCnt;

    // create an imagelist instance
    Imagelist imageList = new ImageList(ImageList::smallIconWidth(), Imagelist::smallIconHeight());

    super();

    // add images to the instance (return value is not needed)
    // Note that a legacy ResID is used in the new Image contstructor.
    // This is a compatibility mapping of resource to symbol.
    imgCnt = imagelist.add(new Image(#ImageInfo));
    imgCnt = imagelist.add(new Image(#ImageWarning));
    imgCnt = imagelist.add(new Image(#ImageError));

    // pass the image list instance to the control
    ImageListDM.imageList(imageList);
}

// at runtime, select the image you want to show: when the control has an imagelist instance,
// this int value is used to index into that array
public display int imageListDataMethod()
{
    int imgCnt = imageCnt mod 3;
    imageCnt++;
    return imgCnt;
}

/*
    Note: The legacy image resource ID's #ImageInfo, #ImageWarning, #ImageError are
    mapped from the legacy resource id to a symbol name in the X++
    class ImageLoader
*/
```

Display method that returns an ImageRes (legacy image resource ID)

```
// this is an example of backward compatibility the use of ImageRes will become obsolete
display ImageRes checkIfError(HRMCompEventEmpl _hrmCompEventEmpl)
{
    if (!_hrmCompEventEmpl.RecId)
    {
        return 0;
    }
    if (_hrmCompEventEmpl.Status == HRMCompEventEmplStatus::Ignore   ||
        _hrmCompEventEmpl.Status == HRMCompEventEmplStatus::Approved ||
        _hrmCompEventEmpl.Status == HRMCompEventEmplStatus::Loaded)
    {
        return 0;
    }
    else
    {
        if (_hrmCompEventEmpl.ErrorStatus == HRMCompEventErrorStatus::Error)
        {
            return #ImageError;
        }
        if (_hrmCompEventEmpl.ErrorStatus == HRMCompEventErrorStatus::Warning)
        {
            return #ImageWarning;
        }
        if (_hrmCompEventEmpl.ErrorStatus == HRMCompEventErrorStatus::Info)
        {
            return #ImageInfo;
        }
    }
    return 0;
}
```

Display method that returns a container

```
public display container checkIfError(HRMCompEventEmpl _hrmCompEventEmpl)
{
    ImageReference  imageReference;
    container       imageContainer;
    if (_hrmCompEventEmpl.RecId && _hrmCompEventEmpl.Status == HRMCompEventEmplStatus::Created)
    {
        switch (_hrmCompEventEmpl.ErrorStatus)
        {
            case HRMCompEventErrorStatus::Error:
                imageReference = ImageReference::constructForSymbol('Error');
                break;
            case HRMCompEventErrorStatus::Warning:
                imageReference = ImageReference::constructForSymbol('Warning');
                break;
            case HRMCompEventErrorStatus::Info:
                imageReference = ImageReference::constructForSymbol('Info');
                break;
        }
    }
    if (imageReference)
    {
        imageContainer = imageReference.pack();
    }
    return imageContainer;
}
```

Obtaining and displaying an image from the user by using file upload

Model a page that has an image control and a **FileUpload** button.

```
// model a new FileUpload control (style=minimal)
// class declaration
FileUpload uploadControl;

// form init() create a callback event handler to be notified when upload is complete
public void init()
{
    //when uploading an image, this method is called upon completion.
    uploadControl = FileUpload1;
    uploadControl.notifyUploadCompleted +=  eventhandler(this.UploadCompleted);
}

// form close() release the callback event handler
public void close()
{
    // when the form closes, release the eventhandler for file upload callback
    //  FileUpload uploadControl;
    super();
    //  uploadControl = FileUpload1;
    uploadControl.notifyUploadCompleted -=  eventhandler(this.UploadCompleted);
}

// when the upload completes, grab the image and store it in the database
/// <summary>
/// This method is called by the file upload mechanism, when the upload completes
/// </summary>
public void UploadCompleted()
{
    Binary binaryImage;
    System.Net.WebClient webClient;
    System.IO.MemoryStream stream;
    String255 myUrl;
    if(uploadControl.uploadSuccess())
    {
        InteropPermission perm = new InteropPermission(InteropKind::ClrInterop);
        perm.assert();

        // BP Deviation Documented
        webClient = new System.Net.WebClient();

        // BP Deviation Documented
        // if success, downloadURL contains the path to the Azure blob location for the file
        stream = new System.IO.MemoryStream(webClient.DownloadData(uploadControl.downloadUrl()));

        // grab the data and assign to the image field
        binaryImage = Binary::constructFromMemoryStream(stream);

        // assign to the database field (type=container)
        FMVehicleModel.Image = binaryImage.getContainer();

        CodeAccessPermission::revertAssert();
    }
}
```

Example of in-memory bitmap manipulation

In this example, an image is created from scratch. However, developers can also load a bitmap from an alternative source and then manipulate the image as desired (for example, by cropping, stretching, or resizing, or by changing the opacity). After any manipulation is completed, the developers can display the image by using the image control, or they can assign it to a data source field.

```
public void clicked()
{
    Binary binaryImage;
    Image  image;
    int x,y;

    super();

    InteropPermission perm = new InteropPermission(InteropKind::ClrInterop);
    perm.assert();

    /*
    In this example, we'll create a bitmap programmatically, we'll use a memory
    Stream o'bytes to then convert to the container format the image control expects.
    */
    System.Drawing.Bitmap bitmap = new System.Drawing.Bitmap(100,100);
    System.IO.MemoryStream myStream = new System.IO.MemoryStream();

    // draw some stuff (or load a bitmap from an alternative source)
    for( x=0; x < bitmap.Height; ++x)
    {
        for( y=0; y< bitmap.Width; ++y)
        {
            bitmap.SetPixel(x,y,System.Drawing.Color::White);
        }
    }

    for(x=0; x < bitmap.Height; ++x)
    {
        bitmap.SetPixel(x,x, System.Drawing.Color::Red);
    }

    // move our bitmap to an in memory stream
    bitmap.Save(myStream, System.Drawing.Imaging.ImageFormat::Bmp);

    // stream goes to raw binary
    binaryImage = Binary::constructFromMemoryStream(myStream);

    // create a blank image and copy our binary data to the image format
    image = new Image();
    image.setData(binaryImage.getContainer());

    // copy the image data to the image control
    MyImage.image(image);

    // alternatively, skip the image conversion step and assign directly to the data field
    binaryImage = Binary::constructFromMemoryStream(myStream);

    // assign to the database field (type=container)
    datafield.Image = binaryImage.getContainer();

    CodeAccessPermission::revertAssert();
}
```

Additional examples (URL, binary, and symbol)

The following table explains two concepts: Image Class and FormImageControl.

Image Class	This class is a run-time representation of an image.	Four constructors: • Image::ConstructBinary(INT64Encode); • Image::ConstructSymbol(SymbolName); • Image::ConstructURL(URL); • Image::Construct(ResourceName);
FormImageControl	This control is used to add an image at run time.	.image(new image());

Using a display method to show an image from a URL string

In this example, a display method is used to translate a string that contains a URL to the format that the image control expects.

```
public display container imageDataMethod()
{
    ImageReference imgClass = ImageReference::constructForUrl(this.ImageURL);
    return imgClass.pack();
}
```

This code sends a small JSON message to the control on the client. This message instructs the control to treat the image as a URL and let the browser do the work of downloading the image. No download occurs on the server.

Using a display method to show a blank image

There might be times when you have no image for a particular record in a grid, but you don't want an empty space where the image should be. This example shows how you can use a display method to check for an image value and then substitute a placeholder image instead.

```
public display container customerImage()
{
    ImageReference imgRef;
    container imgContainer = this.Image;
    if(imgContainer == connull())  // there is no image… the container is null
    {
        imgRef = ImageReference::constructForSymbol("Person");  // show a generic person outline image
        imgContainer = imgRef.pack();
    }
    return imgContainer;
}
public display container statusImageDataMethod()
{
    ImageReference statusImage;
    if (this.Status == NoYes::Yes)
    {
        statusImage = ImageReference::constructForSymbol("Accept");
    }
    else
    {
        statusImage = ImageReference::constructForSymbol("Cancel");
    }
    return statusImage.pack();
}
```

Taking an image URL and storing the image in table

You can have a container field for the image column on your table. You can then use code that resembles the following example to store the **ImageReference** pack.

```
ImageReference imgClass;
CLIControls_ImageTable imgTable;
ttsbegin;
imgClass = ImageReference::constructForUrl(
    "http://dynamics/PublishingImages/ERPLogos/DynamicsLogo.jpg");
imgTable.ImageField = imgClass.pack();
imgTable.insert();
ttscommit;
```

Like the display method that is described in the "Using a display method to show an image from a URL string" section, this code causes the user's browser to download the image from the specified URL. Although this approach involves some overhead, you can use a single API to handle images that are created from binary data, URLs, AOT resources, or symbols. You can even mix and match image types between rows of data.

Font and background colors for input, table, and grid controls

2/18/2021 • 3 minutes to read • Edit Online

This topic provides information about the new color picker control that lets users select a color.

Traditionally, color has been considered an ideal way to communicate with a user. For example, the color red is often used to draw the user's attention to information that is important. However, some users can't distinguish certain colors or shades, and some users are blind. Therefore, we don't recommend that you use color alone to communicate information to the user. Instead, you should use color together with a symbol or additional text to convey information to all users.

Color selection in Dynamics AX 2012

In Microsoft Dynamics AX 2012, color selection had these characteristics:

- It used the Win32 color picker.
- It required Win32 application programming interfaces (APIs) for RGB/decimal conversion. (The input control accepted a decimal value for RGB.)

```
Public void lookup()
{
    #DEFINE.COLORVALUE(64)
    Int r,g,b
    container choosencolor;
    Binary customcolors = new Binary(#COLORVALUE);
    CCColor colorvalue;

    Super();

    [r,g,b] = WinAPI::RGBint2Con(this.backgroundColor());

    chosenColor = WinAPI::chooseColor(element.hWnd(),r,g,b, customColors, true);

    If(chosencolor)
    {
        [r, g, b] = chosencolor;
        Colorvalue = WinAPI::RGB2int(r,g,b);
        This.backgroundColor(colorValue);
        employeeWorkPlannerForm.parmAbsensceColor(colorvalue);
        Employeetable.columns(employeeworkplannerform.numberofcolumns());
        Absenscecolorparm = colorvalue;
    }
}
```

Color selection for input controls

In the current version, the color picker control is a standard control type. The color picker control can be put directly in a form, or it can be used as part of a custom lookup for an integer or string control. The following example shows how to interact with the color picker control in a custom lookup. However, the code is similar if you put the color picker in a form and provide the user with a button to select a color.

- A color picker control can be hosted in a form or a custom lookup to let the user visually pick a color or specify an RGB value.

- The return value is a decimal value that can be assigned directly to an input control property. (No run-time RGB conversion is required.)

```
[Control("String")]
class stringControl
{
    /// <summary>
    ///
    /// </summary>

    public void lookup()
    {
        int color = hex2Int(this.valueStr());
        color = ColorSelection::selectColorStringControl(this, color);
        this.text(int2hex(color));
        this.backgroundColor(color);
    }
}

[Control("Integer")]
class integerControl
{
    /// <summary>
    ///
    /// </summary>
    public void lookup()
    {
        int color = this.value();
        color = ColorSelection::selectColor(this, color);
        this.value(color);
        this.backgroundColor(color);
    }
}
```

Using color in a table control

There is no design-time experience for coloring input controls. In other words, you can't model an input control so that it's "blue" by default. However, there are run-time capabilities that let you change color values. The following example shows how you can change the way that the cells of a table control are colored.

```
public FormControl editControl(int column, int row)
{
    stringEdit.colorScheme(FormColorScheme::RGB);
    stringEdit.backgroundColor(WinAPI::RGB2int(225,225,125));
    stringEdit.foregroundColor(WinAPI::RGB2int(8,10,200));
}
```

	Column1	Column2	Column3	Column4	Column5	Column6
ROW1	Text 1	Text 2	Text 3	Text 4	Text 5	
ROW2	Text 2	4	Text 6	8	Text 10	✓
ROW3	Text 3	6	Text 9	12	Text 15	
ROW4	Text 4	8	Text 12	16	Text 20	✓
ROW5	Text 5	10	Text 15	20	Text 25	

Using color in a grid control

```
public void displayOption(Common _record, FormRowDisplayOption _options)
{
    CLIParentTable table;
    table = _record;

    if(!cleared)
    {
        _options.affectedElementsByControl(CliParentTable_AInt.id());
        _options.affectedElementsByControl(CliParentTable_AEnum.id());
        _options.affectedElementsByControl(CliParentTable_AString.id());
        _options.affectedElementsByControl(CliParentTable_EditMethodString.id());

        if(table.AInt<=20)
        {
            _options.backColor(WinAPI::RGB2int(255,165,0));
        }
        else if( table.AInt>20 &&  table.AInt <60)
        {
            _options.backColor(WinAPI::RGB2int(255,255,0));
            _options.fontItalic(true);
            _options.textColor(WinAPI::RGB2int(255,0,127 ));
            _options.fontStrikethrough(true);
        }
        else
        {
            _options.backColor(WinAPI::RGB2int(128,0,128));
                _options.fontUnderline(true);
        }
    }

    super(_record, _options);
}
```

Static RGB instead of run-time conversion from integer to RGB values

Previously, run-time conversion that used **WinAPI::RGB2Int** was required, because the Win32 color picker returned an RGB value, whereas the background color APIs accepted an integer. This run-time conversion isn't required, because the new color picker returns an integer to match the control's consumption of an integer. Additionally, it's understood that .NET code often uses RGB values for colors. Therefore, in those cases, run-time conversion of colors isn't required for each use. Instead, you can define static color variables. Here are three examples.

```
Static int GrayColor = 220 + 220 <<#offset8 + 220<<offset16;

Static int GrayColor = 0xdcdcdc

Static int GrayColor = 14474460; // DCDCDC or 220,220,220
```

Right-to-left language support and bidirectional text

2/18/2021 • 12 minutes to read • Edit Online

In the area of right-to-left (RTL) language support, one consideration is the combination of RTL text and left-to-right (LTR) text in the same string. This topic discusses the issue of bidirectional text and how it's handled.

A great example of right-to-left language support: Microsoft Word

In the area of right-to-left (RTL) language support, one consideration is the combination of RTL text and left-to-right (LTR) text in the same string. One example of a program that implements this functionality correctly is Microsoft Word. If you're trying to understand the correct behavior of mixed language presentation, you can use Word for validation. The problem is that most software just implements the Unicode standard to display bidirectional data, without evaluating how that data is actually used. Additionally, there's no attempt to provide the interactive experience that the user actually requires.

To understand how Word "gets it right" and provides a great experience, you can inspect the XML of a Word document. There, you will see that Word tracks (and stores together with the run of characters) the keyboard that is used to enter each character, and that it treats each character as a member of the language that is associated with the keyboard. Therefore, the character is given the behavioral aspects of that language.

Keeping track of character orientation in a financial program that might record billions of transactions and multi-billions of characters would produce significant transnational and spatial overhead if we stored contextual information for each character. Therefore, this behavior would be considered only for special conditions.

Bidirectional text

To support Arabic and Hebrew, both of which are RTL languages, there is an RTL orientation for the controls in each form, so that an RTL reader can interact with the form in a natural reading manner. For the most part, RTL orientation of the controls works as expected and provides RTL users with the experience that they expect. Finance and Operations apps and modern browsers support RTL orientation, and Finance and Operations app conform to that functionality. However, in some cases, extensible controls (custom controls) require special code to orient their elements correctly.

A point of reference in this article is the Win32 CEdit control, which is used primarily for standard text entry (account name, description, user name, and so on, in Microsoft Dynamics AX 2012). The behavior of the HTML Input control mimics the functionality of the CEdit control. Therefore, the same behavior applies to Finance and Operations.

The CEdit control is a Win32 control that is governed by the rules for bidirectional text management that are defined by the Unicode standard. Bidirectional text occurs when the control hosts both RTL text (such as Arabic or Hebrew) and LTR text within the same string of characters.

When you evaluate the examples in this article, remember that, regardless of the orientation of the form (RTL or LTR), the actual text that is presented is never reversed or "mirrored." English text is always read LTR and Arabic/Hebrew text is always read RTL. When LTR and RTL text are combined, the reader must jump to the beginning of the run of characters in a given orientation. For example, when a mixed string is read from right to left, the individual words might be read like this:

——————> <—————— ——————> <——————

English and Arabic/Hebrew text together: Bidirectional issues

The visual presentation (glyphs) of English, Arabic, and Hebrew characters on the corresponding keyboards clearly differ. However, those three keyboards also share some symbols. These symbols include numerals, and formatting characters such as parentheses, brackets, and underscores. According to the Unicode bidirectional algorithm, when these characters are used in a bidirectional string, their RTL/LTR orientation depends on **the context of the characters that surround them**. From the Unicode standard:

> **NOTE**
>
> You can find the Unicode display algorithm at https://www.unicode.org/reports/tr9/. (Section 3.3.4 of the algorithm describes how to position neutrals.)

- Characters that have a weak bidirectional type determine their directionality according to their proximity to other characters that have strong directionality.
- Characters that have a neutral bidirectional type determine their directionality from either the surrounding strong text or the embedding level.

The following table describes the bidirectional character types.

CATEGORY	TYPE	DESCRIPTION	GENERAL SCOPE
Strong	L	Left-to-Right	LRM, most alphabetic, syllabic, Han ideographs, non-European or non-Arabic digits, …
Strong	LRE	Left-to-Right Embedding	LRE
Strong	LRO	Left-to-Right Override	LRO
Strong	R	Right-to-Left	RLM, Hebrew alphabet, and related punctuation
Strong	AL	Right-to-Left Arabic	Arabic, Thaana, and Syriac alphabets, most punctuation specific to those scripts, …
Strong	RLE	Right-to-Left Embedding	RLE
Strong	RLO	Right-to-Left Override	RLO
Weak	PDF	Pop Directional Format	PDF
Weak	EN	European Number	European digits, Eastern Arabic-Indic digits, …
Weak	ES	European Number Separator	Plus sign, minus sign

CATEGORY	TYPE	DESCRIPTION	GENERAL SCOPE
Weak	ET	European Number Terminator	Degree sign, currency symbols, …
Weak	AN	Arabic Number	Arabic-Indic digits, Arabic decimal and thousands separators, …
Weak	CS	Common Number Separator	Colon, comma, full stop (period), Non-breaking space, …
Weak	NSM	Nonspacing Mark	Characters marked Mn (Nonspacing_Mark) and Me (Enclosing_Mark) in the Unicode Character Database
Weak	BN	Boundary Neutral	Default ignorables, non-characters, and control characters, other than those that are explicitly given other types
Neutral	B	Paragraph Separator	Paragraph separator, appropriate Newline Functions, higher-level protocol paragraph determination
Neutral	S	Segment Separator	Tab
Neutral	WS	Whitespace	Space, figure space, line separator, form feed, General Punctuation spaces, …
Neutral	ON	Other Neutrals	All other characters, including OBJECT REPLACEMENT CHARACTER

The fundamental problem with the Unicode standard for bidirectional text is that it fails to capture the user's intent. Therefore, the algorithm will move characters around within the same string, and put them in a location that the user didn't specify or doesn't want. This issue is troublesome for accounting and financial systems, because the data that users enter into the system might not match the corresponding source documents.

As we mentioned, Arabic, English, and Hebrew keyboards share some of the same characters. However, in some cases, those characters are positioned differently, depending on the keyboard that was used to type them, and/or the context of the surrounding characters and the orientation of the input control. These language-neutral characters include commas, periods, parentheses, hyphens, and underscore characters.

In some cases, the rules for displaying the same characters varies between languages. Additionally, those rules can change, depending on the kind of data that is displayed. For more information about this issue, see the "Issue: The hyphen used together with numbers: Language-specific behavior" section, later in this article.

Some people expect that characters that are entered in RTL mode will appear the same when the form is viewed

in LTR mode. In other words, the expectation is that a customer can have some users who use Hebrew and others who use English on the same installation or in the same company.

Issue: The underscore character used together with numbers

Description: 123_456 appears as 456_123, although the user wants it to appear as 123_456.

Example: The user wants to enter an item number (such as 123_456) or a journal name (such as BA_Chk_Rev) that includes underscore characters for grouping purposes.

There is a difference between the Unicode standard and what our users want to see in a financial program. Even Word presents 123_456 as 456_123 for both Arabic and Hebrew. This behavior occurs because the underscore character is a grouping mechanism. It splits the number into groups of numbers that can be read from right to left.

Numbers are read from left to right in Arabic and Hebrew. "Item" numbers, regardless of the combination (RTL_LTR_RTL, LTR_RTL_LTR, Neutral_RTL, and so), should appear exactly the same in a paragraph of any direction or alignment. This issue isn't easy to resolve for plain text programs. All the customer knows is that the physical item number is (from left to right) 123_456, and that the string should appear as 123_456 in every language, so that the number that users see always matches what they know the physical number is.

Control behavior: None of the off-the-shelf controls provide the desired behavior. Word fails too.

WPF RICHTXT	WIN32 CEDIT	WIN32 RICHTXT	WORD
No	No	No	No

Workarounds:

- When you're using an English keyboard, use a different delimiter. For example, enter **123.456** or **123/456**.
- When you're using a Hebrew keyboard, use a different delimiter. For example, enter **123.456**. The slash character (/) on a Hebrew keyboard produces a period (.).

Recommendation: None of the off-the-shelf controls provide the requested behavior. One alternative is to identify fields that must allow for this directional formatting and flag those fields as LTR for data input (right-aligned for display purposes). The program can't automatically determine that a field must have this behavior. Therefore, if we expose an RTL/LTR flag, a customizer can modify targeted fields for the desired behavior. Although this approach enables this specific scenario, it's important to understand that, if you extend the scenario by using characters in a combination of RTL and LTR languages, you will introduce other issues. Another alternative is to educate users about the fundamental behavior when underscore characters and numbers are used together. When underscore characters and numbers are required, users can then use a workaround to obtain the desired display behavior.

> **NOTE**
> The underscore character doesn't present an issue when you combine English and RTL languages (pattern: RTL_LTR_RTL). Therefore, if you force a control to LTR when the input includes numbers/text/underscore characters, the behavior won't be as expected. Users will have to manually reposition the cursor after each use of an underscore when they type RTL text. However, the behavior will be as expected for the use of numbers/text/underscore characters.

Hebrew: קשגכ_english_יקנקרק

Arabic: شقلاهؤ_english_شقشلاهؤ

Issue: The hyphen used together with numbers: Language-specific

behavior

Description: LTR is expected for Hebrew, whereas RTL is expected for Arabic.

Example: Item names that include numbers Arabic and Hebrew treat the hyphen differently when it's used together with numbers. An Arabic keyboard treats the hyphen as an RTL character, whereas a Hebrew keyboard treats it as an LTR character. Therefore, similar typed strings should be presented differently, depending upon the keyboard that was used. Some readers will be familiar with this example from a meeting of interested parties: 1-2-3-a-b-c

Arabic: The desired behavior is correct in Dynamics AX 2012. 1-2-3-ؤ-لال-ش

Hebrew: The desired behavior is incorrect in Dynamics AX 2012. 1-2-3-ב-נ-ש

The Unicode standard doesn't provide for language-specific or keyboard-specific behavior. Instead, it supplies fundamental bidirectional behavior and treats the hyphen as an RTL character. Therefore, it presents the Arabic string correctly but the Hebrew string incorrectly.

Control behavior: The WPF RichTxt control produces the correct/desired behavior for each language.

WPF RICHTXT	WIN32 CEDIT	WIN32 RICHTXT	WORD
Yes	No	No	Yes

Workarounds for the Hebrew user:

- Don't type hyphens between the numbers. For example, if you type 1 2 3-A-B-C on a Hebrew keyboard, it appears in RTL as C-B-A-3 2 1. You should assume that the ABC order is correct for Hebrew, which is an RTL language. The English ABC text is reversed here for demonstrative purposes.
- Use a different delimiter between the numbers. For example, the slash character (/) on a Hebrew keyboard produces a period (.).

Recommendation: This pattern is an issue for Hebrew users who want to use numbers or hyphens in item names. Therefore, a global solution might not be appropriate, because there are exceptions. Phone numbers, Social Security numbers, and other source document identification numbers are always read LTR.

The WPF RichTxt control provides the desired behavior according to the strict guidance. However, it isn't clear that this behavior is always the desired behavior. That is, phone numbers, US Social Security numbers, and so on, should always be read and appear in LTR order, regardless of the language orientation. The alternative is to identify fields that must enable this behavior. The program can't automatically determine that a field must have this behavior. Therefore, you might have to use a descriptive property on the control, so that users can specify "Structured Formatting." If none of these approaches can be achieved, you must educate Hebrew users about the fundamental behavior when hyphens are used together with numbers. Users can then use one of the preceding workarounds to get the desired display behavior, by omitting the hyphens between numbers.

Hebrew example: (Desired and correct Operations for both Arabic and the Hebrew example in Hebrew)

Pattern: First (עברית) hyphen second (English) hyphen third (שלום) hyphen forth (Hello)

Correct: עברית-English-שלום-Hello

Pattern:

1. First (English letter) hyphen second (Hebrew letter) hyphen
2. First (Hebrew letter) hyphen second (English letter) hyphen

A RTL form appears as desired:

1. ש--a

2. -aש-

Exception for phone numbers: Often, Arabic users don't have to use hyphens in phone numbers, because international phone numbers rarely use hyphens to separate digits. Any fundamental changes in the behavior of hyphens (for example, if you introduce use of the WPF RichTxt control) will cause phone numbers to appear incorrectly for Arabic users.

Phone numbers are always read LTR and often include hyphens. Phone numbers sometimes appear correctly, when they are shown in a grid through a display method that presents the string as LTR.

Currently, the input of phone numbers by using numbers and hyphens produces the correct display, such as 701-225-2188.

There is an issue with phone numbers if you try to use a US pattern that includes parentheses.

Arabic/Hebrew/English (desired): (701)225-2188

Arabic (actual):)701(225-2188

Hebrew (actual):)701(225-2188

Recommendation: Expose an RTL flag for controls or an extended data type for phone numbers. A customizer can force the control into LTR mode. This approach will let users enter values in the order that they want.

Issue: LTR text combined with neutral characters in RTL input

Description: English text is combined with parentheses or other neutral characters.

Example: A company name together with the company abbreviation, such as "Dynamics (DAT)"

In a typical example, the company name is followed by the company abbreviation, which is enclosed in parentheses. In this case, "Dynamics (DAT)" is shown as "(Dynamics(DAT". This behavior occurs because the closing parenthesis isn't surrounded by two English characters. Therefore, the parenthesis is treated as an RTL character. It's changed to the RTL closing parenthesis and moved to the end of the string (in RTL orientation).

Control behavior: None of the controls provide the desired behavior.

WPF RICHTXT	WIN32 CEDIT	WIN32 RICHTXT	WORD
No	No	No	Yes

The WPF RichTxt control has a flag that tries to format text according the first character in the string. Although the algorithm **should** fix this issue, it doesn't.

Workarounds: Don't use weak or neutral characters for grouping when you use English. For example, use "Dynamics DAT".

Recommendation: None of the controls provide the desired behavior. You must educate users about the fundamental behavior when weak or neutral characters are used together with English text. Don't use weak or neutral characters unless English characters appear on each side.

Create icons for workspace tiles

2/18/2021 • 2 minutes to read • Edit Online

This topic provides guidelines and recommendations for creating and assigning icons to custom workspace tiles.

The dashboard contains a set of workspace tiles to which the user has access. Each of these tiles contains an icon specific to that workspace. For out-of-the-box workspaces provided by Microsoft, the icons used on the workspace tiles generally correspond to a symbol from the Dynamics Symbol font. This topic discusses the guidelines and recommendations for creating and assigning icons to tiles for workspaces created by Microsoft Certified Partners or individual customers.

Implementation details

For workspace icons, we recommend using an AOT resource for the icon. While the out-of-the-box symbols will work, we recommend creating your own so that multiple workspaces don't use the same icons. For each workspace that needs an icon, create a new image file that adheres to the guidelines below. Note that the recommended guidance for newer versions of the product has changed.

Modeling details

When you create a workspace tile, you need to follow these guidelines:

- Add an AOTResource for each new icon.
- On the tile corresponding to the workspace, set the following properties:
  - ImageLocation=AOTResource
  - NormalImage=<name of AOTResource>

Icon creation

Guidelines for creating images for custom workspace tiles are below. The recommended dimensions for the image and icon are based on the out-of-the-box workspace icons. While images of other sizes are allowed, the size and positioning of the icon relative to the full image should be maintained regardless of the image size.

Following these recommendations ensures that your workspace icon matches the styling and size of other workspace icons and that the content of your workspace icon does not get cropped by the CSS applied to the image.

- The image file should be a PNG file with a 1:1 aspect ratio.
- The recommended minimum image size is 50 × 50 pixels (px), where the icon is contained in a square that is centered in the image. For the minimum 50 × 50 px image size, the icon should be contained in a **30 × 30 px** square in the center of the image.

  > **NOTE**
  >
  > The crispness of out-of-box workspace icons and custom workspace icons might differ at different zoom levels. The reason for this difference is that images in PNG format have a fixed resolution, whereas out-of-box workspace icons are font glyphs that scale smoothly. For better resolution at different zoom levels, consider creating larger images with the same relative dimensions. For example, create 200 × 200 px or 400 × 400 px images.

- The icon should have a **white background with transparent content**.

- The framework will set a default background color for the transparent portions of your image so that it will match the current user theme.

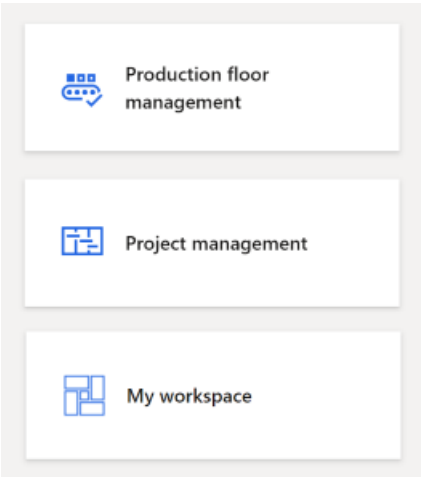
Example

Consider the following image/icon that is to be used for a new workspace.

This icon would be converted to an image with a **white background and transparent content** with the icon centered in a larger image canvas as shown.

To understand how this relates to the sizing recommendations, here is the workspace icon image overload with the new sizing recommendations.

Using this image on a workspace tile yields the following result on the dashboard.



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Public JavaScript APIs for extensible controls

2/18/2021 • 2 minutes to read • Edit Online

This topic shows where to find documentation about the public JavaScript APIs that can be used by extensible controls.

To minimize future breaks in extensible controls, an effort has been made to differentiate between the public and non-public JavaScript application programming interfaces (APIs) available to extensible controls. Control authors should ensure they only use public APIs, as **any non-public API may be removed or modified in a future release**. One of the planned modifications is to prefix the names of the non-public APIs with underscores to clearly denote access level. Documentation for the full set of public APIs can be found in Extensible Controls - Public JavaScript APIs.

Control checklist

2/18/2021 • 14 minutes to read • Edit Online

This article categorizes and describes all the release criteria for controls.

Introduction

Typically, when you author a new control, the primary focus is on scenario functionality and technical implementation. However, before a control can be considered ready for shipment, it should conform to a set of best practice, quality, and development release criteria, as outlined in this article.

Control criteria checklist

This checklist assumes that you're familiar with the basics of control development. The following items highlight important implementation requirements that should be met by all controls.

Basic usage ready

A control must meet these requirements to be considered a functionally compatible and complete control.

Classes

These naming conventions are best practices but aren't functional requirements of a control:

- The Runtime class is named **[Name of control]Control**.
- The Design-time class is named **Build[Name of control]Control**.
- Any control-specific Component classes are named **[Name of control][Name of component]Component**.
- Any generic Component classes are named **Build[Name of component]Component**.

Resources

These naming conventions are best practices but aren't functional requirements of a control:

- The HTML Resource is named **[Name of control]HTM**, and the physical file is named **[Name of control].htm**.
- The JavaScript Resource is named **[Name of control]JS**, and the physical file is named **[Name of control].js**.

X++ Runtime

These items apply to the "Runtime" X++ class.

- FormControlAttribute is supplied with the build class name.
- The FormControlAttribute is supplied with the template ID.
- FormControlAttribute is supplied with the resource bundle path.
- The **FormTemplateControl** class is extended (directly or through inheritance).
- A FormProperty exists for each change-tracked property (that is, each property that must be read in the client JavaScript).
- The **New** method initializes each FormProperty instance.
- The **setTemplateId** and **setResourceBundleName** inherited methods are called with the same values that are supplied in the FormControlAttribute.
- The **ApplyBuild** method is used to interpret any design-time properties, and to apply design-time properties to the run-time properties as appropriate for the control.

- A Property getter/setter method exists for each FormProperty.
- A FormPropertyAttribute is supplied on each FormProperty's getter/setter method.
- **Anytype** is used as the argument type for FormProperties that have **FormPropertyKind::BindableValue**.
- All FormProperties are specified as **ReadOnly** to the JS class, via the third argument to the FormPropertyAttribute.
  - This argument affects only the read/write behavior that is seen by JavaScript, not by X++.
  - We don't recommend that you allow JavaScript to write directly to properties, because every property state change should be validated. We recommend that you use FormCommands for this purpose, instead of writeable properties.
- FormCommands that allow the state of FormProperties to be changed must validate that the control is in a valid state to allow for the property change.
  - Controls that are disabled, read-only, invisible, and so on, should prevent inappropriate state changes.

X++ Design time

These items apply to the "Design/Build" X++ class:

- The FormDesignControlAttribute is supplied with the control common name, **[Name of control]**, without "Control" appended at the end.
  - The name that is supplied here will appear in Microsoft Visual Studio when the control is added to a form.
- A backing field exists for each design-time property.
- A Property getter/setter exists for each design-time property.
- A FormDesignPropertyAttribute is supplied to each design-time property.
- No code outside of the Design property getters/setters should exist in this class. (In other words, there should be no **new()** methods, and so on.)

HTML

These items apply to the .htm file, which is also referred to as the resource bundle:

- External resources, such as scripts, style sheets, and other HTM files, are loaded by using HTML standard <script> and <link> tags.
- All external resource loading tags are placed above the outermost HTML element in the file, so that they are loaded and processed first.
- The template ID is supplied via the **id** attribute on the outermost HTML element.
- The visibility of the outermost HTML element is bound to the **Visible** property.
- The sizing of the outermost HTML element is bound to the sizing binding handler.
- Binding handlers are used to programmatically modify HTML. (APIs such as getElementById in the JavaScript constructor aren't used.)

JavaScript

- The whole JavaScript code is wrapped in an anonymous function.
- Localizable strings are stored in the Globalize culture info object.
- Localizable strings are also stored in a label file, according to the instructions in Create localizable labels.
- Default values are provided for all properties that aren't initialized inside the constructor.
- The JavaScript constructor is added to the control JavaScript namespace.
- A reference to **this** is stored in an object that is named **self**, and **self** is used instead of **this** throughout the constructor.
- The base JavaScript control behaviors are inherited.
- Default values are applied by using a framework utility function.
- Client-side properties are defined in the scope of the constructor and are added to **self**.
- Observable and computed properties are used only for UI-bound behaviors.

- A prototype exists and contains all static methods that are specific to the control.
- Binding handlers that are specific to the control are stored in the control's namespace (not in the global control namespace).
- The control doesn't use or load external plug-ins (Microsoft ActiveX, Flash, Java, and so on).

Interactivity

CSS/LESS

- Prefix all class names with the template ID to prevent conflicts with other controls.
- Don't use class names that are defined on other controls, because those classes can change.

Layout and resizing

- The control uses the Sizing API for its outermost element. This requirement helps guarantee that the framework can correctly size and arrange the outermost element of the control.
- For advanced layout scenarios for elements that are contained in the control, use CSS Flexible Boxes, which are supported by the HTML standard.

Browser support

- The control correctly renders and supports all intended user interaction patterns on all supported browsers:
  - Microsoft Edge/Internet Explorer 11
  - The latest version Chrome
  - The latest version of iPad/MacOS Safari

Tab sequence

- Make sure that the control meets the W3C standards for tab sequence.

Globalization

Right-to-left languages

- Full RTL support will arrive after RTW.

Localizable labels

- The control uses a label file for UI text that is used only on the client side (not used in X++). For instructions about how to create and use these labels, see Create localizable labels.

Task Recorder compatibility

- Basic recording support
  - For any control that accepts user input, or that a user can interact with, input/actions must be recordable by Task Recorder. For Task Recorder to record the input, the control must use the SysTaskRecorder X++ API to specify the properties that should be recorded.
- Basic task guide support
  - For any control that can be recorded by using Task Recorder, the control should have task guide support. This support includes verifying that the task guide pop-up prompt points to the correct UI elements of the control, based on the input/action that was recorded.
  - In addition, validate that, when the task guide is locked ("on-rails"), the user can interact with the expected parts of the control. For example, for a combo box control, the user should be able to open the drop-down box to select a value, in addition to typing the value directly.
- Advanced recording support
  - Support for Cut, Copy, Paste, and Validate can be evaluated on a per-control basis. There are JavaScript and X++ methods that controls can implement to enable these features.

Threat modeling the control

- Vulnerabilities in the logical control (X++/C++) that are related to serialization/deserialization of data types, and command/property execution, should be reviewed and fixed.
  - Serialization threats are related to the way that the control parses or interprets data types.

    - Any serializer for data types (except the built-in X++ Data Contracts and primitive type serialization) must be reviewed.
    - Any parser must be reviewed to verify that it doesn't allow for arbitrary code execution or other exploitation.
    - Any data access queries that are executed by the control (or by any helper classes that the control uses) must be evaluated and reviewed.
  - Command/property execution threats are related to the way that the logical control handles an action that the control determines to be invalid.
    - Example of a command threat: A click command is executed when the control is in a disabled state. You must make sure that the click command is handled appropriately, based on the control state.
    - Example of a property thread: A property change is executed when the control is ready-only. You must make sure that the property change is handled appropriately, based on the control state.
  - You must review the use of any .NET libraries to make sure that the .NET library is also secure.
- Vulnerabilities that are related to the client-side parts of the control (HTML, JavaScript, CSS, third-party libraries) should follow general secure web development principles. Specific examples include XSS vulnerabilities.
  - Any control that renders user data as HTML/JavaScript/CSS exposes an XSS vulnerability that must have mitigations identified.
  - Any control that renders external content in an iFrame exposes an XSS vulnerability that must have mitigations identified.
  - Any control that makes calls to third-party services must have mitigations identified and requires direct review by the client team.
  - Any control that handles authentication must have mitigations identified.

Control criteria details

This section explores the control criteria in more detail.

Basic usage ready

X++ Runtime

- **FormControlAttribute** Each control must supply the FormControlAttribute to the class declaration. The attribute must specify the build/design-time class that accompanies the control. The attribute must also specify the HTML template ID and the physical HTML file name (the resource bundle name).
- **FormTemplateControl** Each control must extend FormTemplateControl to participate in the control lifecycle.
- **FormProperty** Each control must declare FormProperties for every statically defined property that must participate in the change tracking system. For properties that are used only on the server side, no FormProperty is required.
- **New** Each control must implement the **New** method in order for its properties to participate in the change tracking system that propagates value changes between the client and server parts of the control. Inside the **New** method, each FormProperty
- **ApplyBuild** Each control must implement the **ApplyBuild** method in order for the control to be initialized based on the values that are set on it at design time. This method is primarily used to copy or transform design-time values into their Runtime equivalents. However, not all design-time properties must have Runtime equivalents, and not all Runtime properties must source their initial values from design-time properties.
- **Property getter/setter** Each control must implement property getters/setters for every FormProperty that is used by the control. These methods should be parm methods. Therefore, the names should begin with “parm” for methods that are getters/setters, “get” for methods that are only getters, and “set” for methods

that are only setters. At a minimum, a FormPropertyAttribute must be supplied to each method, together with a FormPropertyKind and the name of the property as it should be made accessible to the HTML and JavaScript in the client.

HTML

- **Template ID** Each control must provide an HTML **id** attribute on the outermost HTML element of the control's markup. In order for the control to be loaded at run time, this ID must match the ID that is supplied to the FormControlAttribute.
- **Scripts and style sheets** Each control must use HTML standard <script> and <link> tags to consume other JavaScript or CSS files. These tags should be placed at the beginning of the HTML file for the control, before the HTML definition element for the control. If the files that must be loaded have dependencies, make sure that the order of the <script> or <link> loading tags is appropriate. Tags that appear first are loaded first. To load JavaScript or CSS from AOT Resources, use a site root–relative path (/Resource/Scripts or /Resources/Styles).
- **Data binding** Each control can participate in the HTML binding framework through use of the **data-dyn-bind** attribute on HTML elements that are contained in the control. The binding attribute enables HTML element properties to be bound to observable or computed properties that are located in the current data context.

JavaScript

- **Script encapsulation** Each control must wrap all its JavaScript in an anonymous function. This requirement helps prevent the framework's global JavaScript namespace from being populated with control-specific logic.
- **Localizable strings** Each control must use the Globalization API to store any string messages that are used by the control's JavaScript. Therefore, the control's JavaScript should not hard-code any strings that are displayed in the UI. Instead, the JavaScript should reference the string messages that are stored via the Globalization API. To load strings in the globalization object in HTML and JavaScript, you can use the $dyn.label API and pass in the identifier of the label. For more information, see Create localizable labels.
- **Default values** Each control must provide default values for any properties that aren't initialized in the JavaScript. Therefore, any properties for which values are passed in to the JavaScript constructor on initialization (that is, the FormProperties in the X++ Runtime class) must provide a default-value dictionary for these properties.
- **JavaScript constructor** Each control must implement a constructor in the controls namespace. This constructor is the first line of code that will be executed when the control is loaded in the client. After the constructor is completed, the constructor's associated object, *this*, is passed in to the HTML as the default data context.
- **Inheriting base control** Each control's constructor must "inherit" from the base JavaScript control class. The base JavaScript control class contains behaviors that are required by each control.
- **Applying default values** Each control must use the provided framework function to apply the default values to the control's properties.
- **Adding client-side properties/functions** Each control can add client-side-only properties and functions to the JavaScript class, in addition to the server-side FormProperties and Commands that are passed in to the control constructor. The pattern for adding client-side-only properties/functions is to maintain a local copy of the *this* object and add the functions/properties to the local copy. After the control constructor is completed, all properties, functions, FormProperties, and Commands that have been added will be available in the HTML as the default data context.
- **Adding observable and computed properties** Each control can add client-side-only properties that participate in the observability patterns of the client. An observable property is initialized by using the $dyn.observable([initial value]) function. A computed property is initialized by using the $dyn.computed([function(){}]) function. Controls should use observable/computed properties sparingly, because these property can significantly harm performance if they are used incorrectly.
- **Control JavaScript prototype** Each control must implement a JavaScript prototype that extends the base control prototype. The prototype should contain any "static" JavaScript methods (methods that require no

references to local variables) that are used by the control.

Interactivity

Layout and resizing

- To enable the form developer to determine the control layout and size, set the width and height that are specified by the form developer on the control by using the $dyn.layout.sizing API, as shown in the following code. This is standard code that should be applied to all HTML control templates.

```
<div id="MyControl" data-dyn-bind="
sizing: $dyn.layout.sizing($data)>
</div>
```

Task Recorder recording support

Controls must use the SysTaskRecorder X++ API to indicate which actions on the control are "recordable."

- For controls that enable values to be set via properties, **SysTaskRecorder::addPropertyUserAction** should be called when the value is being set in X++. This method call tells Task Recorder to record the setting of the property.
- A similar method exists for commands (**SysTaskRecorder::addCommandUserAction**).

For more information, see Control the text that Task Recorder generates for a control.

Task Recorder playback support

- Task Recorder uses the control's properties and commands to play back the control. The control must make sure that the commands and properties that it instructs Task Recorder to record can also be executed by Task Recorder when the control is played back. Task Recorder will rely on the interactable names for the properties and commands. The interactable name for a method is the name that is specified in the FormPropertyAttribute or the FormCommandAttribute.

Task guide support

The task guide will ask the JavaScript part of the control for the DOM element that the task guide should point to. All controls inherit basic task guide support, where the default DOM element is the outermost element of the control.

- A control can provide finer-grained details about where the task guide should point by implementing the getTaskGuideParams function in the JavaScript prototype for the control. This function accepts an argument (which is frequently named **options**), and this argument has a property that is named **target**. This **target** property accepts the jQuery element that the task guide should point to.
- In addition, the argument will contain information about the action that was originally recorded for the control, such as the property/command name and any arguments. The control can react to the various properties/commands that it supports by supplying the target with the DOM element that corresponds to the property/command that the user recorded.
- For some advanced scenarios, the target can also be supplied with an observable. The control can then update this observable with various DOM elements, based on events that the control exposes. This can be done by initializing the target with the observable (which contains a DOM element), and then observing an event and updating the DOM element via code in the event handler.

Task Recorder copy/paste/validate support

For controls that need to support either Copy, Paste, or Validate (mainly for advanced X++ testing purposes), the SysTaskRecorder API exposes static methods that enable the control to inform Task Recorder when a value has been copied, pasted, or validated.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

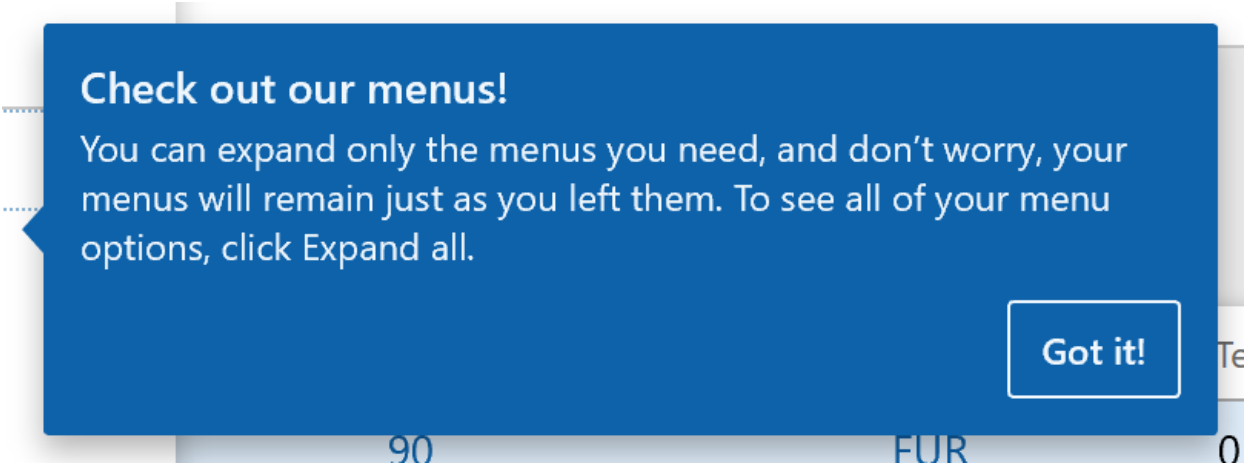
Feature callouts

2/18/2021 • 2 minutes to read • Edit Online

Introduction

While documentation is helpful for explaining new features, it's also important to raise awareness of these new capabilities as users encounter the feature while using the product. As a result, feature callouts are available in Platform update 26. You can use feature callouts to point out a new capability to a user and optionally provide a hyperlink for the user to learn more about the feature.

In this topic, the APIs that are used to construct feature callouts are discussed in detail.



The "Got it" button

When a feature callout is triggered, the user can simply click the **Got it** button to dismiss the popup. This saves the state of this feature callout in the personalization subsystem, which prevents that specific feature callout from being triggered again.

Resetting feature callouts

Even though the feature callout state is stored in personalization, clearing personalizations will not delete the state of all previously dismissed feature callouts. Instead, separate actions have been added to reset all feature callouts so that they fire again. These actions are located on the **Personalization** tab on the **Usage data** page as well as on the **Manage per user** tab on the **Personalization** page.

Disabling feature callouts

If needed, administrators can turn off feature callouts for an environment using the **Feature callouts enabled** option on the **Client performance options** page.

Implementation details

The SystemNotificationsWhatsNewManager class contains two variant APIs for triggering a feature callout.

AddWhatsNewWithActionLink()

Add a feature callout to a control with a "Learn more" link that is configured to open the documentation associated with the new product capability.

Parameters

PARAMETER	DESCRIPTION
ruleID	Generate a unique GUID.
title	Provide a (localized) title.
bodyText	Provide a (localized) description.
targetControl	Provide the name of the control you want to attach the feature callout to.
urlLink	Provide the URL to open in a new tab when the "Learn more" link is clicked. If a URL is not specified, then a "Learn more" link will not be displayed.

AddWhatsNew()

Add a feature callout to a control without a "Learn more" link.

Parameters

PARAMETER	DESCRIPTION
ruleID	Generate a unique GUID.
title	Provide a (localized) title.
bodyText	Provide a (localized) description.
targetControl	Provide the name of the control you want to attach the feature callout to.

Example

The following code snippet will trigger a feature callout attached to the control named *TestStringControl*.

```
public void init()
{
     super();

     SystemNotificationsWhatsNewManager::AddWhatsNewWithActionLink(
          MyTestKey,
          "My title" ,
          "My description",
          TestStringControl.name(),
          "https://www.microsoft.com"
     );
}
```

Notes

- Multiple feature callouts can be shown on a page at one time.
- Only one feature callout is allowed per control. If multiple callouts exist, the last one to get triggered will be displayed.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Slider and MessageBox dialogs

2/18/2021 • 2 minutes to read • Edit Online

Dialogs

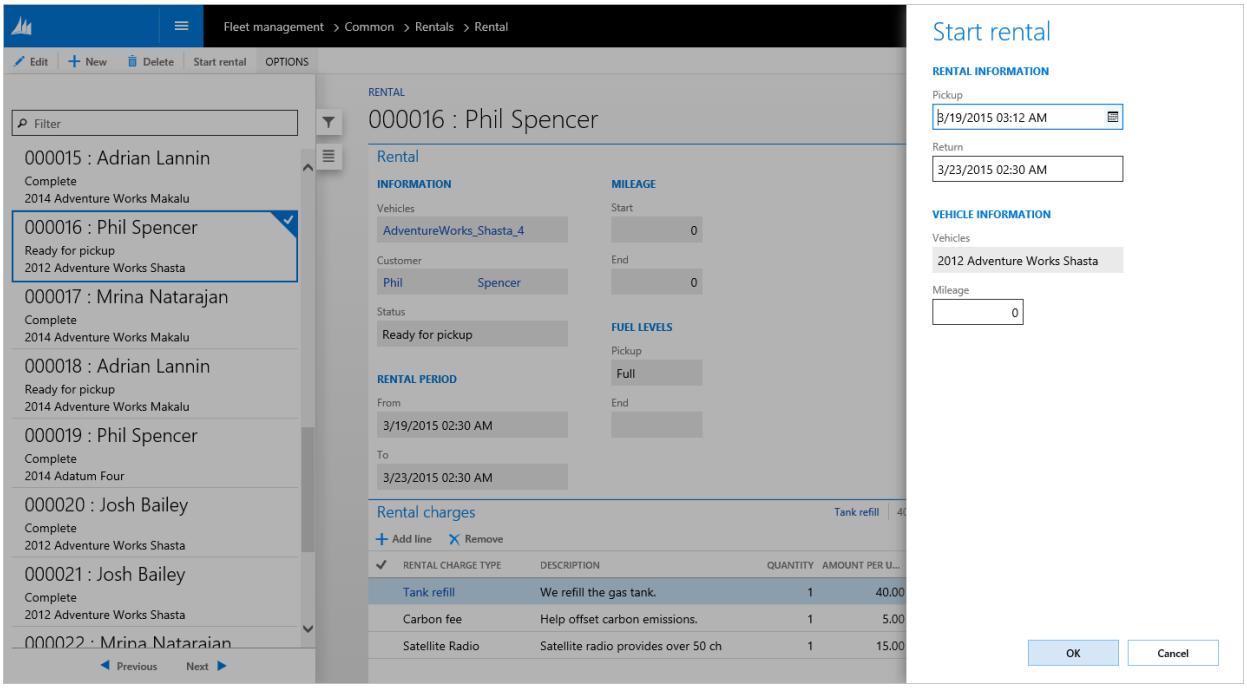
There are two dialogs that replace the existing dialog box, the Slider and the MessageBox from Dynamics AX 2012:

- Slider
- MessageBox

The following sections discuss the specific goals for each concept.

Slider

The slider, or slider dialog, is a dialog box that "slides" in on top of the active page's content from the right edge of the screen. In the following screen shot, the slider is the white region that has the caption **Start rental** on the right side of the window. Notice that the area to the left of the slider is shaded to help the user understand that the page beneath the slider isn't currently available for interaction.



After a slider opens, the user can dismiss it in two ways:

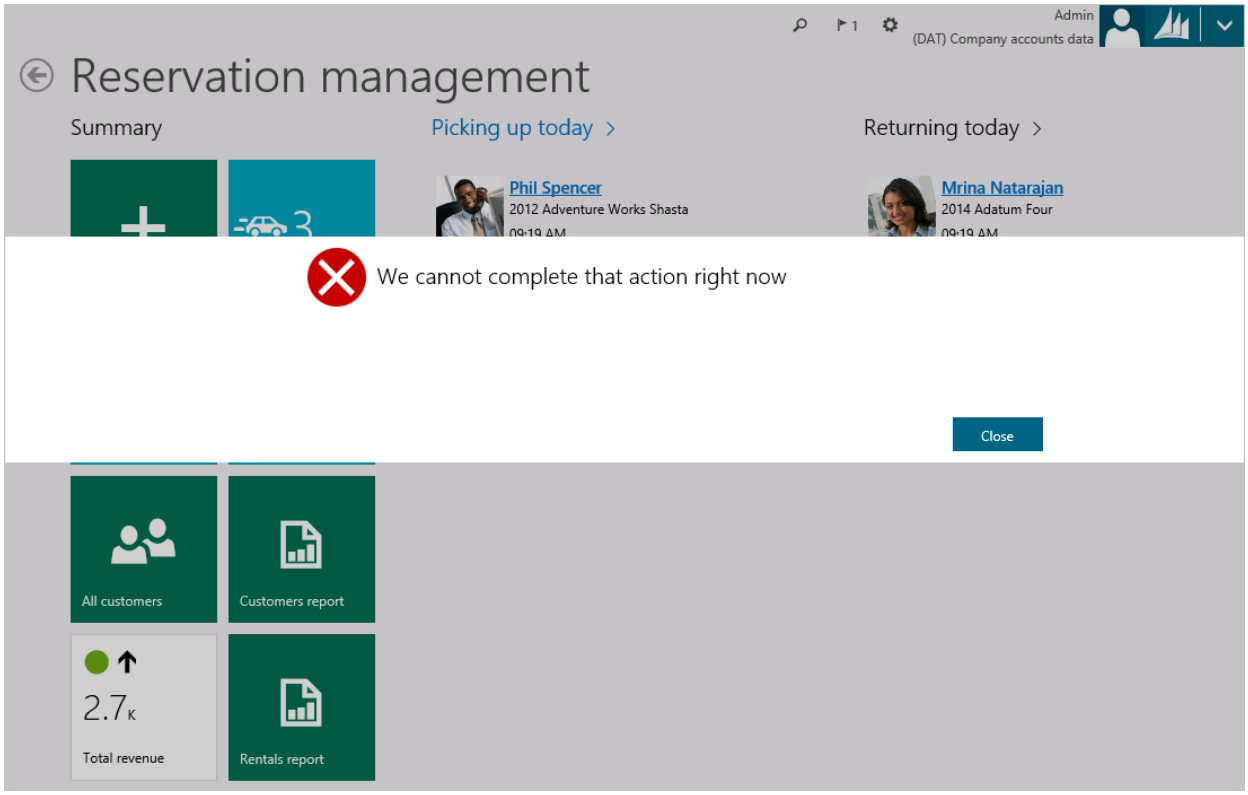
- Perform an action within the slider that causes the underlying form to dismiss itself. For example, click **Cancel**, or enter required information and then click **OK**.
- Click outside the slider in the shaded area to the left. This cancels the slider, and no further actions are performed.

A slider contains a modeled form and is used to gather information from the user. Therefore, a slider should be used in most situations where a dialog box has been used in the past. For example, a slider is typically used when the user creates a new record, as in the preceding screen shot. However, a slider should not be used for simple notifications or messages to the user. For these situations, a MessageBox should be used, as described in the next section. To model a slider, you create a form, and then set the **Style** property to **Dialog** on the **Form.Design** node. You then model the form elements that you require (for example, fields and buttons). The

caption is defined by **Form.Design.Caption**. To simplify the process for creating sliders, we have provided the **SysBPStyle_Dialog** form as a template for modeling slider dialogs. To use this template, copy it into a new form, and then extend it as you require.

MessageBox

A MessageBox is a type of dialog that is rendered as a "lightbox" on top of an existing page. A MessageBox appears as a full-width modal pop-up. The following screen shot shows an example of a MessageBox.



A MessageBox is the correct mechanism to use when you must interrupt the user to notify the user about a critical situation. For example, a MessageBox is used to display a Message center error message to the user. Because a MessageBox is modal, the user can't interact with the page beneath the MessageBox until that MessageBox has been dealt with or dismissed. In the preceding screen shot, notice that the page is obscured by the MessageBox. Additionally, the areas above and below the MessageBox are shaded to help the user understand that the page isn't currently available for interaction. Be aware that, unlike a slider, the user can't dismiss a MessageBox by clicking outside it, in the shaded areas. A MessageBox can be triggered by using either the Box application programming interface (API) or any of the methods that are described earlier for triggering the display of an error. For more information, see the Message API: Message center, message bar, message details.

Messaging APIs - Action center, message bar, and message details

2/18/2021 • 5 minutes to read • Edit Online

This topic describes the messaging system in Finance and Operations apps, specifically in terms of the application programming interfaces (APIs) that are used to create and route messages to end users.

Introduction

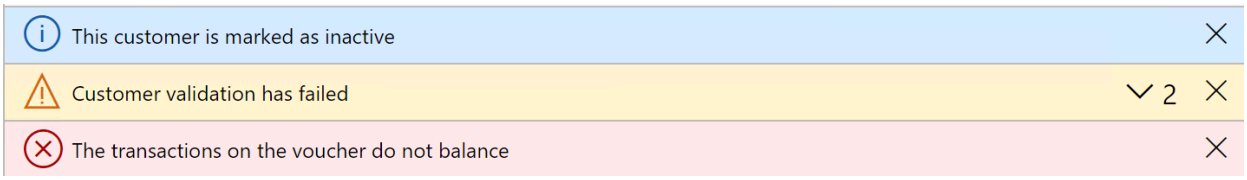
A new messaging system was created for Finance and Operations apps to improve this experience. Compared to earlier versions, the messaging system for Finance and Operations apps includes the following features:

- Improved association of a message with its context (form versus global).
- Improved level of interruption (none, subtle, and interrupting).
- Improved clarity between types of messages and their use.
- The control that is used to display messages is deterministic and based on form context.

Backwards compatibility of info(), warning()/checkfailed(), and error()

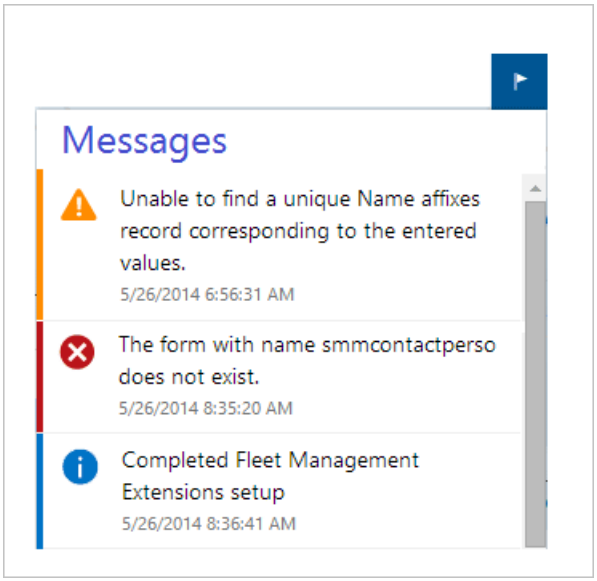
The **info()**, **warning()**, and **error()** application programming interfaces (APIs) from earlier versions of Finance and Operations apps are still supported; however, these APIs now sit upon the framework's new messaging system. Messages are routed deterministically to the message or Action center (in a non-interrupting manner) by using the context of the API call to determine the best way to present the message to the user. In general, if the use of the API originated from a form, the message appears in a message bar on that same form. (Drop dialogs and slider dialogs are both considered forms.)

The following illustration shows **info**, **warning/checkfailed**, and **error** message bars that correspond to page actions, or synchronous-authored messages that come from **info()**, **warning()**, and **error()**.



> **NOTE**
>
> If these APIs are called from a slider dialog, but that slider dialog is closed before the message appears, the message is shown in a message bar on the slider dialog's parent page. If that slider dialog is closed before the message appears, and there is no parent page, the message is routed to the Action center. The messaging API never fails to show a message. If an appropriate host page isn't found, the message is sent to the Action center.

If **info()**, **warning()/checkfailed()**, or **error()** is called from an asynchronous process (for example, a batch), there is no form context to consider, and the messages are sent to the Action center. (To open the Action center, click the **Show messages** button on the navigation bar.) The following illustration shows examples of each type of message in the Action center.



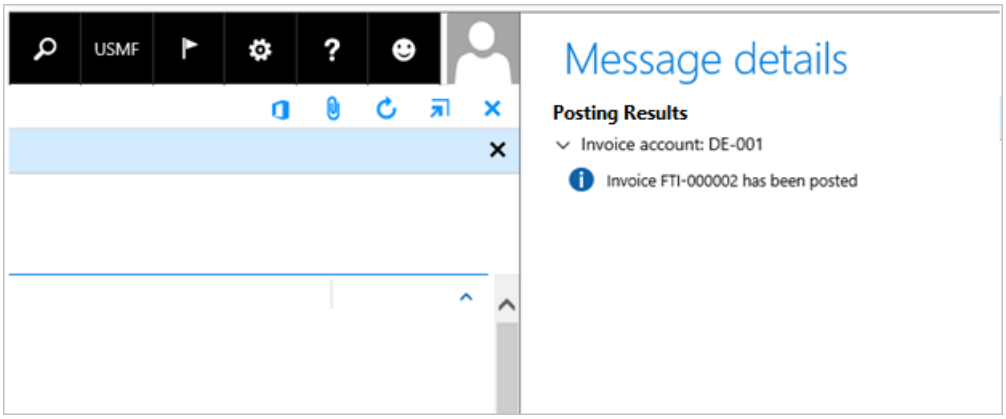
NOTE

Use the **Box()** API to express an interrupting error to the user.

Backwards compatibility of SetPrefix()

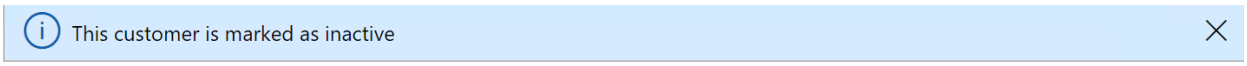
Finance and Operations apps also support the **SetPrefix()** API for backwards compatibility. However, in the messaging system, the results of **SetPrefix()** don't actively interrupt the user; instead, the results are collected and stored (as in previous versions), and a message bar or Action center notification is presented to the user. This notification indicates that the related task has been completed, and that it might have messages that the user should review. The "Notification of results" message actually uses the task's first call to **SetPrefix()** to frame the message. This behavior is similar to the behavior in previous versions, where the first call was the "title" of the results. In this example, "Posting Results" comes from the application's first call to **SetPrefix()**.

The user can then click **Message details** to open the new **Message details** pane.



Message()

The **Message** API provides some useful messaging capabilities. The **Message()** API gives you more control over the lifecycle of a message by allowing you to explicitly add and remove messages. This API can be useful when validation messages need to be removed at times other than when a save boundary has been crossed, or for displaying informational messages about aspects of the user's experience that aren't necessarily related to data validation. In this example, the message is shown when the current record is displayed.



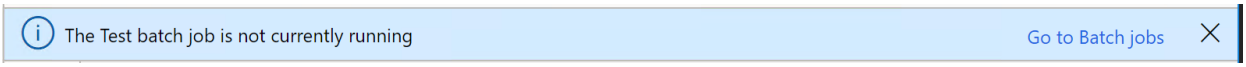

```
messageId = Message::Add(MessageSeverity::Informational, "The customer is marked as inactive");
```

The message can then be cleared when a new record is shown on the page.

```
Message::Remove(messageId);
```

Starting in version 10.0.10 / Platform update 34, you can use the **Message::AddAction()** method to embed an action within a message. This method supports adding a single action that is associated with a display or action menu item, which is then visualized as a link button. The actions are only supported in messages that are routed to the message bar until version 10.0.16 / Platform update 40, at which time these actions can be seen in messages that are routed to the Action center or the Message details pane.

In this example, a message is triggered for a system administrator indicating a particular required batch job is not running and exposes an action to go directly to the **Batch jobs** page.




```
MenuItemMessageAction actionData = new MenuItemMessageAction();
actionData.MenuItemName("BatchJob");
str jsonData = FormJsonSerializer::serializeClass(actionData);

int64 messageId = Message::AddAction(MessageSeverity::Informational, "The Test batch job is not currently
running", "Go to Batch jobs", MessageActionType::DisplayMenuItem, jsonData);
```

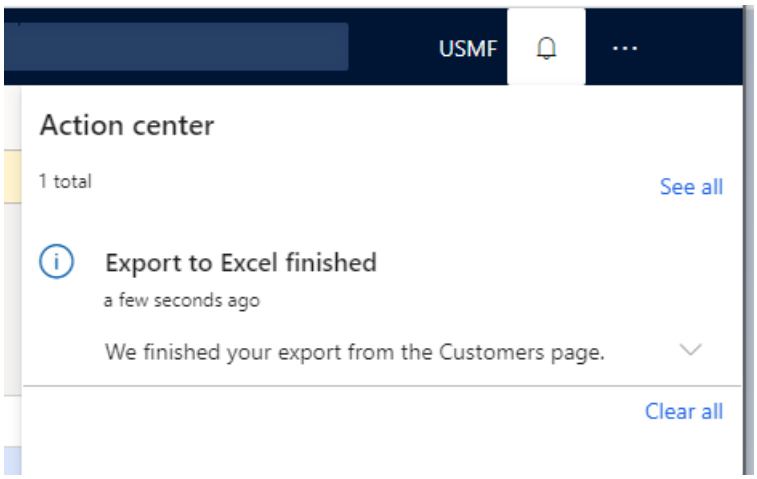
The following messaging types are supported: **MessageSeverity::Info**, **MessageSeverity::Warning**, and **MessageSeverity::Error**. Messages that use the **Message()** API are also deterministic. They can be routed to a message bar or the Action center.

SystemNotificationsManager()

The **SystemNotificationsManager()** API targets notifications designed to be sent to the Action center. This API provides the following features:

- Associating one or more actions to the notification
- Routing a notification to a set of users, or to all the users in one or more security roles
- Defining an expiration date for the notification
- Tracking the state of the notification (e.g. you can mark a notification as "Completed")

In this example, a notification is raised after an export to Excel is completed by a user. The message will be available in the Action center for the next 48 hours, after which the link to the exported file is no longer available.




```
// Set up the notification
SystemNotificationDataContract notification = new SystemNotificationDataContract();
notification.Users().value(1, curUserId());
notification.Title("Export to Excel finished");
notification.RuleId('ExcelStaticExport');
notification.Message("We finished your export from the Customers page");
notification.ExpirationDateTime(DateTimeUtil::addHours(DateTimeUtil::utcNow(), 48));

// Set up the action associated with the notification
SystemNotificationActionDataContract action = new SystemNotificationActionDataContract();
action.Message("Click to download");
action.Type(SystemNotificationActionType::AxActionMenuFunction);

SystemNotificationMenuFunctionDataContract actionData = new SystemNotificationMenuFunctionDataContract();
actionData.MenuItemName(menuItemActionStr(ExportToExcelStaticOpenFileAction));
actionData.Data(fileName);
action.Data(FormJsonSerializer::serializeClass(actionData));
notification.Actions().value(1, action);

SystemNotificationsManager::AddNotification(notification);
```

Additional resources

User interface development home page

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Messaging system for Finance and Operations apps

2/18/2021 • 10 minutes to read • Edit Online

This topic describes the rich, powerful messaging system in Finance and Operations apps.

A new messaging system was created for Finance and Operations apps to improve this experience. Compared to earlier versions, the messaging system for Finance and Operations apps includes the following features:

- Improved association of a message with its context (form versus global).
- Improved level of interruption (none, subtle, and interrupting).
- Improved clarity between types of messages and their use.
- The control that is used to display messages is deterministic and based on form context.

Where can messages be surfaced to users?

Messages in Finance and Operations apps are generally shown in one of these places: message bars, the Action center, or message boxes.

Message bars – Messages for synchronous tasks on the current page

Message bars are available on primary pages, and in drop dialogs and slider dialogs. Message bars are used primarily for data validation. They can also be used to communicate messages about the state of a page or data, such as messages that are used for date effectivity. Message bars can express **info**, **warning**, and **error** statuses. Message bars should not be used for messages that require the user's immediate attention. A message bar appears when a message is first received and must be used to communicate messages only about the current page. Messages that are sent to message bars are associated with the current page. Therefore, when the user navigates away from a page that includes message bars, those messages won't appear on the new page. However, if the user navigates back to the original page, the page's messages will once again appear. Include the following information in messages:

- The condition that generated the message.
- The result if the user continues without resolving the condition that generated the message.

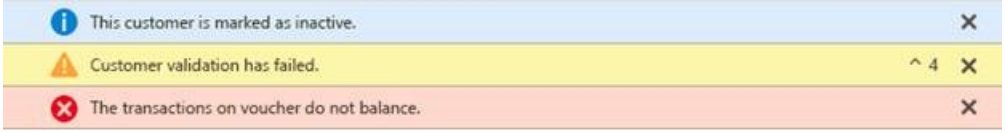
Examples

```
This customer is marked as inactive.
```

```
Customer validation has failed.
```

```
The transaction on voucher do not balance.
```

Presentation



Action center – Messages from asynchronous tasks

The Action center is located in the navigation bar. It contains messages that don't require any immediate action by the user and aren't required for the current task to continue. Typical examples include feedback from

background processes such as a batch job or report completion. The Action center can express **info**, **warning**, and **error** statuses. Before it's opened, the Action center indicates the number of messages that have been received since last time that it was opened.

The Action center can hold up to 500 messages. Messages are then cycled on a first in, first out basis.

Message boxes – Errors and immediate notifications (completed synchronous operations)

Use message boxes to alert users about issues that require immediate attention. Because message boxes interrupt users and prevent them from continuing until the message is read and dismissed, they should be used only for messages that users can't handle later. Include the following information in error messages that appear in message boxes:

- The error that occurred.
- The cause of the error.
- Information about how to resolve the error.

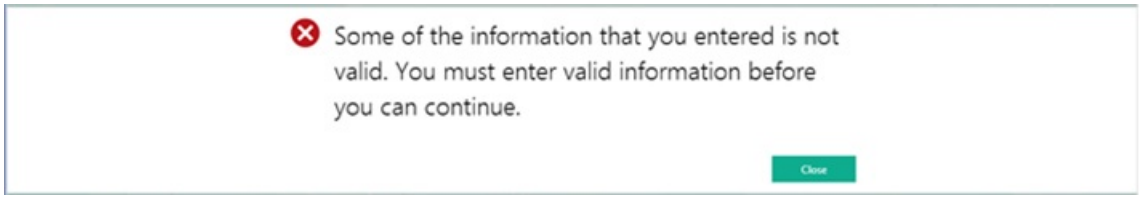
An error message should include the following two components:

- **The main instruction** – This text appears in bold.
- **The message details** – This text appears below the main instruction.

Examples

- You can't delete the reference number because the reference number is used in version %1.
- You have insufficient rights to perform this export.
- The root folder for catalog import processing is not configured. Configure the root folder using the Vendor catalog import parameters form.

Presentation



Messages of the **error** type block the user's interaction by overlaying the current page with a modal "light box" that contains the message.

Should I show the user a notification, a warning, or an error?

In earlier versions of Finance and Operations apps, the **info**, **warning** (**checkFailed**), and **error** statuses weren't always used consistently across scenarios. A message might be reported as a warning in one scenario but as an error in another scenario. When you're deciding which status to express, use these definitions:

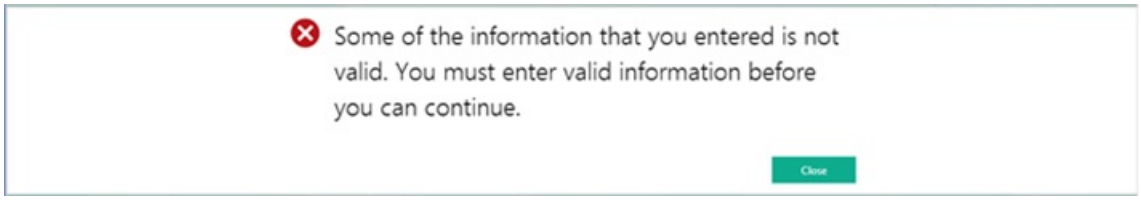
- **Notification** – A notification informs the user about events that might or might not be related to the current user activity. A notification can be caused by a user action or a system event, or it can provide information from the program that might be useful. Typically, a notification doesn't require immediate user action. You notify the user by using the **info()** application programming interface (API).
- **Warning** – A warning alerts the user about a condition that might cause an issue in the future. Specifically, a warning is used for data that is in an incorrect state. Although any attempt to use this invalid data might produce an error, the fact that the current state of the data is incorrect isn't an error condition, and *the user should only be warned about the incorrect state of the data*. You express data validation issues by using the **warning()** or **checkFailed()** API.

> **NOTE**
>
> Because of the way that the system handles the cleanup of validation messages, when the **warning()** or **checkFailed()** API is used, the return value of **validate()** should *always* be set to **false**. Otherwise, warning messages might not be shown to the user.

- **Error** – An error alerts the user about a problem that has already occurred. A user action that has failed is an error condition. Errors can be *non-interrupting* (*passive*) or *interrupting*. In a non-interrupting error, users can perform other activities before they try to correct the issue. In an interrupting error, users can't proceed or complete the task until they correct the error condition. You express a passive (non-interrupting) error by using the **error()** API. You express an interrupting error by using the **box::** API.

Should this message interrupt the user?

If a task (batch job or other operation) fails, it's often appropriate to notify the user passively. Because the user can correct the issue and retry the operation at any time, the user doesn't have to be notified immediately. In these cases, the **error()** API is appropriate, and the user doesn't receive an interrupting dialog. However, in other cases, the user can't proceed until the issue is corrected. For example, if the user tries to save a page that still has invalid data, the client interrupts the user by presenting an error dialog. In these cases where it's more appropriate to interrupt the user by presenting a dialog, the **box::** API should be used.



Will my message end up in a message bar or in the Action center?

The messaging system is *deterministic*. In other words, the messaging system uses the context of the call to determine the best way to show the message to the user. Messages are shown either in a message bar that appears at the top of pages or in the Action center, which appears in the navigation bar. The location of the message depends on where in code the message is sent from.

In general:

- If the message is caused by a page action that is synchronous (that is, the user must wait for the result), the result is shown in a message bar on the current page. (The exception is a slider dialog that was closed immediately after the action was started. Messages for slider dialogs "bubble up" to the parent page.)
- If the message is caused by an action (for example, a batch job) that is asynchronous (disconnected) and the user can continue to perform other tasks or even navigate to another page while that action is being processed, the message is routed to the Action center.

Messaging from asynchronous or long-running background tasks

A (potentially) long-running task should not present a message bar to the user, because message bars at the top of a page are used to present information about the current page, not some background task that might have started hours earlier. In some cases, a user who has many background tasks running continues to navigate between pages while the tasks are being completed. Therefore, messages that are presented on the current page to notify the user about background tasks are easily overlooked or ignored. Therefore, by design, background tasks send their messages to the Action center. When a new message appears in the Action center, a notification informs the user, who might be waiting for the results of an asynchronous task.

Messaging from dialogs and slider dialogs

The deterministic messaging system tries to send messages to the current page. However, not every call from a

dialog or slider dialog is routed to that dialog or slider. In some cases, the messaging system sends the message to the parent page instead. This behavior can occur when the messaging system is called while the dialog or slider is being closed. In some cases, the messaging system can be called when the close process for the dialog or slider is started, but the client interrupts the close process for valid reasons. Therefore, there is a "point of no return," after which the messaging system no longer tries to send a message to the dialog or slider, and instead sends the message to the parent page. When the user clicks the **OK** button on the form is entering its closing sequence, shown in the code example that follows.

```
closeOK()
{
    // current form
    super(); // calls close()
    // parent or message center
}
Close()
{
    // current form
    super();// point of no return
    // parent or message center
}
```

If the client calls **closeOK()** or **close()** directly, then the final result might be the page or the parent page.

When are validation messages cleaned up?

With the messaging system in use for Finance and Operations apps, the validation message (called using the same APIs) appears in a message bar on the page itself in a passive manner. The invalid value remains but is flagged as invalid. The user can continue to enter data and can correct the validation issue at any point before the data is saved.

When a validation issue has been corrected so that the corresponding message in the message bar is no longer valid, the messaging system removes the message. The timing of message removal depends on the level where the validation logic is defined.

- If the validation logic is defined at the control or field level, the message is removed when a valid value is entered in the control or field.
- If the validation logic is defined at the table level, the message is removed the next time that the user crosses a save boundary.

If the developer needs more control over when a message needs to be removed from the UI, the **Message()** API can be utilized. See the Messaging APIs article for more details.

I'm migrating from an older version. How do I change my existing code to use the new messaging system?

In many cases, no changes are required. The messaging framework was designed to innovate and maintain backward compatibility for many common scenarios. In some cases, the program might improve the wording of messages. Alternatively, the program might use **error()** instead of **warning()**, or **warning()** instead of **error()**, to better align with the usage guidance (warnings are for data that isn't valid, whereas errors are for failed actions). In other cases, you might decide that messages that appear on a slider dialog are more appropriate for the parent page.

How to create a collection of related messages?

You use **SetPrefix()** to create collections of related messages [See the Messaging APIs for more details on **SetPrefix()**. This API is largely backward compatible but is presented in a non-interrupting manner. A results

window isn't opened directly; instead, the user is passively notified by either an Action center message or a message bar on the page that started the task that used the **SetPrefix()** API to group the result messages into a collection. The message severity shown to the user reflects the severity of the most critical message in the collection. For example, if the collection contains no errors or warnings, the message bar is of the **info** type.

If the collection contains one or more calls to **warning()**, the message bar is of the **warning** type.

If the collection contains one or more calls to **error()**, the message bar is of the **error** type.

The use of **SetPrefix()** is also deterministic. In other words, if you use **SetPrefix()**, and there is no page context (for example, an asynchronous batch operation), the notification of results is sent to the Action center, which isn't associated with any page.

Form patterns for migrated forms

2/18/2021 • 12 minutes to read • Edit Online

This topic provides information that will help you select the best form pattern for the forms that you migrate.

Introduction

The selection of a form pattern is an important step in the process of migrating a form. A pattern that is a good fit for the target form reduces the amount of migration work that is required. By contrast, a pattern that isn't a good fit can cause wasted time and effort. Therefore, it's important that you do some investigation, so that you can select the best form pattern for the form that you're migrating. Here is some guidance and tips for determining the appropriate pattern for a form:

- Investigate the form's metadata in the form designer. Pay close attention to the following details:
  - Form name
  - Form.Design.Style
  - Control names
  - The way that the controls are organized
  - The number and names of the data sources
- Investigate the form's visuals by running the form and looking at the way information is displayed.

Selecting a form pattern via metadata

Use Form.Design.Style for guidance

The **Form.Design.Style** property often contains the name of the pattern that was previously targeted for the form. If the **Style** property correctly matches the metadata, you can use the following table to find a pattern that is likely to be a good fit for the form.

FORM.DESIGN.STYLE VALUE	CORRESPONDING PATTERN
DetailsFormMaster	Details Master
DetailsFormTransaction	Details Transaction
Dialog	Dialog
DropDialog	Drop Dialog
FormPart, where there are just fields	Form Part FactBox Card
FormPart, where there is a grid	Form Part FactBox Grid
ListPage	List Page
Lookup	Lookup
SimpleList	Simple List

FORM.DESIGN.STYLE VALUE	CORRESPONDING PATTERN
SimpleListDetails, where there are 2–3 fields in the navigation list (recommended)	Simple List Details – List Grid
SimpleListDetails, where there are and 4–5 fields in the navigation list	Simple List Details – Tabular Grid
SimpleListDetails, where there is a tree (rare)	Simple List Details – Tree
TableOfContents	Table of Contents
Auto, where there is an **Overview** tab, a **General** tab, and a single data source	Task Single
Auto, where there are two sets of **Overview** tabs, **General** tabs, and/or headers plus lines	Task Double
Auto, where there is focus on a single record	Simple Details
Auto, where the form name ends in "Lookup"	Lookup
Auto, where there is a single tab control and **Next**/**Previous** buttons	Wizard
Auto, where the form name ends in "Wizard"	Wizard
Auto, where there is just a grid and some buttons	Simple List

When a form doesn't match the Style property

Sometimes, a form has an incorrect **Form.Design.Style** property value.

FORM.DESIGN.STYLE VALUE	WHAT THE FORM MIGHT ACTUALLY BE
DetailsFormMaster	DetailsFormTransaction, if there is lines detail, or if controls have names that contain "lines"
SimpleList	SimpleListDetails, if there is more than just a grid and some custom filter fields
SimpleListDetails	SimpleList, if there is just a grid and some custom filter fields
SimpleList	ListPage, if there are numerous FactBoxes in the **Parts** node, or if the form has a corresponding Details Form

Selecting a form pattern via visuals

Although this approach is less useful than looking at the form metadata, you can get a lot of information about a form by running and examining it. Use the form visuals as an additional data point to help you select a form pattern. Look through the screen shots of migrated forms to find a form that looks like the target form. Additionally, make sure that the description or intent of the pattern matches the description/intent of the form.

Selecting a form pattern via the designer

Right-click the **Design** node of the target form, select **Apply pattern**, and then click the pattern to apply.

Form pattern reference guide

List of classes of top-level form patterns

FORM PATTERN	WHAT IT'S USED FOR
Details Master (two variants)	A form that displays the details of a complex entity
Details Transaction	A form that displays the details of a complex transaction entity and its lines (for example, and order and its lines)
Dialog (six variants)	A form that is used as a dialog to gather a set of information
Drop Dialog (two variants)	A form that is used as a drop dialog to gather a small set of information to provide context for an action
FactBox (two variants)	A Microsoft Dynamics AX 2012 FactBox that displays information about a related record or set of records
List Page	A Dynamics AX 2012 List Page
Lookup (three variants)	A form that is used as a lookup
Simple Details (four variants)	A form that is focused on a single record
Simple List	A form that displays details for a simple entity as a grid that has fewer than 10 fields per record
Simple List & Details (three variants)	A form that displays information about an entity of medium complexity
Table of Contents	A form that displays setup information or loosely related information sets
Task (two variants)	A legacy form pattern that is used to display master or transaction entities
Wizard	A form that displays a set of tab pages to the user to gather information in a predetermined order
Operational Workspace	A form that is used to display an overview of an activity and is meant to be a primary means of navigation
Workspace Panorama Sections (three variants)	A form that is used to show content for a panorama section (via a Form Part Control) in the Operational Workspace

Finding forms that currently use a particular form pattern

For a full list of forms that are currently using a particular form pattern, generate the **Form Patterns** report from within Microsoft Visual Studio. For information on running the report, see Form pattern add-ins. You can filter the report in Excel to find forms that use a particular pattern.

Form pattern visuals and descriptions

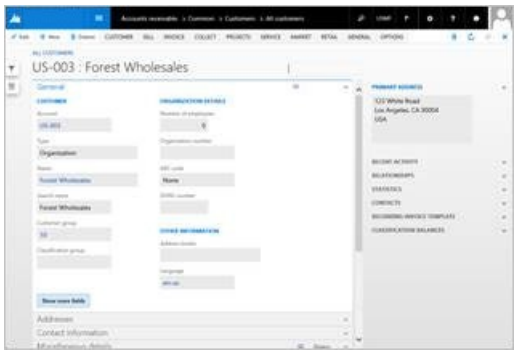
For each form pattern class, information is provided about each variant. This information includes a short

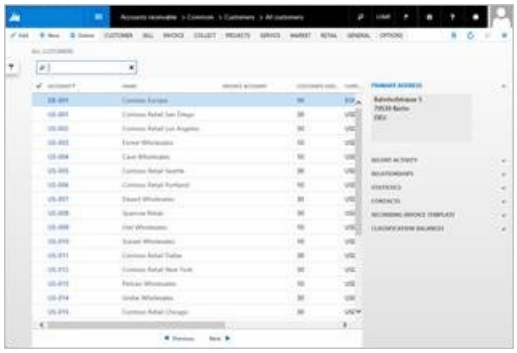
description and an illustration of an example form.

Details Master

Details Master[Default] This form pattern is used to display the details of a complex entity on FastTabs. It includes a grid view and a details view.

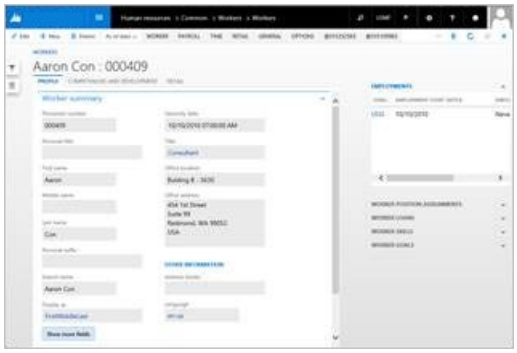
Form: CustTable

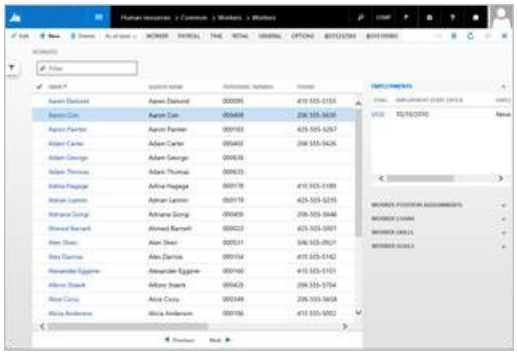




Details Master w/ Standard Tabs Use this Details Master variant when your form has a large number of FastTabs (>15) that can be grouped into categories.

Form: HcmWorker

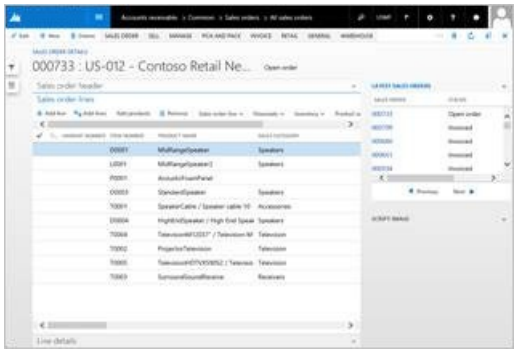


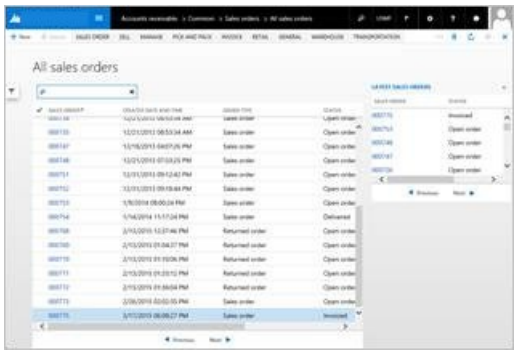


Details Transaction

Details Transaction Use this form patter to show the details of a complex transaction entity and its lines (for example, an order and its lines).

Form: SalesTable





Dialog

Dialog – Basic[Default] This form pattern is used to gather or show a set of information.

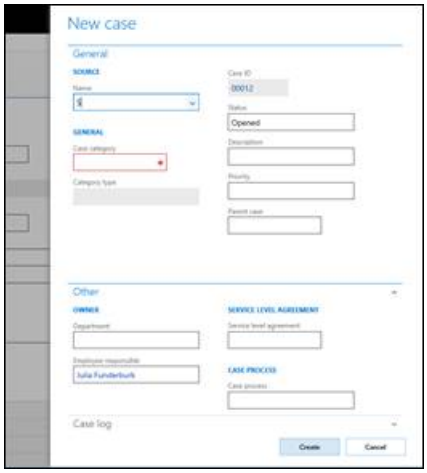
Form: ProjTableCreate

Dialog – Read Only Use this Dialog variant when your Dialog just displays information that can't be edited. It has only a **Close** button.

Form: SalesTablePostings

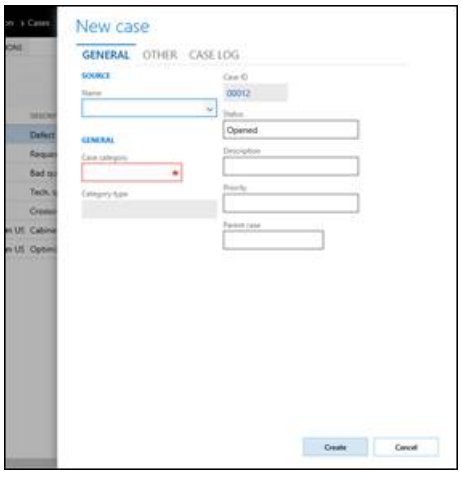
Dialog – FastTabs Use this Dialog variant when your Dialog content is grouped into FastTabs.

None currently in product.



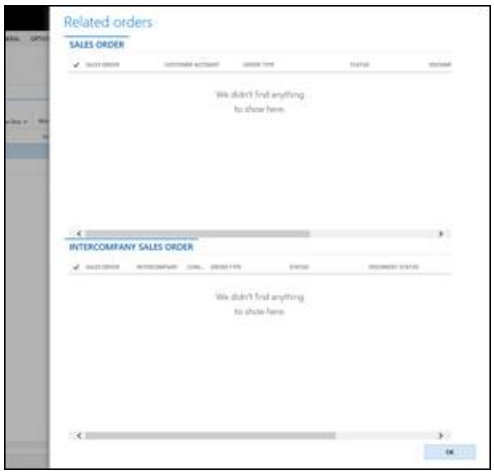
Dialog – Tabs Use this Dialog variant when your Dialog content must be grouped into tabs.

Form: CaseDetailCreate



Dialog – Double Tabs Use this Dialog variant when your Dialog content has two tabs that are stacked on top of each other.

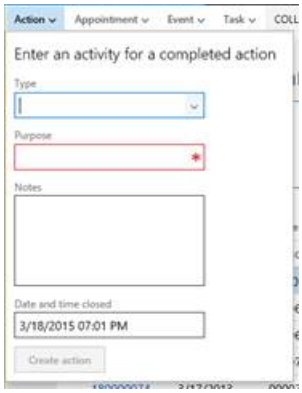
Form: PurchTableReferences



Drop Dialog

Drop Dialog[Default] This form pattern is used to initiate actions when the number of fields is small (less than five).

Form: CustCollectionsNewActivityAction



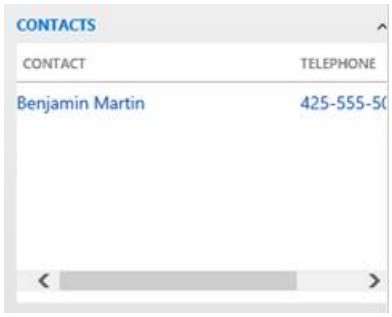
Drop Dialog – Read Only Use this Drop Dialog variant when the fields in the Drop Dialog aren't editable. No **OK/Close** button is modeled.

No example currently exists in the product.

FactBox

FactBox Grid Use this FactBox variant to show a child collection of related information.

Form: ContactsInfoPart



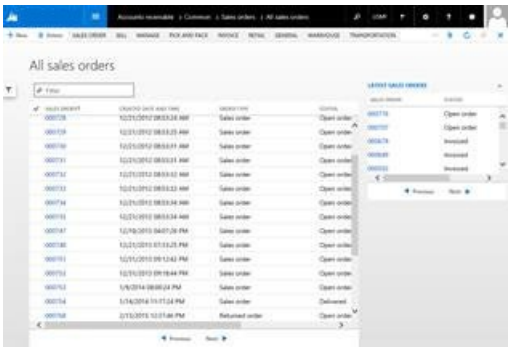
FactBox Card Use this FactBox variant to show a set of related fields.

Form: CustStatisticsStatistics

List Page

List Page The Dynamics AX 2012 list page that is just a grid that is optimized for browsing records and acting on those records.

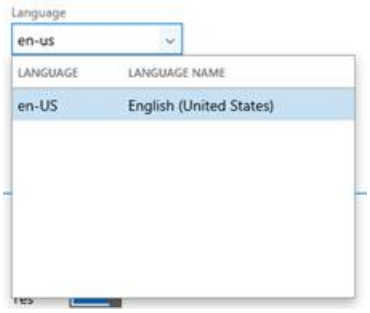
Form: SalesTableListPage



Lookup

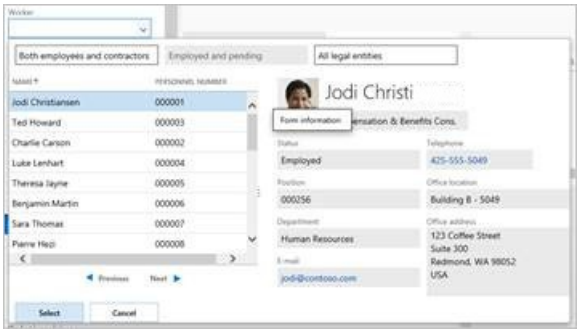
Lookup Basic[Default] This form pattern is used if the lookup form is a grid or tree that has optional filters or buttons at the bottom.

Form: SysLanguageLookup



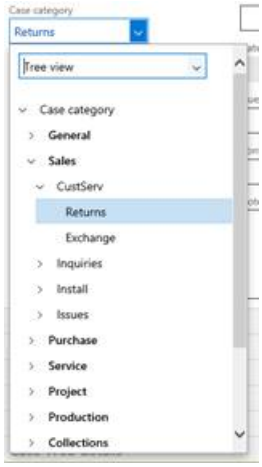
Lookup w/Preview Use this Lookup variant when, in addition to the basic pattern, a preview of the current record is also shown.

Form: HcmWorkerLookup



Lookup w/Tabs Use this Lookup variant when there are multiple views of a lookup (for example, a grid view/tree view or multiple filtered lists).

Form: CaseCategoryLookup



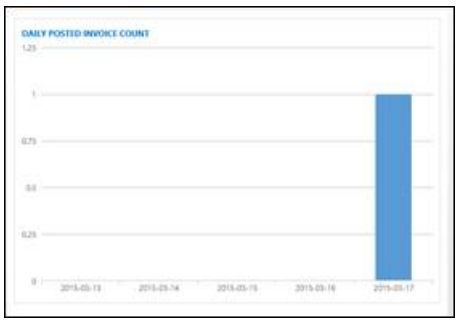
Panorama Section

Form Part Section List Use this form pattern to show a list in a workspace section. This should be modeled as a separate form and rendered in the workspace via a Form Part Control.

Form Part Section List - Double Use this variant when you must also show a secondary list. This secondary list isn't initially visible.

Hub Part Chart Use this variant to show a chart in a workspace section. This should be modeled as a separate form and rendered in the workspace via a Form Part Control.

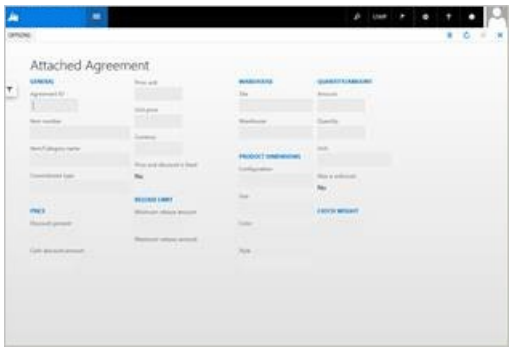
Form: VendInvoiceJourCountChart



Simple Details

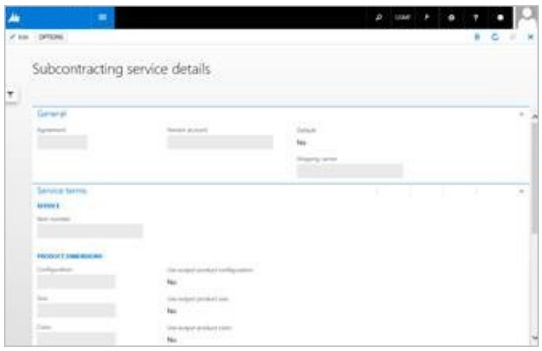
Simple Details w/Toolbar and Fields Use this form pattern to show fields for a single base record.

Form: AgreementLine



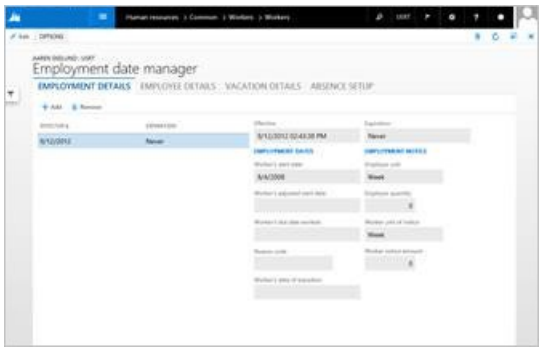
Simple Details w/FastTabs Use this Simple Details variant when the record's information is organized into FastTabs.

Form: PlanActivityServiceDetails



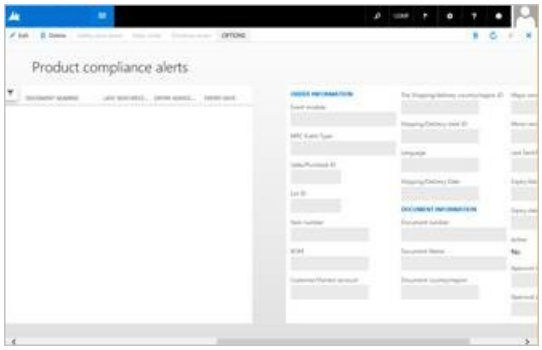
Simple Details w/Standard Tabs Use this Simple Details variant when the record's information is organized into regular tabs.

Form: HcmEmploymentDateManager



Simple Details w/Panorama Use this Simple Details variant to display a record's information in a horizontally scrolling panorama.

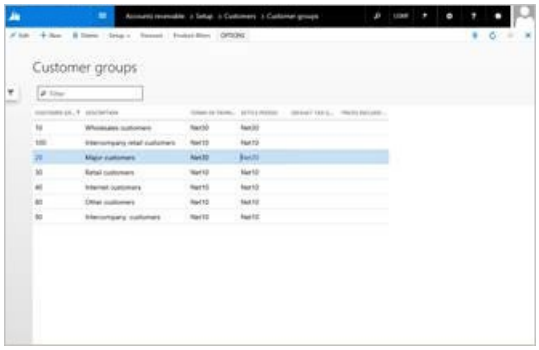
Form: PdsMRCEventTracker



Simple List

Simple List This form pattern is used to maintain data for simple entities.

Form: CustGroup

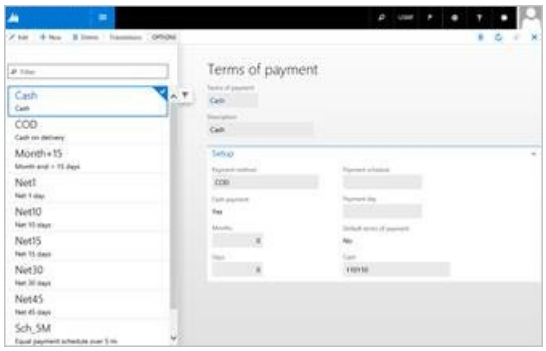


Simple List and Details

Simple List & Details – List Grid[Default] This form pattern is used to maintain data for entities of medium complexity. A list grid that has 2–3 fields in the navigation list is the preferred pattern for this form style in the

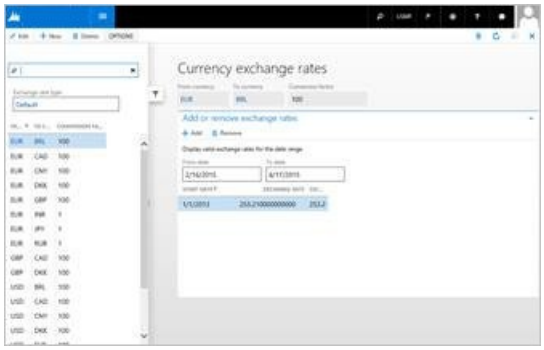
current version.

Form: PaymTerm



Simple List & Details – Tabular Grid Use this Simple List & Details variant if you require more than three fields in the list part of the form.

Form: ExchangeRate



Simple List & Details – Tree Use this Simple List & Details variant if the list part of the form is a tree.

Form: FiscalCalendars

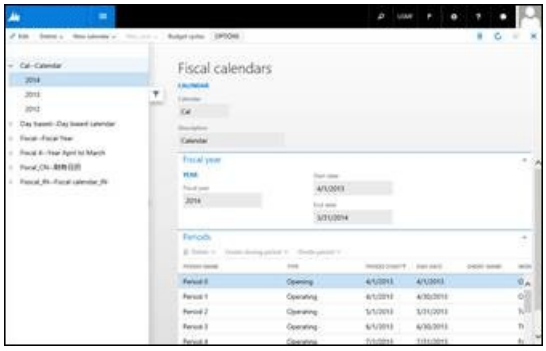
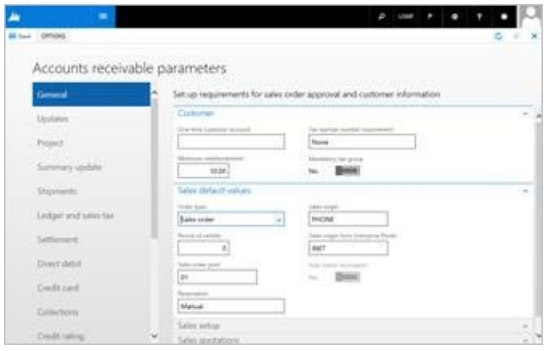


Table of Contents

Table of Contents Use this form pattern to show setup information or loosely related information sets.

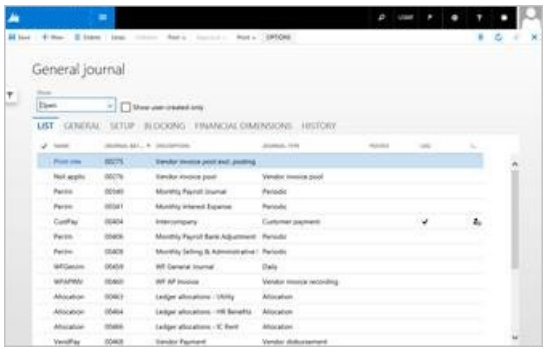
Form: CustParameters



Task

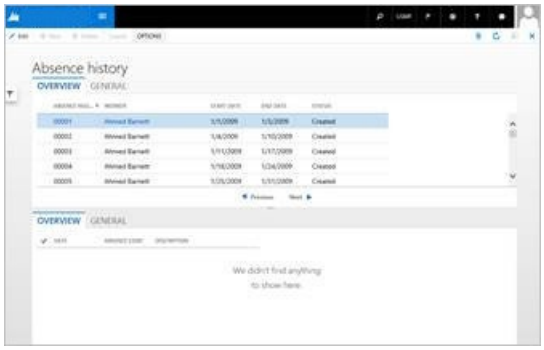
Task Single This legacy form pattern is used to display entities. It should be used only for migration, not for new forms.

Form: LedgerJournalTable



Task Double This legacy form pattern is used to display transaction entities. It should be used only for migration, not for new forms.

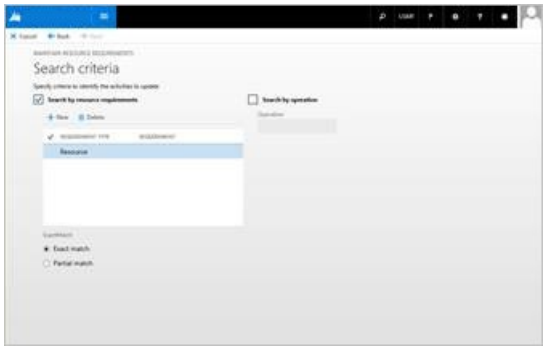
Form: HRMAbsenceTableHistory



Wizard

Wizard This form pattern is used to display a set of page views to the user to gather information in a predetermined order.

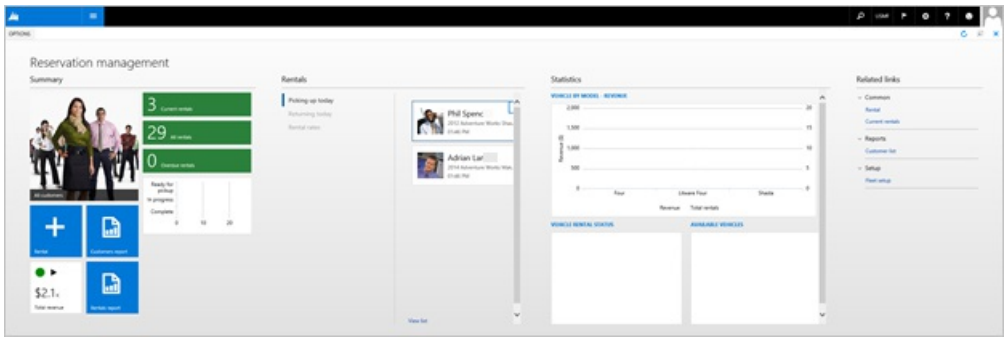
Form: WrkCtrBulkResReqEditWizard



Workspace

Operational Workspace[Default] This is the preferred, performance-enhanced variant of the Workspace pattern.

Form: FmClerkWorkspace



Workspace: This is the old Workspace pattern. It will be removed soon, so don't use it. It is included here only for completeness.

Do not use.

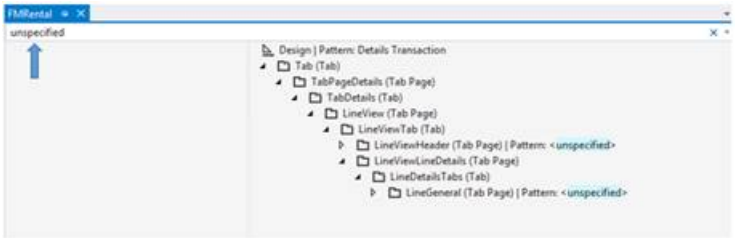
Subpattern reference guide

List of subpattern classes

FORM PATTERN	WHAT IT'S USED FOR
Custom Filters (two variants)	Containers that display QuickFilters and any other modeled custom filters
Fields (five variants)	Containers that primarily display individual fields
Dimension Expression Builder	Containers that include a Dimension Expression Builder control
Dimension Entry Control	Containers that include a Dimension Entry Control
List Panel	Containers that display two lists that users move items between
Nested Simple List and Details	Containers that are used to embed a simpler Simple List and Details form inside a section in a form
Toolbar and Fields	Containers that display actions above a set of fields
Toolbar and List (two variants)	Containers that display actions above 1–2 grids
Workspace-related (eight variants)	Containers that correspond to various sections inside an Operational Workspace

Finding containers that require that a subpattern be applied on a form

When a form is open in the Visual Studio designer, you can easily search for containers that must still have subpatterns applied by searching for "unspecified" in the control search box at the top of the designer (as shown in the following screen shot).



Subpattern visuals and descriptions

For each subpattern class, information is provided about each variant. This information includes a short description and an illustration of an example form.

Custom Filters

Custom Filters Use this form pattern when custom filters are modeled. QuickFilter isn't required.

Form: LedgerJournalTable (TopFields)

Custom and Quick Filters Use this variant when a QuickFilter is required.

Form: CustTable (CustomFilterGroup)

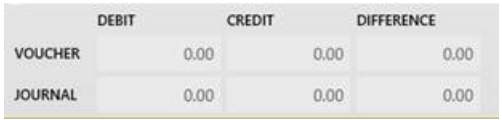
Fields

Fields and Field Groups Use this form pattern to get a responsive layout for containers that contain only fields.

Form: InventLocation (LocationNames)

Tabular Fields Use this form pattern to get a structured layout of fields. It is intended primarily for totals.

Form: LedgerJournalTransVendPaym (Balances)



Fill Text Use this form pattern when a single input control requires full width.

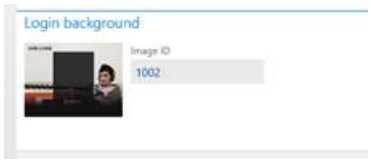
Form: FmRental (Notes)

Horizontal Fields and Button Group Use this form pattern when a field has an inline action.

Form: SalesTable (GroupHeaderAddressHeaderOverview)

Image Preview Use this form pattern for containers that have image controls (and optional related fields).

Form: RetailVisualProfile (Login)



Toolbar and List

Toolbar and List Use this form pattern on containers that have only actions and a grid.

Form: VendTable (TabCommunication)

Toolbar and List – Double Use this Toolbar and List variant when the containers have two grids.

Form: SalesQuickQuote (TabPageExistingItems)

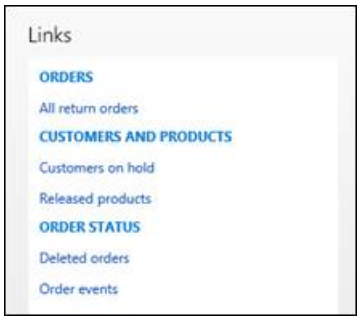
Workspace Related

Section Tiles Use this variant to show a set of tiles/charts in a workspace section. This should be modeled in a tab page on the workspace form. Charts are defined by using Form Part Controls

Form: SalesOrderProcessingWorkspace

Section Related Links Use this variant to show a set of hyperlinks in a workspace section. This should be modeled in a tab page on the workspace form.

Form: SalesOrderProcessingWorkspace



Section Tabbed List Use this variant when multiple list variants must be included. Only one is shown at a time.

Section Stacked Chart Use this variant when you must include up to two charts in an Operational Workspace.

Section PowerBI Use this variant when a Power BI section must be included.

Workspace Page Filter Group Use this form pattern to add a single filter to your workspace.

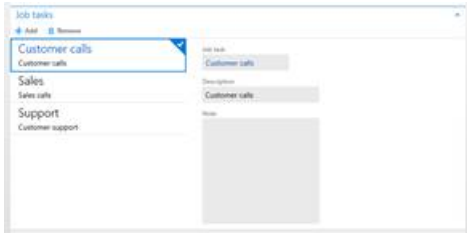
Filters and Toolbar – Stacked Use this subpattern in the Form Part Section List pattern, so that actions appear below filters.

Filters and Toolbar – Inline Use this subpattern in the Form Part Section List pattern, so that filters and actions appear on the same line.

Other

Nested Simple List & Details Use this form pattern to embed a simpler Simple List & Details form inside a tab or group.

Form: HcmJob (TaskTabPage)



List Panel Use this form pattern when users must move items back and forth between two lists.

Form: CLIControls_ListPanel (FormTabPageControl1)

Toolbar and Fields Use this form pattern on containers that have only actions and fields

Form: HcmPosition (WorkerAssignmentTabPage)

Dimension Entry Control Use this form pattern on tab pages that have only a Dimension Entry Control.

Form: CustTable (TabFinancialDimensions)

Dimension Expression Builder Use this form pattern on containers that include a Dimension Expression Builder control.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Form styles and patterns

2/18/2021 • 13 minutes to read • Edit Online

This topic describes the concept of form patterns and discusses the process for applying and removing patterns. A list of frequent questions are also answered in this topic.

Dynamics AX 2012: Form styles and templates

In Microsoft Dynamics AX 2012, several form styles were introduced and formalized. Primary data types are represented by the List Page and Details Form styles. Secondary data types are represented by the Simple List and Details Form and Simple List Form styles. In addition to these core form types, other form styles exist for supporting forms, such as Table of Contents for settings and Drop Dialog for dialog forms, and Lookup for lookup forms. Other less formal form patterns, such as Wizard, also exist. Developers who wanted to build a new form of a specific style in Dynamics AX 2012 often used the corresponding template form as a starting point. After they included form content and made any modifications that were required, developers could then run the Form Style Checker add-in to validate their form in terms of structure and property values against that form style's template form.

Finance and Operations: Form patterns

Form patterns (a new concept that is the evolution of the Dynamics AX 2012 form templates, style, and Form Style Checker) are now an integrated part of the form development experience. These patterns provide form structure, based on a particular style (including required and optional controls), and also provide many default control properties. In addition to top-level form patterns, subpatterns can be applied to container controls, and that provide guidance and consistency for subcontent on a form (for example, on a FastTab). Patterns have made form development easier by providing a guided experience for applying patterns to forms to guarantee that they are correct and consistent. Patterns help validate form and control structures, and also the use of controls in some places. Patterns also help guarantee that each new form that a user encounters is immediately recognizable in appearance and function. Form patterns can provide many default control properties, and these also contribute to a more guided development experience. Because patterns provide many default layout properties, they help guarantee that forms have a responsive layout. Finally, patterns also help guarantee better compatibility with upgrades. Many of the existing form styles and templates from Dynamics AX 2012 continue to be supported. However, legacy form styles and templates that aren't supported have a migration path. Because the foundational elements are built based on those legacy form styles and patterns, the transition is as easy as possible.

Applying patterns

Applying a pattern is a straightforward process that can modify properties on multiple containers and controls on a form. Here is the standard flow for applying patterns:

1. Acquire the target.
2. Determine the pattern.
3. Apply the pattern.
4. Handle errors.

Acquire the target

First, you must identify a target form and add it to your project. The **Form Patterns** report that is generated by using a Microsoft Visual Studio add-in can help you find forms that don't have patterns. For information on

running the report, see Form pattern add-ins. Open the report file in Microsoft Excel, and filter to a form that has no pattern. Then, in Visual Studio, open Application Explorer, and find the form. Right-click the form, and then select **Add to project**. When you open the form in the designer, it should have the **Pattern: <unselected>** designation on the design node.

Determine the pattern

Decide which pattern to apply. The available patterns include those that are based on Dynamics AX 2012 form templates, and also patterns that are designed for Finance and Operations scenarios. If you require help selecting a pattern, see the Selecting a Pattern topic. For more detailed information about specific patterns, see the individual pattern guideline documents. For more information about applying a pattern, see Select a form pattern.

Apply the pattern

You can apply a pattern in three ways:

- Using metadata
- Using visuals
- Using the designer For more information about applying a pattern, see Select a form pattern.

Handle errors

Information about the pattern appears on the **Pattern** tab. To learn about the pattern structure, click the control names on the **Pattern** tab to navigate the pattern structure. When you save or build the form, the pattern errors appear in the error list in Visual Studio.

- Double-click an error to go to the control that the error was reported for, if the control exists.
- If a control is missing, follow one of these steps:
  - If the control already exists on the form but is in a different place, move the control to the correct place, as indicated by the pattern.
  - If the control doesn't exist, create the control.

Subpatterns

After you apply a pattern to a form, you might have to apply subpatterns to the form's container controls. The process is similar to applying a pattern to a form: acquire the target, determine the subpattern, apply the subpattern, and handle any errors. To find container controls on the form that require subpatterns, search for "unspecified" in the search box at the top of the form designer in Visual Studio. These controls should have the **Pattern: <select>** designation in the form designer. For each container, you should examine the contents and select the most appropriate subpattern. Like form patterns, the available subpatterns cover common container layouts from Dynamics AX 2012 but also include several new subpatterns. If you require help selecting a subpattern, see Selecting a Pattern. For more detailed information about specific subpatterns, see the individual subpattern guideline documents. After you've selected a subpattern, right-click the control in the form designer, select **Apply pattern**, and then click the subpattern to apply.

Frequently asked questions

What does applying a pattern do?

By applying a pattern, you can change multiple properties on multiple nodes in one quick action. Therefore, it's important that you understand what is happening.

Where do I find information about a pattern?

The **Patterns Information** panel (the **Patterns** tab under the form designer), the error list (**View** > **Error List**), and the form statistics add-in (right-click the form name in the form designer, and then select **Add-ins** > **Form Statistics**) all provide valuable information when you're trying to apply patterns. When you apply patterns, it's

important that you have patience, read, and proceed at a steady pace. Pattern guideline documents are also available for each pattern and subpattern. These documents contain a lot of additional information, such as information about when to use a particular pattern, what is included in the pattern, and UX guidelines to beware of when you use a pattern.

What do I do if I make a mistake when I am applying a pattern?

If you make a mistake, there are several actions that you can take:

- **Undo** – The Undo command (Ctrl+Z) is usually available for all actions, even for applying and removing a pattern.
- **Remove the pattern** – If you applied the wrong pattern, remove the pattern by right-clicking and then selecting **Remove pattern**. Note that properties are applied to nodes after a pattern is successfully applied without errors. Therefore, even after a pattern is removed, any properties that were changed by the pattern will still be set to the new values.
- **Revert** – When all else fails, take advantage of the source control system, and revert the changes that were made to a form.

Why are some properties hidden when the pattern has been applied?

Patterns enforce properties after the pattern structure has been successfully met. A property that is controlled by the pattern is hidden from developers because developers don't have to worry about that detail. This makes the development experience cleaner by reducing the "noise" in the Property Pane from properties whose values have been set by the patter and thus cannot be modified by a developer while the pattern is applied. Developers who are interested in the properties that a pattern is setting can remove the pattern. All the property values will then be visible on all the controls that are covered by the pattern.

How do I identify the set of forms that I should be doing more pattern work on?

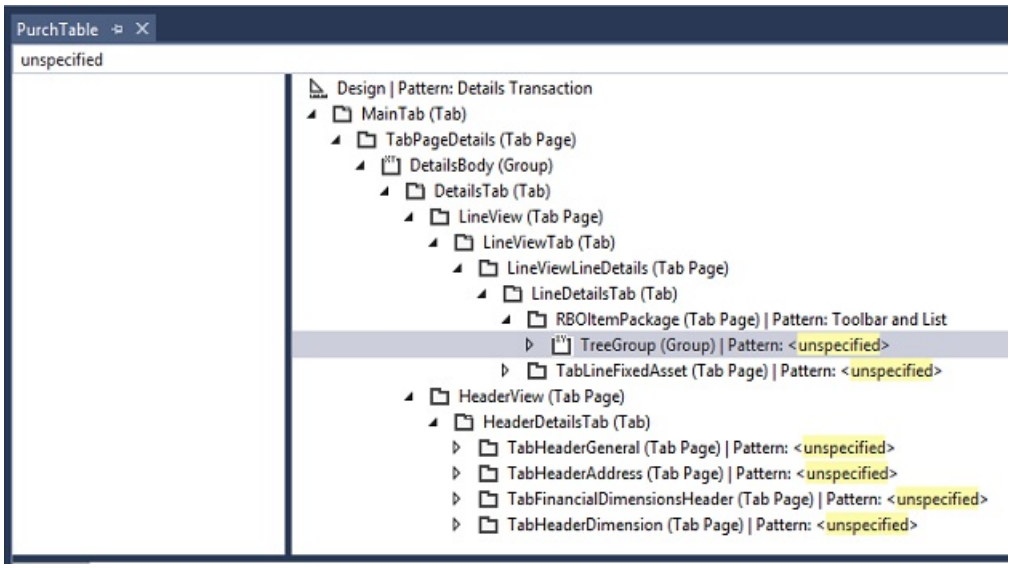
To identify the set of forms that still have remaining patterns work, you should generate and consult the **Form Patterns** report. For information on running the report, see Form pattern add-ins.

- **Filter the "Pattern" column to only show "(Blanks)"** - This will show all the forms with no form pattern applied (no pattern specified on Form.Design).
- **Filter the "Unspecific count" column to only those values "greater than 0"** - This will show all forms where a pattern is expected on either Form.Design or a container control somewhere on the form. You can combine this filter with the previously mentioned filter on the "Pattern" column to show only forms with subpattern work remaining.

Note that there are no remaining patterns work left in your models if there are no rows after the filters described previously are applied. If you want to make sure that all your forms are fully covered by patterns (meaning no unspecified nodes and no Custom nodes), filter the report down to those rows that have "Percent covered controls" less than 100 percent.

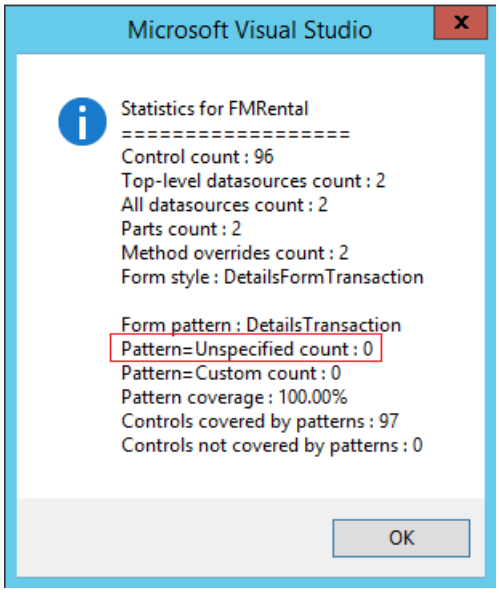
How do I find places in a form where a pattern can be applied?

To find container controls in the form that must still have a pattern applied, search for “unspecified” in the form designer. This search will highlight all the nodes in the form that have the **Pattern: <unspecified>** designation. You can then examine each container individually to apply the most appropriate subpattern.



How do I check whether more pattern work must still be done on a form?

To determine whether more pattern work must be done on an open form, right-click the form in the designer, and then select **Addins** > **Form statistics**. If the **Pattern=Unspecified** count is more than 0 (zero), the form still has containers that must have a pattern applied. Ideally, every form should also have a **Pattern=Custom** count of 0 (zero) and a **Pattern coverage** value of 100 percent. These values indicate that the form is fully covered by patterns.



Why isn't static text allowed inside the Fields and Field Groups subpattern? Isn't static text allowed in forms any longer? What do I do with the static text that the Fields and Field Groups subpattern doesn't accept? How can I show user Help instead of static text?

Static text in a form is often used as a highly visible mechanism for providing form help. Many patterns, such as the Toolbar and List subpattern and the Fields and Field Groups subpattern, don't allow static text to be placed directly on a form. Although user assistance is a good idea, it can often be provided in other ways. At a high level, the goal is to provide better labels and a more understandable user interface, so that explicit Help content isn't required. However, when explicit Help content is required, it should be provided as field HelpText or form-level Help content. If the static text explains the meaning of an image in a grid, consider using a tooltip to provide assistance when the user hovers over (for a mouse) or touches and holds (for touch) that image. Follow these steps to deal with static text on the form:

1. Determine a replacement for the static text on the form:

    - Consider whether the user information that is provided via static text is still required.
    - Consider whether field labels can be made more descriptive.

- Consider whether field HelpText would be appropriate.
- Consider whether form-level Help content would be appropriate.

2. After a replacement has been determined and implemented, remove the static text, and then apply the pattern.

3. If a replacement can't be implemented yet, leave the static text, and continue to use **Pattern:Custom**.

Why can't I change WidthMode on fields in the Fields and Field Groups subpattern? Why can't I specify a manual width on fields in the Fields and Field Groups subpattern?

Both the HeightMode and WidthMode properties are not available on controls inside the Fields and Field Groups subpattern because that subpattern intentionally sets the HeightMode and WidthMode properties to be SizeToContent on input controls. An input control with a SizeToContent width is sized based on a mapping of DisplayLength to one of four pre-defined discrete sizes (extra small, small, medium, or large). This discretization of input control widths was done in an attempt to provide a fresh, clean user experience that is simple and consistent (minimizing the jagged edges caused by arbitrarily wide fields). The discrete sizes were also chosen in a way that allows these fields to be combined together to form organized and visually appealing sets of fields (since the larger field sizes are multiples of the smallest field size in terms of width).

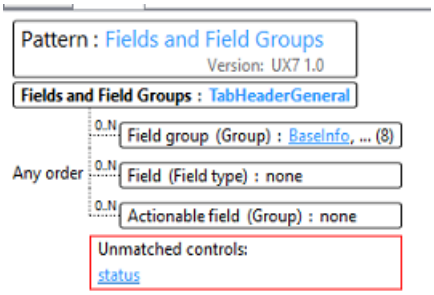
In general, there are two other width options available for fields that are not currently allowed in the Fields and Field Groups subpattern:

- **SizeToAvailable** - We cannot allow SizeToAvailable width controls inside of this subpattern because this sizing option does not work with the ColumnsMode=Fill layout algorithm (nor does it make sense when trying to decide how to best lay out controls into columns)
- **Manual** - Controls with manual widths should be rare; the vast majority of controls should be SizeToAvailable or SizeToContent. Manual-width controls introduce inconsistency into the discrete set of field widths and column widths, especially given the fact that these controls do not adapt their size based on the user-selected density. Manual-width controls may also present difficulties in ensuring a responsive design, as they may be set to sizes that are larger than can be accommodated by a particular viewport. In an effort to preserve the desired responsive field layout and clean interface and since manually-sized controls should be extremely rare, we have opted to not allow them in the Fields and Field Groups subpattern.

Your current options for scenarios where a manually sized control is needed include using a Custom pattern (which is reasonable given the field requires a "custom" size) as well as potentially using the Fill Text subpattern for a wider field (which allows a single full-width field per container, though we plan to extend the Fill Text subpattern to allow an arbitrary number of full-width fields).

Why do I have "unmatched" groups when I try to apply the Fields and Field Groups subpattern?

Groups and controls appear as "unmatched" in the **Pattern Information** panel if they, or any controls inside them, aren't allowed by the pattern.



There are two typical reasons why groups appear as "unmatched" in the Fields and Field Groups subpattern:

- There is more than one level of group depth. **Solution:** Refactor the groups so that they have only one level of depth inside the container that you're trying to apply the Fields and Field Groups subpattern on.
- There is an image or static text inside the group. **Solution:** Remove or relocate that control, if you can.

What do I do if my form is close to a form pattern but deviates in some way that makes it Custom?

Some forms are structurally close to a defined form pattern, but because some aspects don't fit the pattern, a Custom pattern is applied to the form. In this case, you might still be able to get some benefits of the form pattern (for example, the layout properties can be set automatically) by following these steps.

1. Modify your form so that it fully fits the pattern (for example, move or temporarily remove any controls that deviate from the pattern).
2. Apply the desired pattern.
3. Save the form, so that the property values are set by the pattern.
4. Remove the pattern.
5. Move the controls that deviate from the pattern back to their original location.

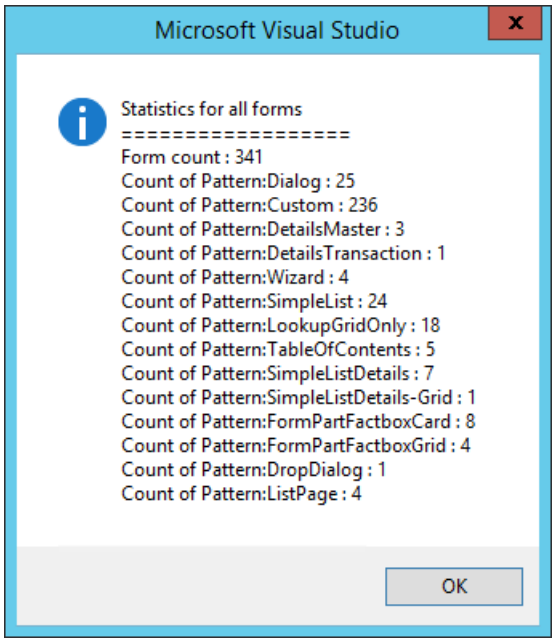
Visual Studio add-ins that support form patterns

2/18/2021 • 2 minutes to read • Edit Online

The tools for Visual Studio include a number of add-ins that support pattern usage.

Form statistics add-in

The **Form statistics** add-in provides a summary of the pattern usage for forms. When you access the **Form statistics** add-in from the **Dynamics 365** menu, it displays statistics for all forms. When you access the add-in from the shortcut menu for a form that is open in the form designer, it displays statistics for that form only.



Forms Pattern report

The **Form Patterns** report provides pattern information about every form, including whether the form uses a top-level form pattern, is a custom form, or is not specifying a form pattern. To generate the **Form Patterns** report, start Microsoft Visual Studio, click the **DYNAMICS 365** menu, expand **Add-ins**, and then click **Run the form patterns report**. The process will take several seconds. After the report has been generated, a dialog box will provide the location of the report. Browse to the specified location, and open the file in Microsoft Excel. You can then filter the report down to the models that interest you.

For more information about Visual Studio add-ins, see Tools add-ins for Visual Studio.

General form guidelines

2/18/2021 • 13 minutes to read • Edit Online

This topic contains the guidelines that apply to all forms, regardless of form pattern. This checklist must be used in addition to any pattern-specific guidelines.

Verification checklist

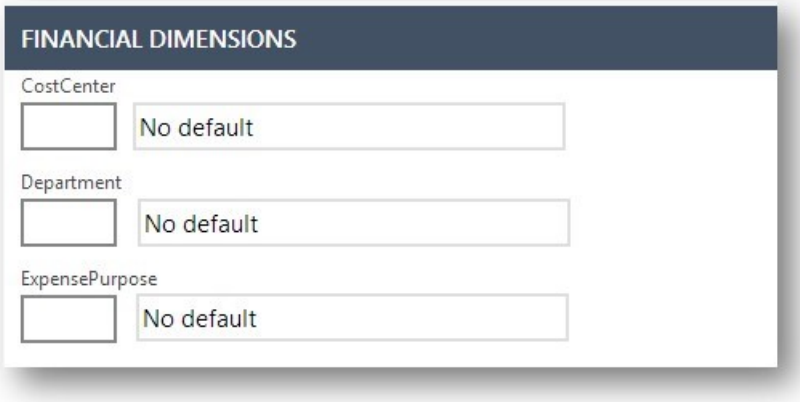
The verification checklist shows the steps for manually verifying that the form complies with the UX guidelines. This checklist doesn't include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps.

Standard form guidelines

Specific form patterns and subpatterns might have exceptions to these guidelines.

- The form layout is responsive when the browser is resized or the app is run on different device sizes. In other words, all the fields should be accessible to the user either by reflowing the layout or by scrolling to the fields.
- Make sure that the form's default View/Edit state is correct. By default, forms are in View mode. If a form should always be in Edit mode, you must explicitly set **Form.Design.ViewEditMode=Edit**.
  - The View/Edit state should be appropriate to the state of the entity. For example, if the state of the entity is **Posted**, and the form can't be edited, the default state should be View mode (and Edit mode should be disabled).
- **Form captions**:
  - Avoid setting the form caption programmatically. Instead, consider setting the **TitleDataSource** property on the form design node to enable the framework to provide the caption dynamically.
  - If you can't avoid setting the form caption programmatically, make sure that it's short (no more than 30 characters). This guideline exists because a large font size is used for the form in some form types.
    - **Exceptions:** Custom Lookups, FactBoxes
  - Form captions should provide the user with the context of the "type" of entity. The font size and the position of the form caption will vary, depending on the type of form.
  - Don't use the form caption to convey contextual information such as the parent record or other status information.
- All labels in the form are in sentence case. The framework guarantees consistency by putting some elements, such as Group labels, FactBox captions, Action Pane tab labels, and Button Group labels, in ALL CAPS. These strings should still be added in sentence case, but the framework will display them in all caps.
- All labels in the form are spelled correctly and use proper grammar.
- Avoid overriding the formatting alignment for extended data types (EDTs).
- When custom filters are used, no more than five should be specified. Instead, consider pre-populating the filter pane with fields.
- Occasionally, a decision must be made to determine whether a control should be temporarily disabled or hidden. The following information can help you make this determination:

    - A control should be temporarily disabled if the specific conditions must be met before the control can be enabled, and the user must first perform some action to meet those conditions.
    - A control should be hidden from the user if there is nothing that the user can do to enable or edit the control.
        - **Exception:** The control is used to convey status to the user.
- No UX guidelines are violated when security is applied for the various roles that have access to this form.
    - **Example:** For role A, field A **is not** required, but for role B, field A **is** required.
- No UX guidelines are violated when country/region codes are applied.
- Two fields can share a single label. Group the fields into a group, and set the **FrameType** property of the group to **GroupedFieldsLabel**.



Other form guidelines

- Use a StaticText control instead of StringEdit for multi-line read-only text. StringEdit controls are semantically incorrect for informational text, because they can never be edited. Additionally, StringEdit controls typically have a border and different layout characteristics than StaticText controls, and these differences negatively affect the user experience.
- Controls that are always read-only but data-bound should be marked as **ViewEdit**=**View**.
- (Dialogs and Drop Dialogs only) The button that is chosen as the default button should be the safest, most secure response to the task that the user is performing. For example, the default button should correspond to the main instruction of a Dialog or Drop Dialog.
    - If safety and security aren't factors, the button that is most likely to be clicked or that is most convenient for the user should be chosen as the default button.
    - **Exception:** Don't select a destructive response as the default button, unless there is an easy, obvious way to undo the command.

Field state guidelines

It's important that the state of a field be set correctly. The state of the field communicates key information about what the user can do and how to do it. The following table outlines key guidelines about when to use the various field states. **Note:** In the past, field states weren't used correctly. People interchangeably used **Enabled**=**No** and **Readonly**=**Yes**. Note the semantic differences between the two states:

- **Enabled**=**No** – The data **is not** valid.
- **Readonly**=**Yes** – The data **is** valid.

STATE	DESCRIPTION
Enabled=No	Disabled fields tell the user that the field isn't valid in the current state of the entity. Use the **Enabled** property to disable a field and prevent data input. A disabled field (**Enabled**=**No**) has these characteristics: • The field is visually presented so that it looks unavailable. The text will be gray to indicate a "not valid" appearance. • The field's value is unimportant and won't be sent to the server for processing. • The field is skipped in the tab sequence. A disabled field might become enabled by actions that the user takes in the form. If the user can never enable a field, consider hiding the field instead.
Readonly=Yes	For read-only fields, the data is valid in the current context but can't be edited. The field isn't skipped in the tab sequence. If the value can never be edited, consider using **ViewEditMode**=**View** for the field.
ViewEditMode=View	Use this state when the value in a field can never be edited by the user and is for informational purposes only.
Secured	Fields are typically hidden if they are secured. In grids, each row can define different levels of security for each column. Therefore, for cells where the user has no access, a padlock appears in the field. This isn't an application-controlled ability and occurs automatically.
Not available	Similar to secured fields, data can appear in a grid where each row has a different set of columns. In this case, cells that aren't applicable to a row display a "not" symbol. This isn't an application-controlled ability and occurs automatically.

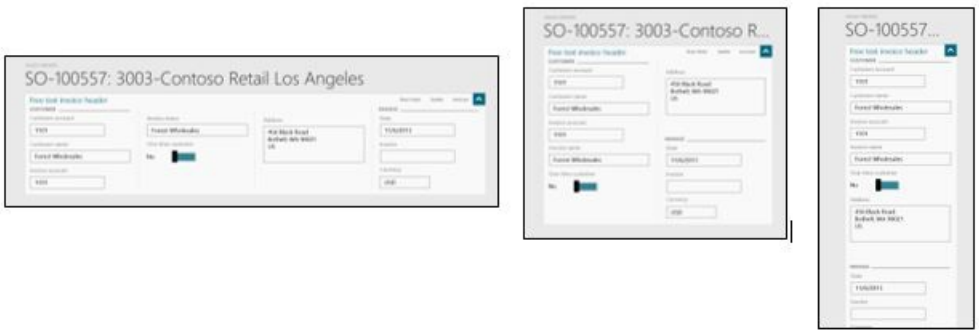
Mandatory field guidelines

Mandatory fields are fields that the user must supply values for to guarantee database referential integrity and business logic integrity. By marking fields as mandatory, you provide an important indicator about what the user must do.

- All mandatory fields are marked.
- Mandatory fields should be set on metadata concepts in this order:
  1. Table
  2. Data source
  3. Form field
- Mandatory fields should be set programmatically, based on the state of the record. For example, a field that is required in order to post an entity should be marked as mandatory.
  - If the user receives a validation message about an empty field, but that field isn't marked as mandatory, the user experience is negatively affected.

FastTabs guidelines

- The fields in groups should flow across the FastTab.



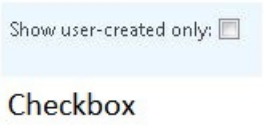
- The content of the first FastTab should be fully visible without scrolling. FastTabs should never horizontally scroll when the fields are displayed.
- The first FastTab should contain the most important fields for this entity (the fields that will be edited most often).
- FastTabs should display summary information.
  - **Exceptions:**
    - FastTabs that contain only a grid aren't expected to display summary fields.
    - Dimensions FastTab pages can't display summary fields, because this functionality isn't currently supported.
- If the FastTab contains a grid, it should follow the Toolbar and List subpattern guidelines.

Radio button guidelines

- Follow all standard Microsoft guidelines for radio buttons. Specifically, observe these guidelines:
  - The radio button control is used to select one option from a set of mutually exclusive choices.
  - There are between two and seven choices. If there are more than seven choices, use a combo box instead.
  - If none of the options is a valid choice, there is another option to reflect this situation, such as **None** or **Does not apply**.
  - Selection of a radio button doesn't perform commands, display other windows, such as a dialog box, or dynamically show/hide other controls that are related to the selection.
  - One radio button option is always selected by default. (However, see the exceptions at the MSDN guidelines link.)
  - The default option is the safest (that is, the option prevents loss of data or system access) and most secure and private option that is available. Alternatively, the default option is the most likely and convenient option.
  - Every radio button has a label.
- If subordinate controls are required for the selection of a radio button, follow these guidelines:
  - The subordinate controls are visible and appear to the right of or below the radio button.
  - Using the subordinate control selects the radio button.
  - The subordinate controls aren't nested radio buttons that have other radio buttons or check boxes.
- Options are listed in a logical order:
  - From the most likely to be selected to the least likely to be selected
  - From the simplest operation to the most complex operation
  - From the least risk to the most risk

Check box and toggle guidelines

Toggle buttons are typically used instead of traditional check boxes.



Checkbox

Toggle

- Follow all standard Microsoft guidelines for check boxes. Specifically, observe these guidelines:
  - By default, use toggle buttons instead of check boxes in forms. The label must follow the Microsoft guidelines for check box labels.
    - **Exceptions:**
      - Use a check box when a large number of related options must be set in a group.
      - Use a check box inside Custom Filter Group subpatterns.
      - Groups where a check box is used on the frame to collect related fields. *This exception is currently under review to add clarity.*
  - Don't use check boxes/toggle buttons as a progress indicator.
  - Don't use check boxes to initiate a command. However, a message box can be shown to the user to refine or clarify the choice.
  - Write the label so that it describes the selected state of the check box. The meaning of the cleared state must be the unambiguous opposite of the selected state.
  - For a group of check boxes, use similar phrasing, and try to keep the length of all labels about the same.
  - Use positive phrasing. Don't phrase a label so that selection of the check box means that an action is **not** performed.

Selection panel guidelines

- Occasionally, you'll require a discrete list of options that the user can select from, where each option presents the same interface. Don't use radio buttons to present the list of options. Instead, use a combo box control inside a Custom Filter Group .
- If the selection list is common to a set of items, the combo box should appear above all the items that are affected.

General grid guidelines

- Grids should be sorted on the first column and in ascending order, unless the scenario requires a different sort order:
  - The identifier (ID) field for documents (Sales Order ID, Purchase Order ID, and so on)
  - The name/description field for entities (Vendor name, Customer name, and so on)
  - Depending on the scenario, sorting can be on another column that makes business sense, such as the sequence number or date.
- **Editable grids**:
  - You must show and set the mandatory fields in the grid. It's not acceptable to alert the user to a missing mandatory field only when the record is saved.
  - All editable grids must have a **New/Delete** or **Add/Remove** button in a toolbar above the grid, or in the global Action Pane.
- Order the columns so that the most important columns are on the left. Amount columns are most usable for a user when they are placed in the rightmost position in the grid.
- Image columns can be used to convey state information for an entity or process, such as the workflow status.
- In general, the image column should be placed on the left side of the grid, and there should not be a column label. The meaning of the image should be indicated by the tooltip.
- Numeric columns should be right-aligned. All other columns should be left-aligned.
- If the form can be opened in View mode and can be used to create records in Edit Mode, the property of the

root data source that is connected to the grid should be **Insert If Empty**=**No**. This setting will prevent situations where a user sees and selects a blank record when a read-only form is opened.

Entity status field guidelines

- The entity status must appear in the upper-right of the form, to the right of the title fields. The Details form patterns provide an optional Group where status fields can be defined.

Standard Action Pane guidelines

- The Standard Action Pane is at the top of the form and follows the standard Action Pane guidelines.
  - **Exceptions:** Dialog, Drop Dialog, Lookups
- For application-added actions, labels for similar actions should be used consistently across all Action Panes.
- If the name of the button doesn't adequately explain the action, a supplemental description should be shown in the tooltip.
- Actions on an Action Pane should apply to the whole entity. Never put actions that apply only to portions of the entity on the Action Pane. Instead, use local toolbars that are near the objects that will be acted on.
- There should be no more than 10 tabs on a Standard Action Pane.
  - There should be between one and eight actions per group.
  - The first tab is the home tab. It should have the same name as the entity and should be singular.
- **Activity tabs**:
  - Actions that are related to a specific activity should be grouped into an appropriately named Activity tab.
  - These tabs should be given the names of activities that the user will understand. The names should consist of action verbs.
  - Remove system-defined actions from Activity tabs.
  - **New** and **Maintain** groups group actions, such as **New**, **Delete**, **Edit**, and **Save**, that are related to these primary actions. The exception is **New** actions that are related to secondary types within the entity.
  - Remove duplicate **Refresh**, **Attachments**, **Export to Excel**, **Restore**, **OK**, and **Done** buttons.
- **General tab**:
  - If there are common, infrequent actions that aren't related to a specific activity, they should appear on the last tab, which should be named **General**.
  - Commands that appear on the **General** tab should not be repeated on other tabs.

Buttons on canvas guidelines

- Buttons in the content area of a form that are related to the form, not to a specific field, should be placed on the standard Action Pane or in a Toolbar.

Button image guidelines

- Buttons that require images should use symbols for their images.
- If both an image and text are shown on a button, the image must be to the left of the text. No other configurations are supported.
- **Standard Action Pane**:
  - Buttons that replace the system **New/Delete** buttons should use the New/Delete symbols.
  - Common actions should have **ButtonDisplay**=**Auto**, unless they have symbols that are used by other system buttons. In that case, they should be **TextOnly**. Consult UX if there are other common actions that you believe should have symbols assigned.

    - Other uncommon actions should be **TextOnly**.
    - Images aren't supported on Action Pane tabs.
- **Toolbars**:
    - **Add/Remove** (if applicable) should be the first buttons, should use the Add/Remove symbols, and should have both images and text (**ButtonDisplay**=**Auto**).
    - Common actions should be moved just to the right of the **Add/Remove** buttons. These actions can also have both images and text (**ButtonDisplay**=**Auto**). If no appropriate image exists, these can be **TextOnly**.
    - Subsequent uncommon actions should be **TextOnly**.
- **Inside MenuButtons**:
    - Images aren't supported on buttons in MenuButtons.

Appendix

Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

- **What types of fields aren't supported as summary fields on Fast Tabs?**
    - For performance reasons, we don't currently support reference groups and display methods. Additionally, unbound fields can't be used as summary fields.

Details Master form pattern

2/18/2021 • 4 minutes to read • Edit Online

This topic provides information about the Details Master form pattern. A details form is the primary method for entering data.

Usage

A details form is the primary method for entering data. These forms let the user view, edit, and act upon data. All content on these form types is structured into FastTabs that can be expanded and collapsed, so that multiple FastTabs can be open at the same time. The FastTabs can contain fields or a grid, and each FastTab can have a local toolbar. Two patterns are described in this document:

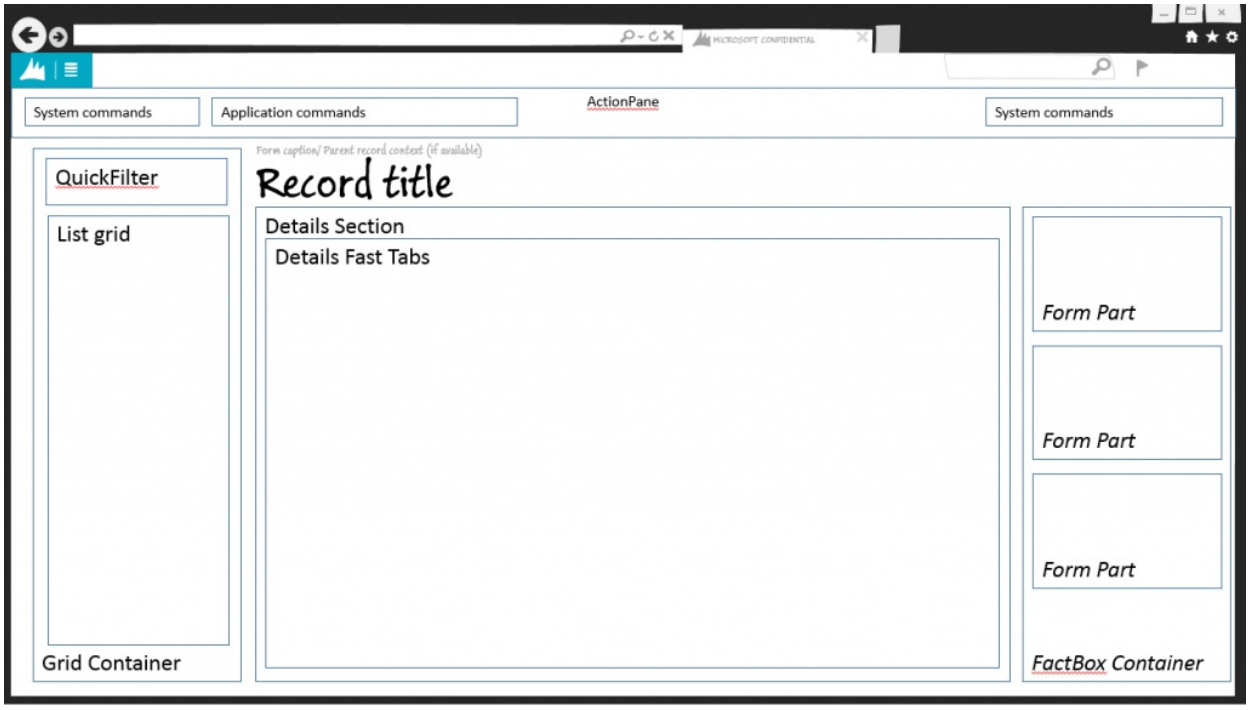
- **Detail Master** – This is the basic Detail Master pattern. This is the pattern that you should use by default.
- **Detail Master w/ Tabs** – You should use this pattern when an entity requires many FastTabs (more than 15) that can be grouped into categories.

In both cases, the grid view is structured the same.

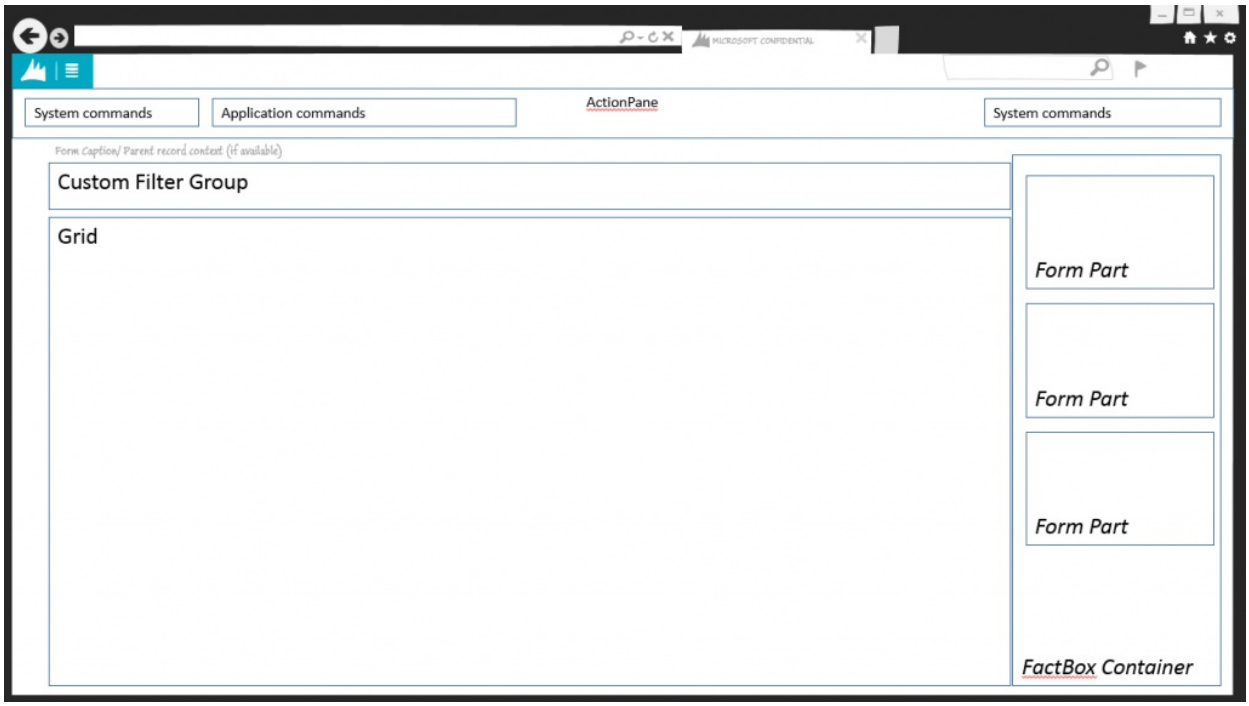
Wireframe

Details Master

Details view

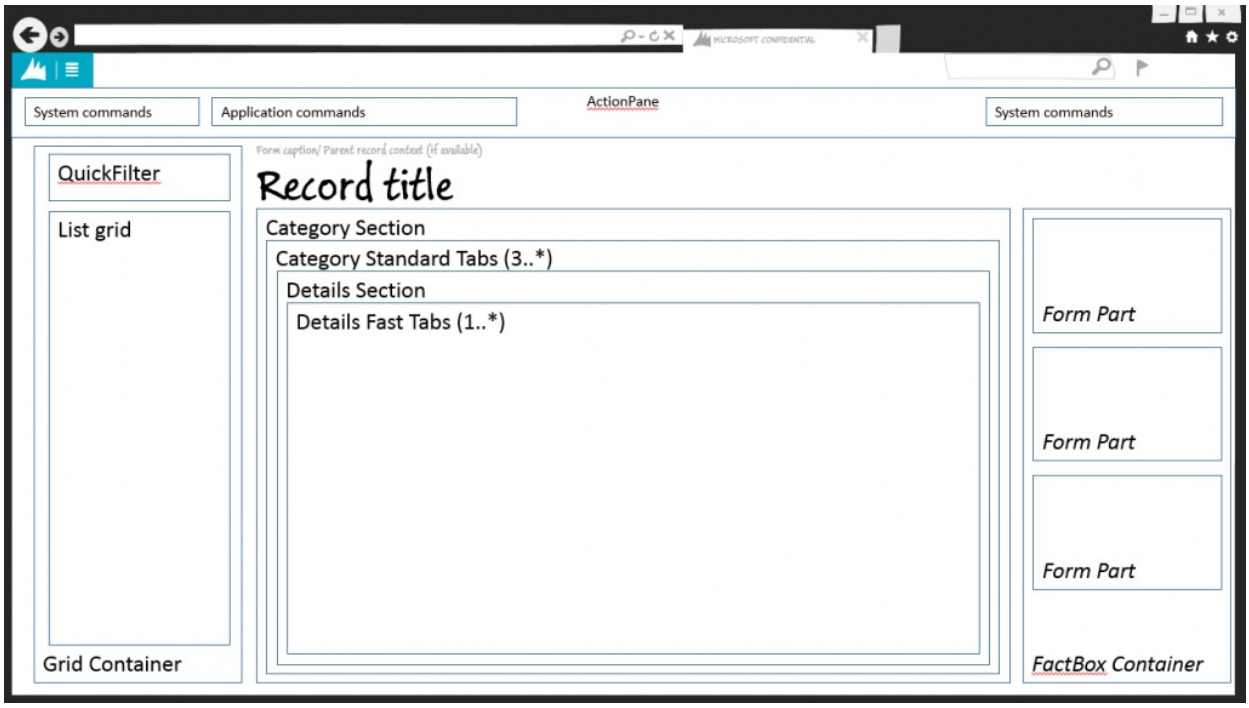


Grid view

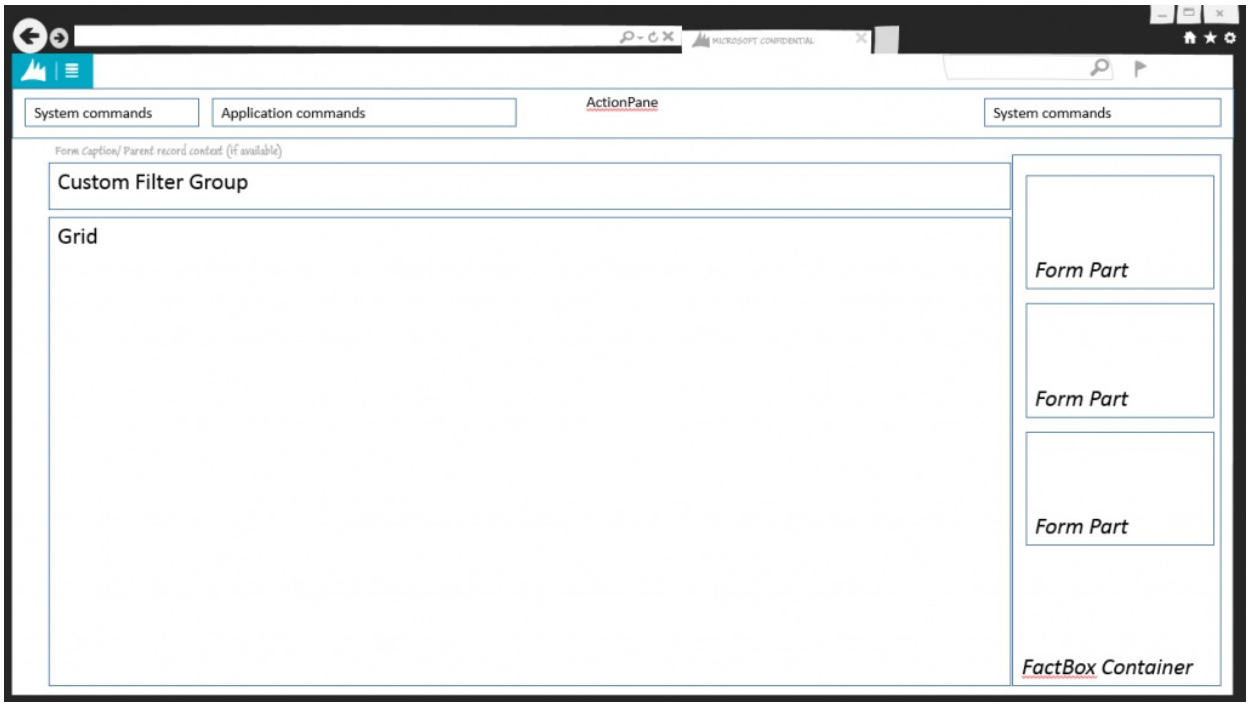


Details Master with Standard Tabs

Details view



Grid view



Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- Added a List style grid to the left of the Details view content.
- Merged List Page and Details Master into a single form.
  - Improves performance when moving between a list and details.
  - Enables bulk editing in the initial list.
  - Allows for elimination of the list page preview pane.
- View/Edit, New, Delete, Save, Refresh, Attachments, and Export to Excel actions are all provided by the foundation and should not have explicit app buttons unless the foundation-provided button is removed.
- Master Details forms that previously used the TOC extension should now use the Master Details w/Standard Tabs pattern.

Model

Details Master (basic) – High-level structure

- Design
  - ActionPane (ActionPane)
  - SidePanel (Group)
    - QuickFilter
    - *CustomFilters (Group) [Optional]*
    - NavigationList (Grid, Style=List)
  - MainTab (Tab ShowTabs=No)
    - DetailsTabPage (TabPage)
      - TitleGroup (Group)
        - HeaderTitle (String)
        - *EntityStatus (Group) [Optional]*

StatusFields (1..N)

- DetailsTab (Tab Style=FastTabs)
  - DetailsTabPage (TabPages *repeats 1..N*)
- GridTabPage (TabPage)
  - CustomFilterGroup (Group)
    - QuickFilter
    - *OtherFilters ($Field) [0..N]*
  - MainGrid (Grid)
  - MainGridDefaultAction (CommandButton)

Details Master with Standard Tabs – High-level structure

- Design
  - ActionPane (ActionPane)
  - SidePanel (Group)
    - QuickFilter
    - *CustomFilters (Group) [Optional]*
    - NavigationList (Grid, Style=List)
  - MainTab (Tab ShowTabs=No)
    - DetailsTabPage (TabPage)
      - TitleGroup (Group)
        - HeaderTitle (String)
        - *EntityStatus (Group) [Optional]*
          - StatusFields (1…N)
      - CategoryTab (Tab Style=Tabs)
        - CategoryTabPage (TabPages *repeats 3..N*)
          - TabHeader (Group)
          - DetailsTab (Tab Style=FastTabs)
            - DetailsTabPage (TabPages *repeats 1..N*)
    - GridTabPage (TabPage)
      - CustomFilterGroup (Group)
        - QuickFilter
        - *OtherFilters ($Field) [0..N]*
      - MainGrid (Grid)
      - MainGridDefaultAction (CommandButton)

Core components

1. Apply the DetailsMaster pattern on **Form.Design**.
2. Address BP Warnings:
   a. **Design.Caption** isn't empty.

b. Form must be referenced by at least one menu item.
c. **TabPage.Caption** isn't empty.

Related patterns

- Details Transaction
- Simple List and Details

Commonly used subpatterns

- Fields and Field Groups
- Toolbar and List
- Toolbar and Fields
- Nested Simple List and Details
- Custom Filter Group

UX guidelines

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn't include any guidelines that will be enforced automatically through the development environment. Open the form in a browser, and walk through these steps. **Standard form guidelines:**

- Standard form guidelines have been consolidated into the Microsoft Dynamics AX General Form Guidelines document.

Detail Master guidelines:

- There should not be any duplicate **New** and **Delete** buttons.
- Should use FastTabs to group the fields instead of traditional tabs. The Details Master w/Standard Tabs pattern groups these related FastTabs into traditional tabs.
  - In its **default** state, the content of the first FastTab should be fully visible without scrolling.
  - **FastTabs** guidelines have been consolidated into the General Form Guidelines document.
- **ActionPane** guidelines have been consolidated into the General Form Guidelines document, in the ActionPane guidelines section.
- **Page title area:**
  - The following format should be used: "<ID> : <Description>"
  - A link to the Details page should be provided in the Main Menu when the List page has been merged into the Details page.
  - The page title should be in a plural form.
- **FactBox** guidelines have been consolidated into the FactBox Form Patterns document.
- **Navigation list grid:**
  - The list style grid should not have fields within a grid row that cause the row to span more than three lines.
    - Typically, just the ID and Description are sufficient.
    - There should be at least two fields.
- **Grid view:**
  - The grid has 2 to 15 fields. Typically, all mandatory fields are included, so that records can be created in the grid.
  - A linked field lets the user open the details for the selected record.
  - The Quick filter should default to the most likely field for a filter scenario.
  - **Grid:**
    - The **ID** field should be the first column (if it's needed in the grid). Otherwise, the **Name** field should be the first column.

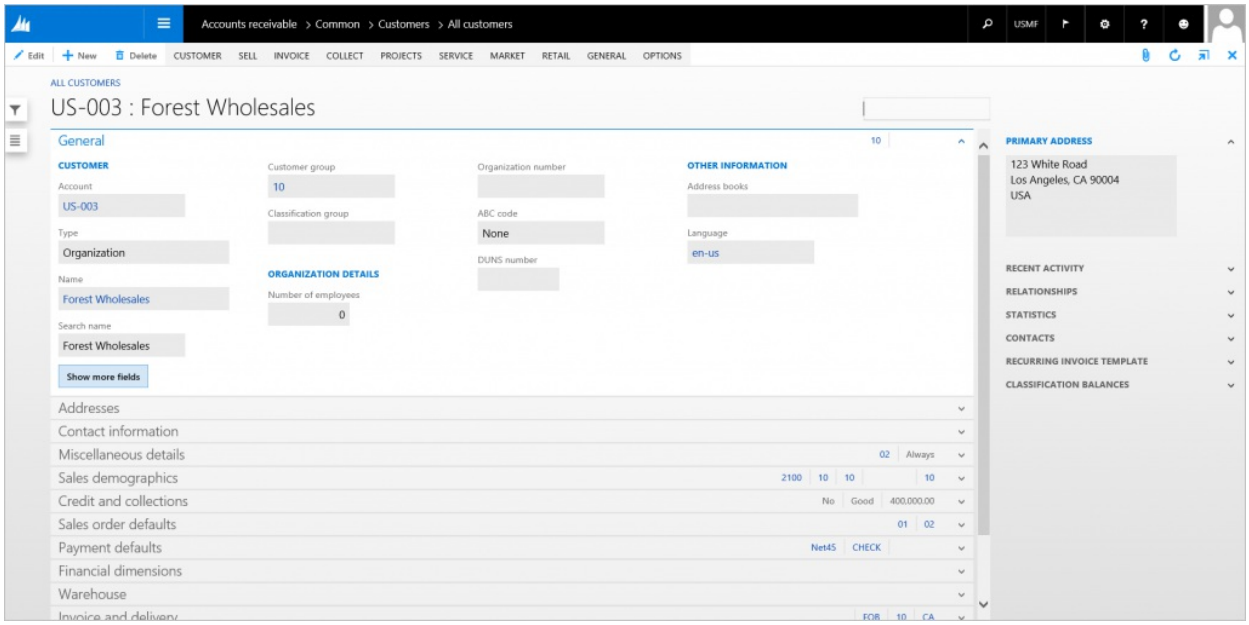
- Additional grid guidelines have been consolidated into the General Form Guidelines document, in the Grid guidelines section.

Examples

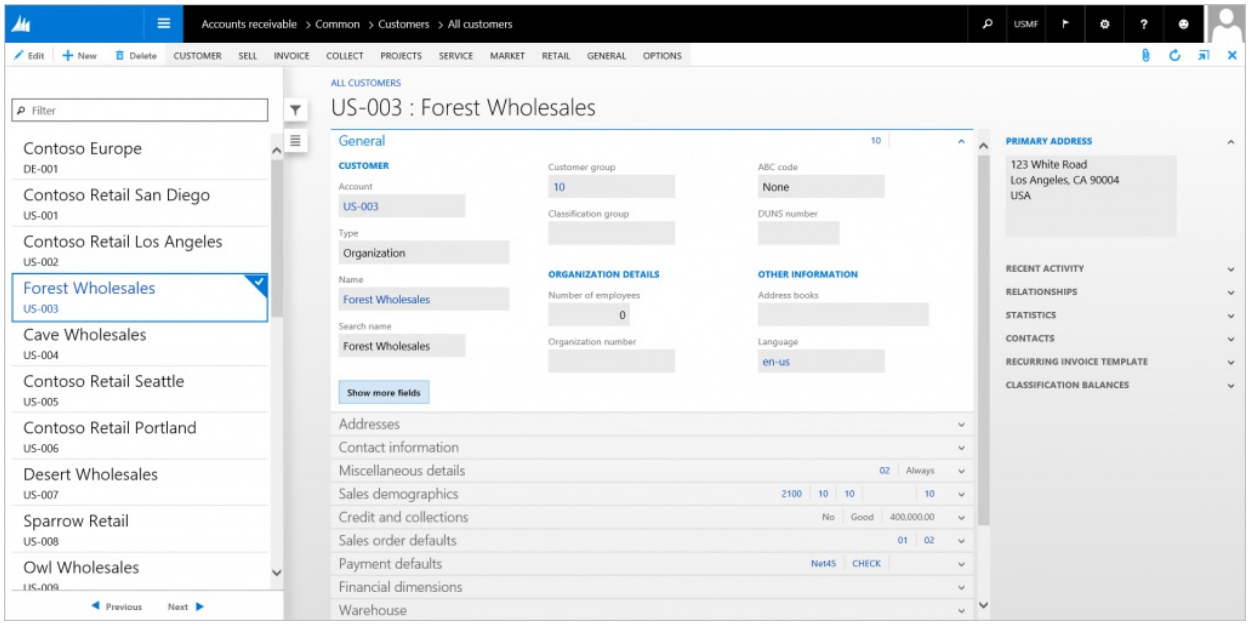
Details Master (basic)

Form: CustTable

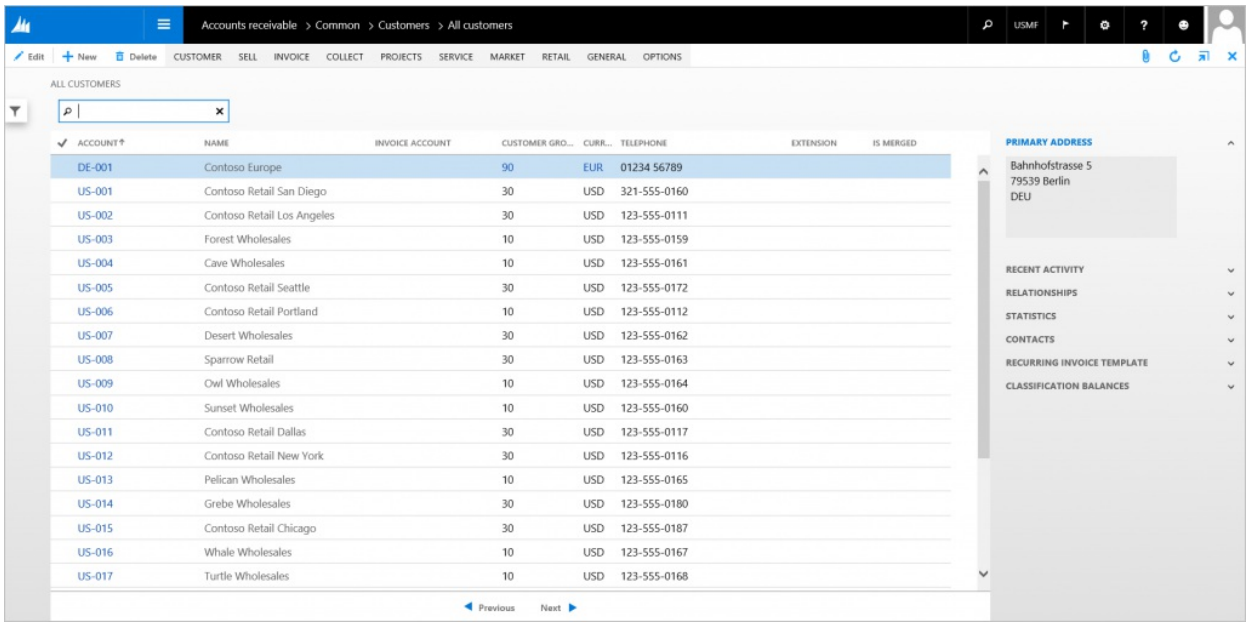
Details view (navigation list off)



Details view (navigation list on)



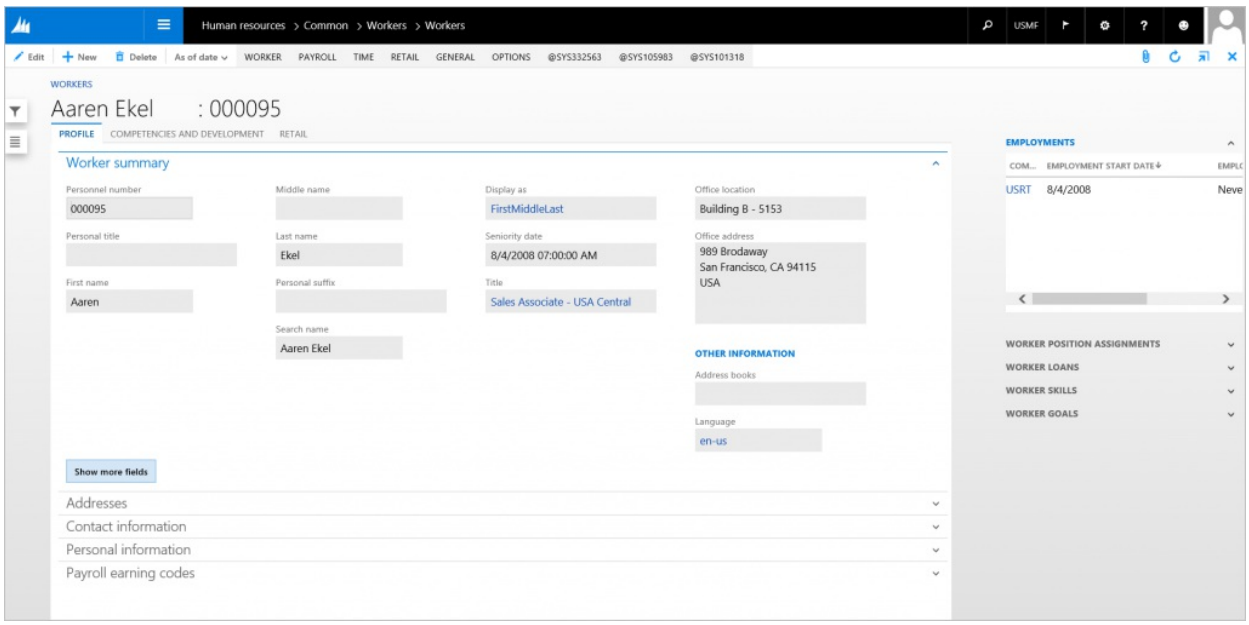
Grid view



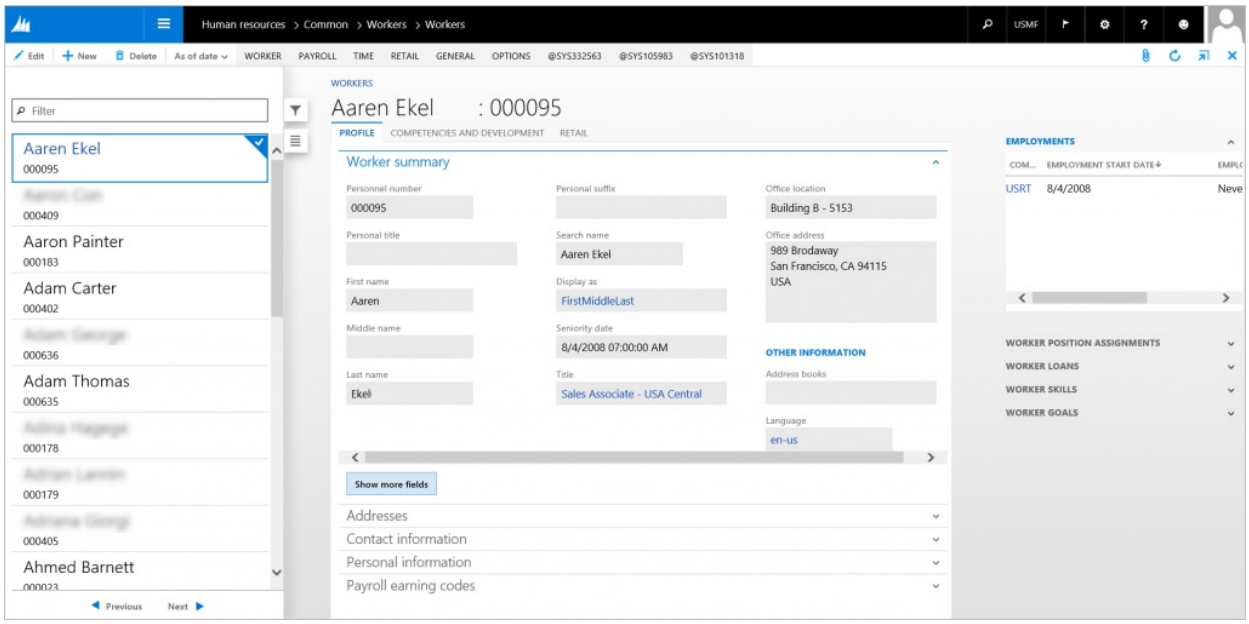
Details Master with Standard Tabs

Form: HcmWorker

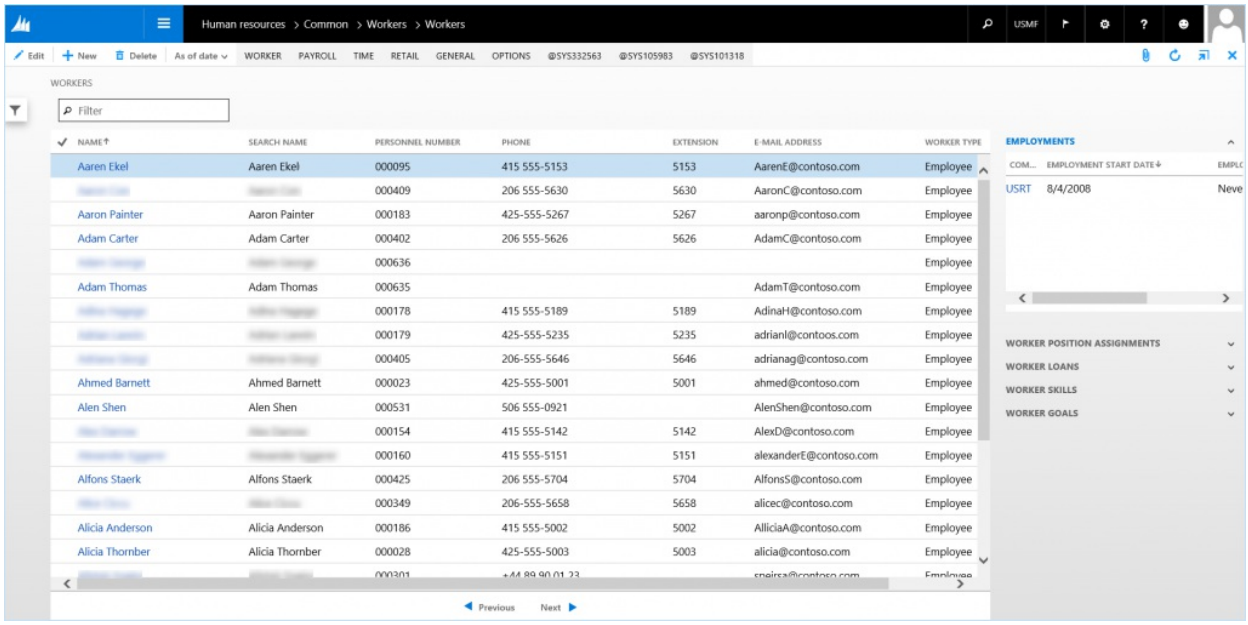
Details view (navigation list off)



Details view (navigation list on)



Grid view



Appendix

Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues

None.

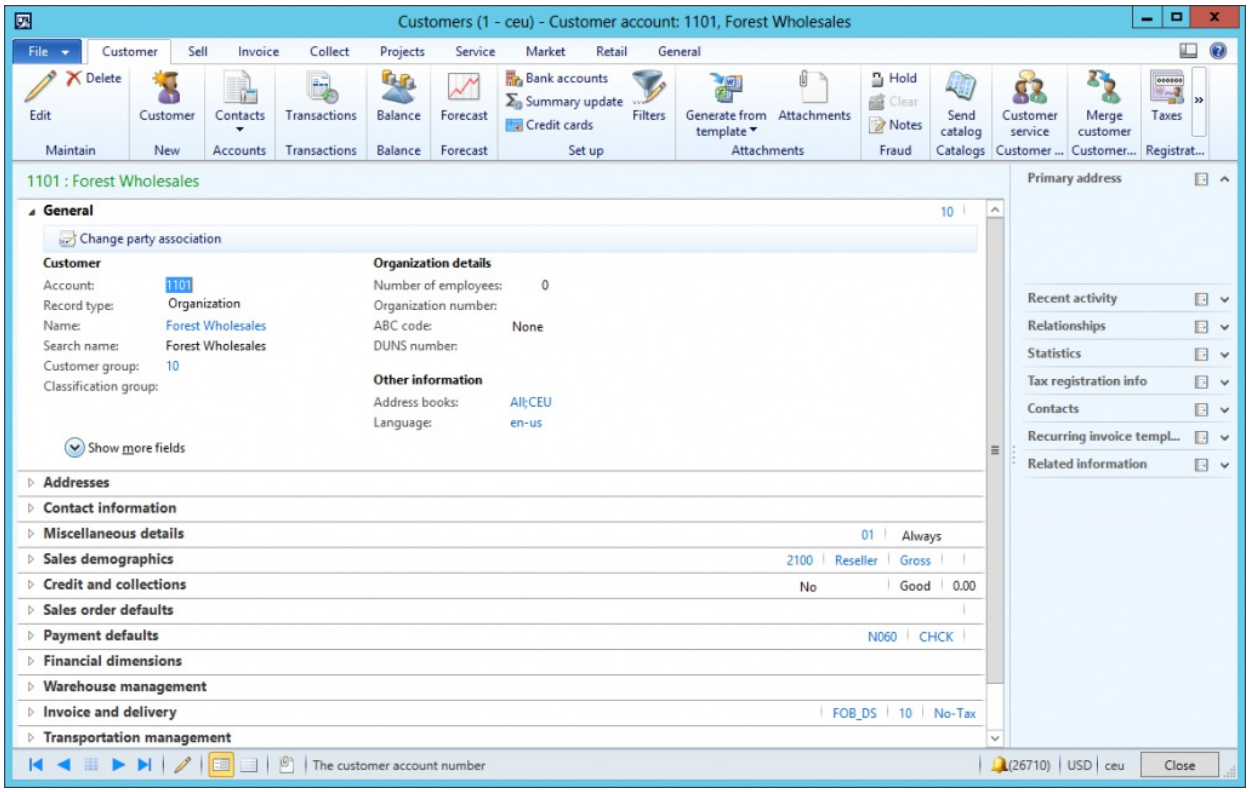
AX 2012 content

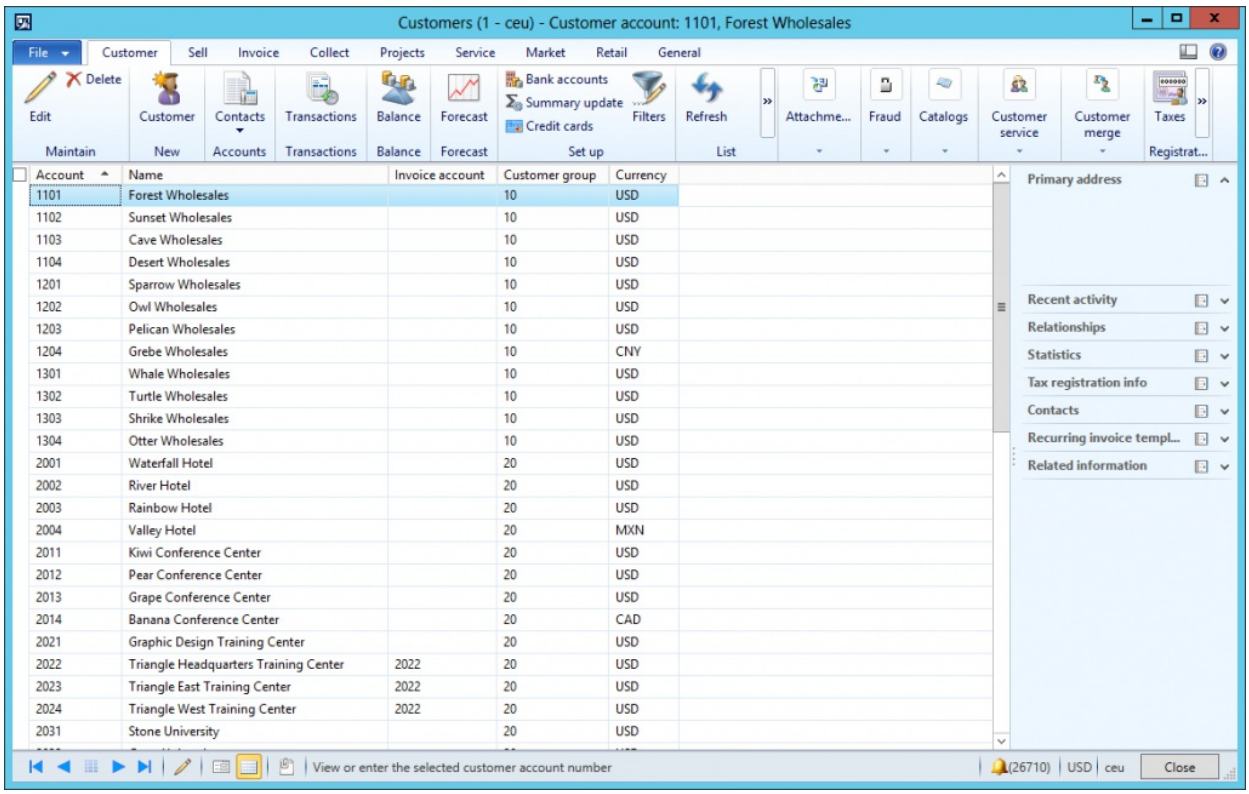
AX 2012 links

- AX 2012 MSDN Details Forms

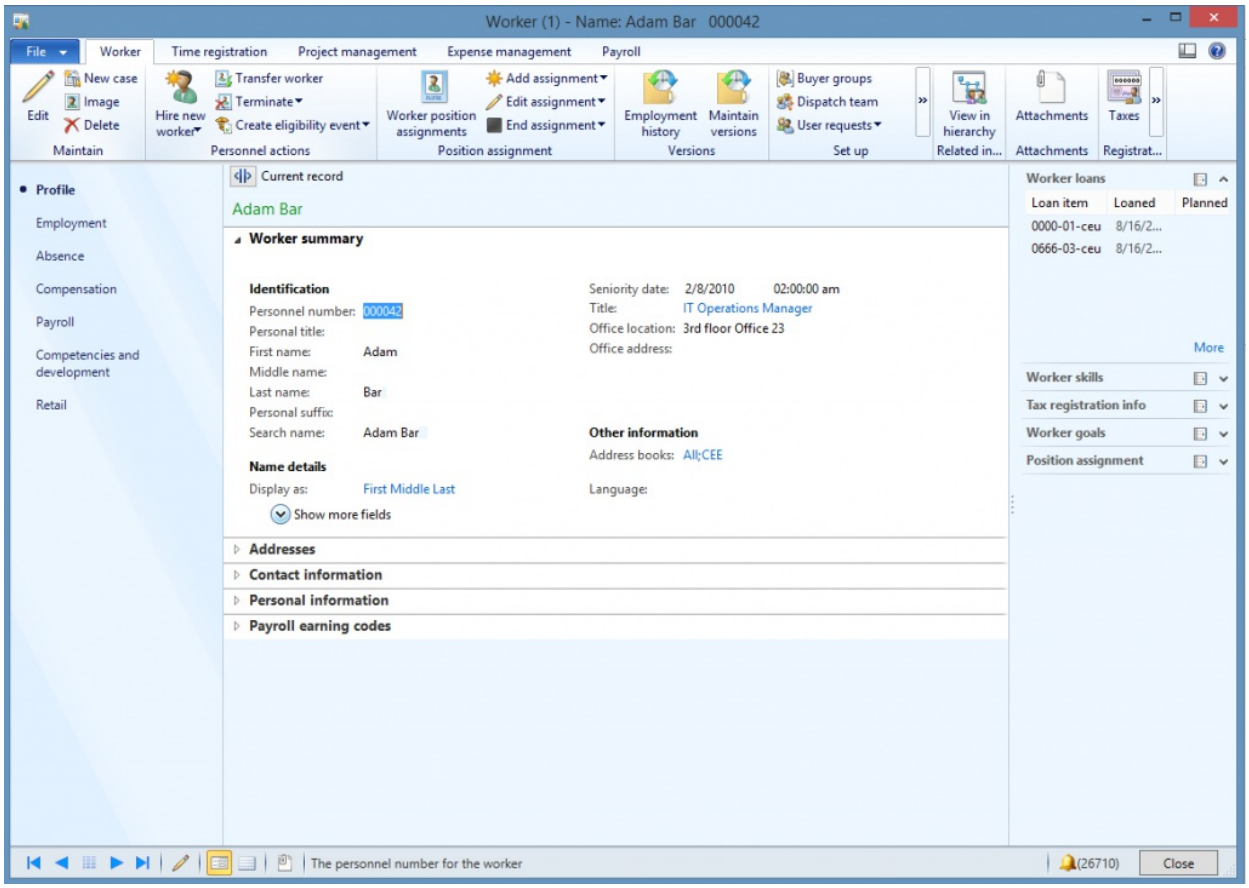
AX 2012 example

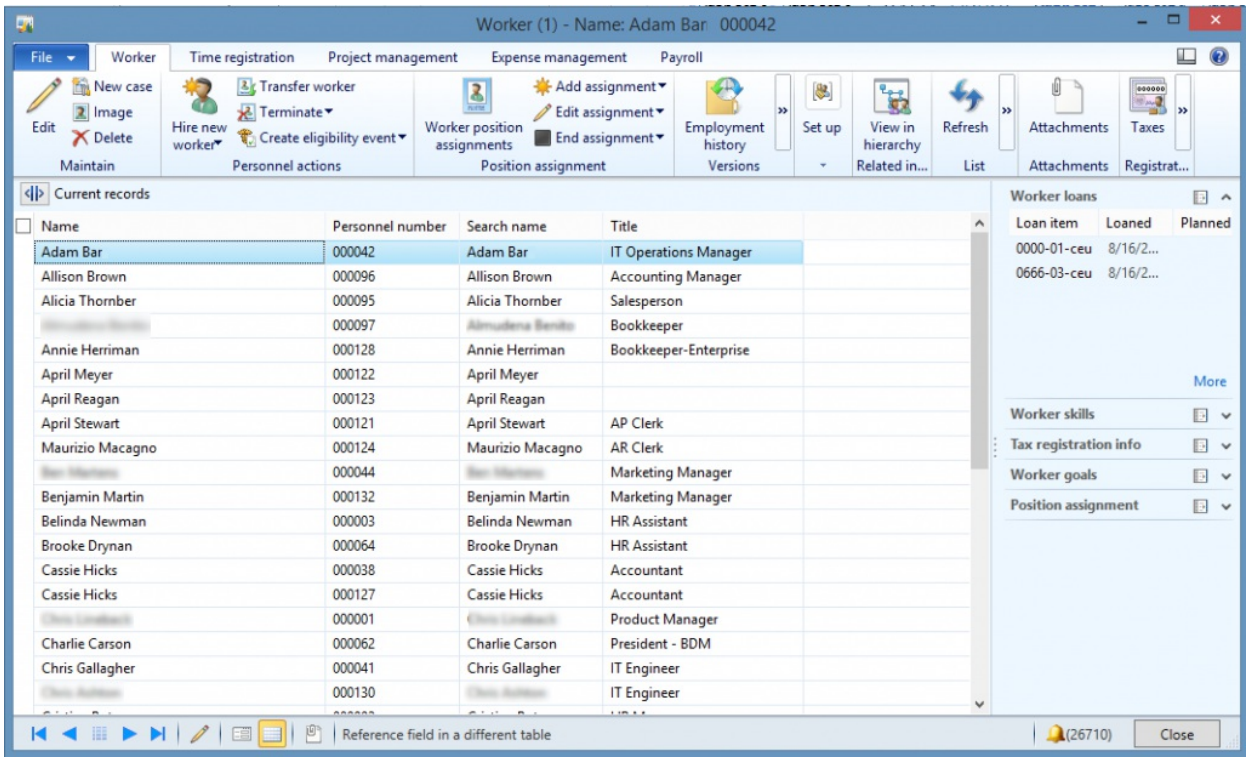
Details Master (basic)





Details Master with Standard Tabs





NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Details Transaction form pattern

2/18/2021 • 4 minutes to read • Edit Online

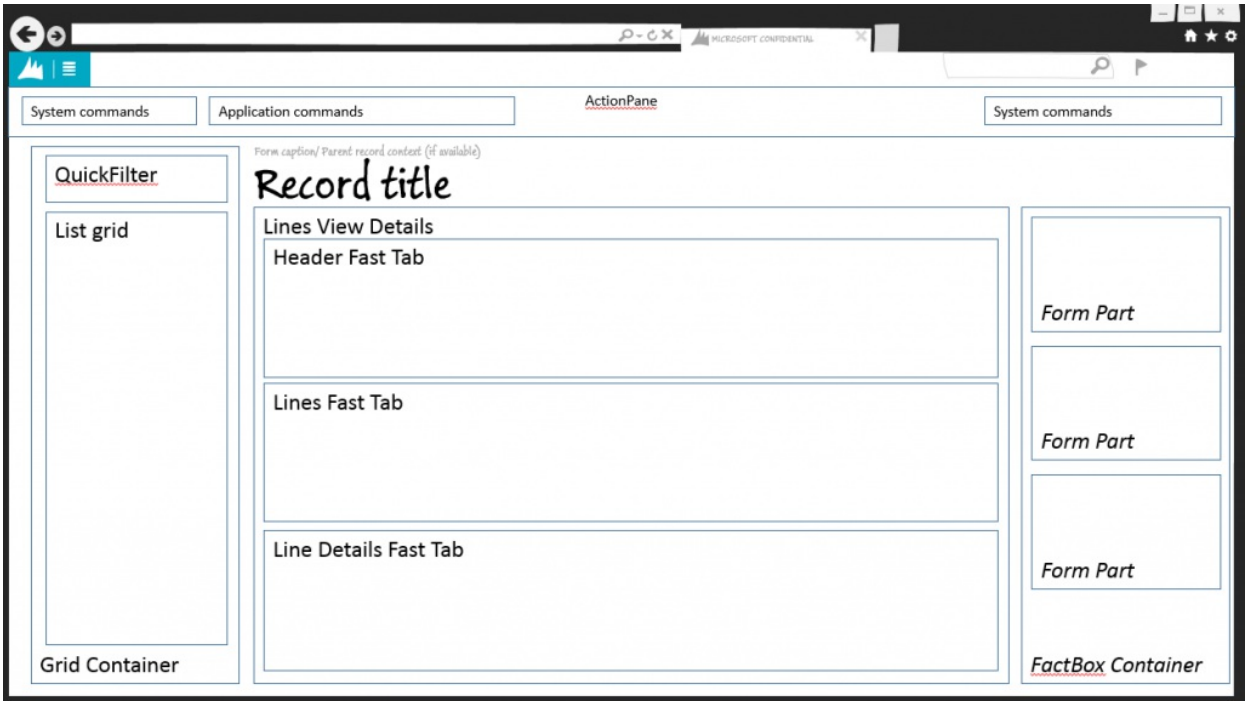
This topic provides information about the Details Transaction form pattern. Forms that use this pattern can have two details views that the user can switch between - a Header view and a Line view.

Usage

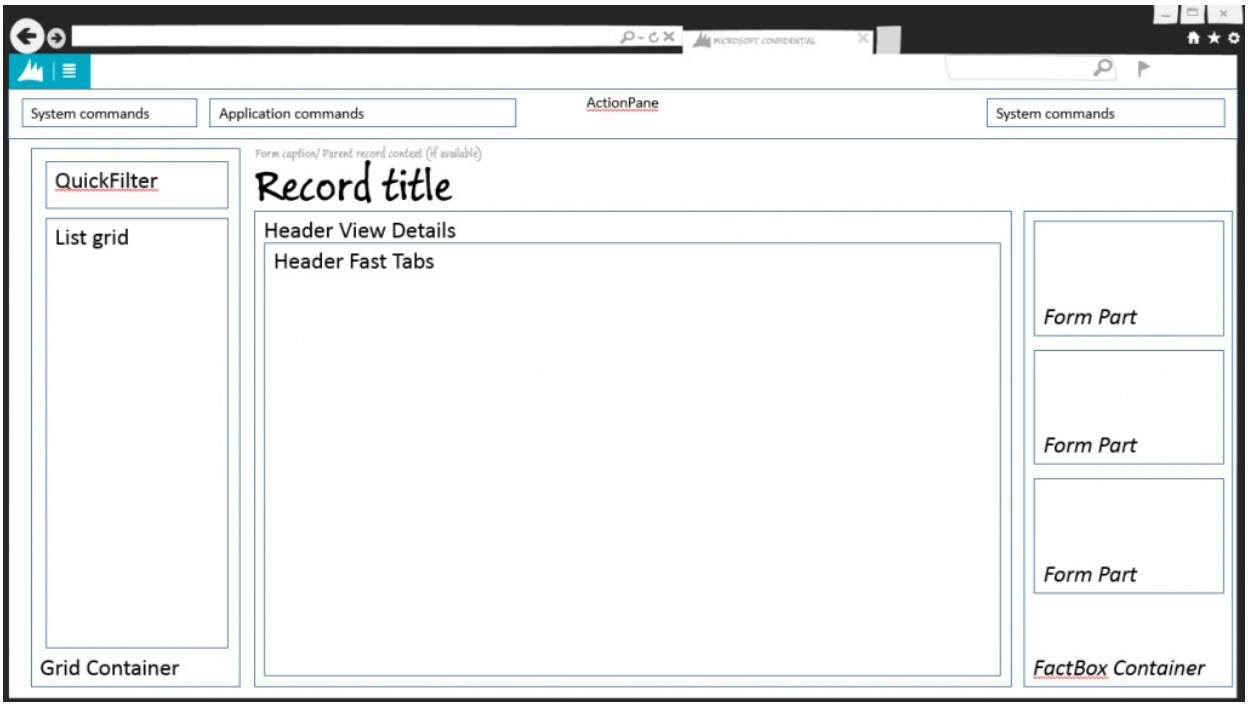
A details form with lines (Details Transaction form) consists of one form that can have two details views that the user can switch between. The Header view contains all fields that are related to or part of the header. The Line view contains the lines grid, line details, and a section that contains a collection of the most important header fields.

Wireframe

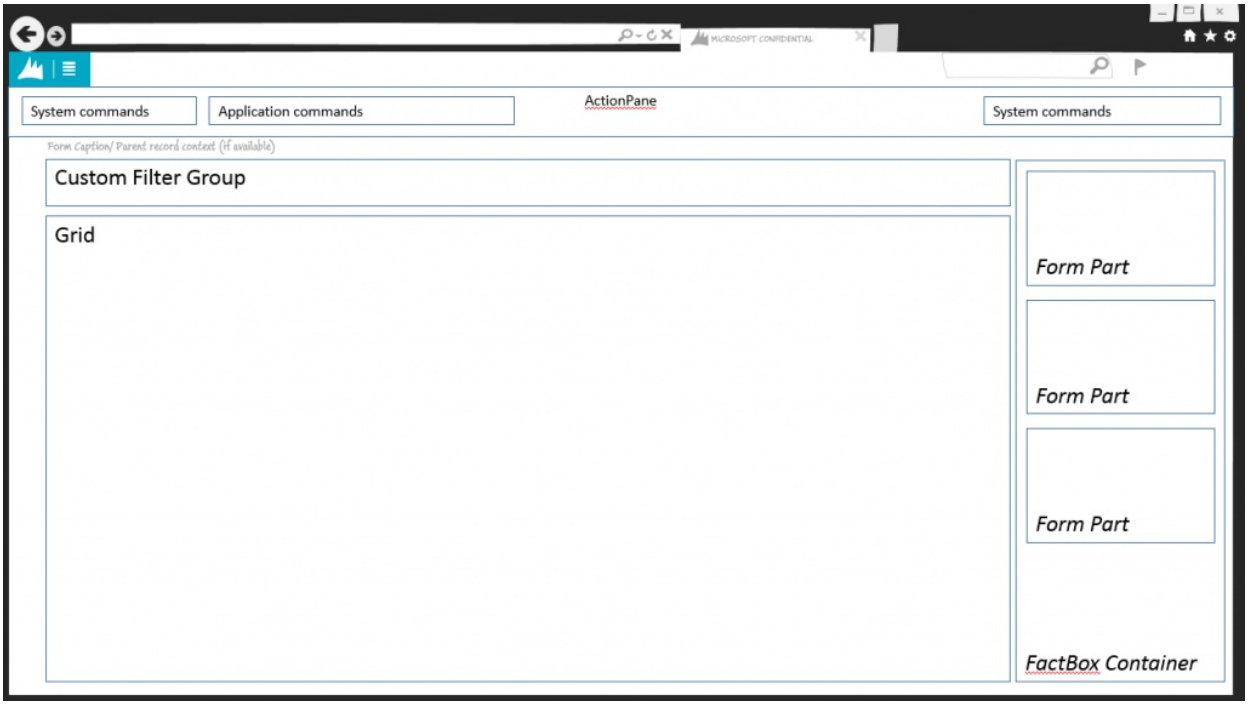
Line view



Header view



Grid view



Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- A List style grid has been added to the left of the Details view content, which is shown in either Header view or Line view.
- List Page and Details Master have been merged into a single form. This change has the following benefits:
  - It improves performance when users move between the list and details.
  - It enables bulk editing in the initial list.
  - It allows for elimination of the list page preview pane.
- View/Edit, New, Delete, Save, Refresh, Attachments, and Export to Excel actions are all provided by the foundation and should not have explicit app buttons unless the foundation-provided button is removed.

Model

High-level structure

- Design
  - ActionPane (ActionPane)
  - SidePanel (Group)
    - QuickFilter
    - *CustomFilters (Group) [Optional]*
    - NavigationList (Grid, Style=List)
  - PanelTab (Tab ShowTabs=No)
    - DetailsPanel (TabPage)
      - TitleGroup (Group)
        - HeaderTitle (String)
        - *EntityStatus (Group) [Optional]*
          - StatusFields (1..N)
      - HeaderLinePanels (Tab ShowTabs=No)
        - LinePanel (TabPage PanelStyle=Line)
          - LineViewTab (Tab Style=FastTabs)
            - LineViewHeader (TabPage)
            - LineViewLines (TabPage)
            - LineViewLineDetails (TabPage)
              - LineDetailsTab (Tab Style=Standard)
                - LineDetailsTabPages (TabPages 1..N)
        - HeaderPanel (TabPage PanelStyle=Header)
          - HeaderViewTab (Tab Style=FastTabs)
            - HeaderViewTabPages (TabPages 1..N)
    - GridPanel (TabPage PanelStyle=Grid)
      - CustomFilterGroup (Group)
        - QuickFilter
        - *OtherFilters ($Field) [0..N]*
      - MainGrid (Grid)
      - MainGridDefaultAction (CommandButton)

Core components

1. Apply the DetailsTransaction pattern on **Form.Design**.
2. Address BP Warnings:
   a. **Design.Caption** isn't empty.
   b. The form must be referenced by at least one menu item.
   c. **TabPage.Caption** isn't empty.

d. **TabPage.DataSource** isn't empty.

Related patterns

- Details Master
- Simple List and Details

Commonly used subpatterns

- Fields and Field Groups
- Toolbar and List
- Toolbar and Fields
- Nested Simple List and Details
- Custom Filter Group

UX guidelines

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn't include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps. **Standard form guidelines:**

- Standard form guidelines have been consolidated into the Microsoft Dynamics AX General Form Guidelines document.

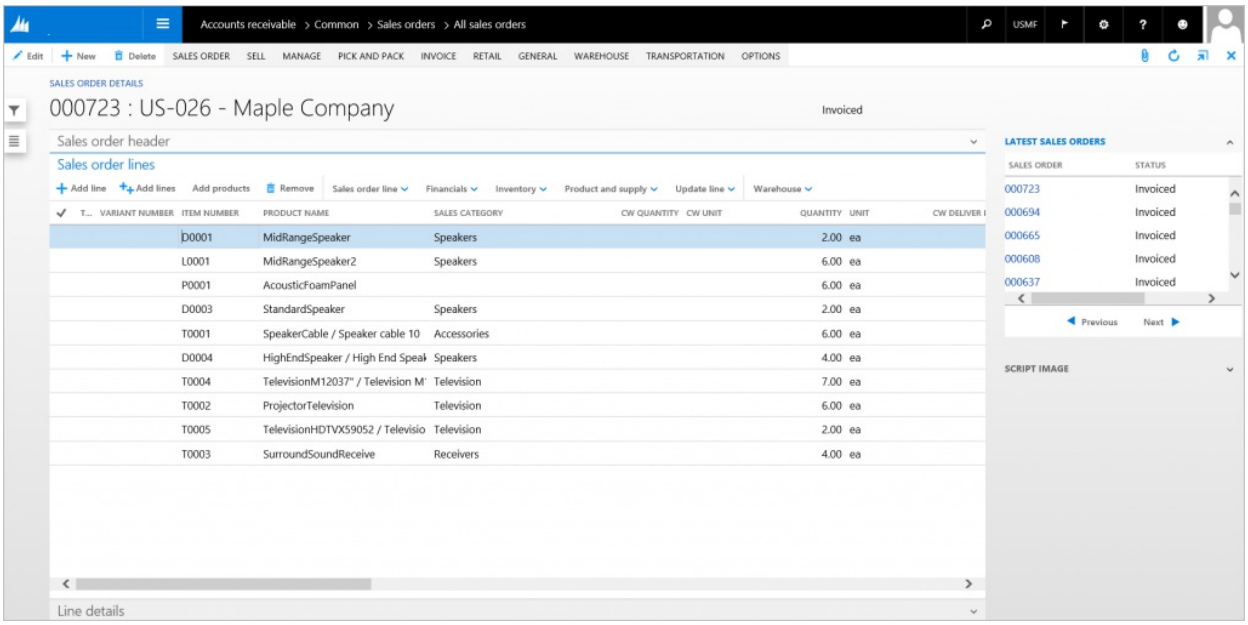
Detail Transaction guidelines:

- There should not be any duplicate **New** and **Delete** buttons.
- **ActionPane** guidelines have been consolidated into the General Form Guidelines document, in the ActionPane guidelines section.
- In its **default** state, the content of the first FastTab should be fully visible without scrolling.
- **FastTabs** guidelines have been consolidated into the General Form Guidelines document.
- **Page title area**:
  - The following format should be used: **<ID> : <Description>**
  - A link to the Details page should be provided on the Main Menu after the List page has been merged into the Details page.
  - The page title should be in a plural form.
- **Navigation list grid**:
  - The list style grid should not have fields within a grid row that spans more than three lines.
    - Typically, just the ID and Description are sufficient.
    - There should be at least two fields.
  - The last field should be the total of the transaction.
- **Grid view**:
  - The grid has 2 to 15 fields. Typically, all mandatory fields are included, so that records can be created in the grid.
  - A linked field lets the user open the details for the selected record.
  - By default, the Quick Filter should use the most likely field for a filter scenario.
  - Focus should be in the Quick Filter when the list page is opened.
  - **Grid**:
    - The **ID** field should be the first column, followed by the master entity **ID** and **Name** fields.
    - Additional grid guidelines have been consolidated into the General Form Guidelines document, in the Grid guidelines section.
  - **FactBox** guidelines have been consolidated into the FactBox Form Patterns document.

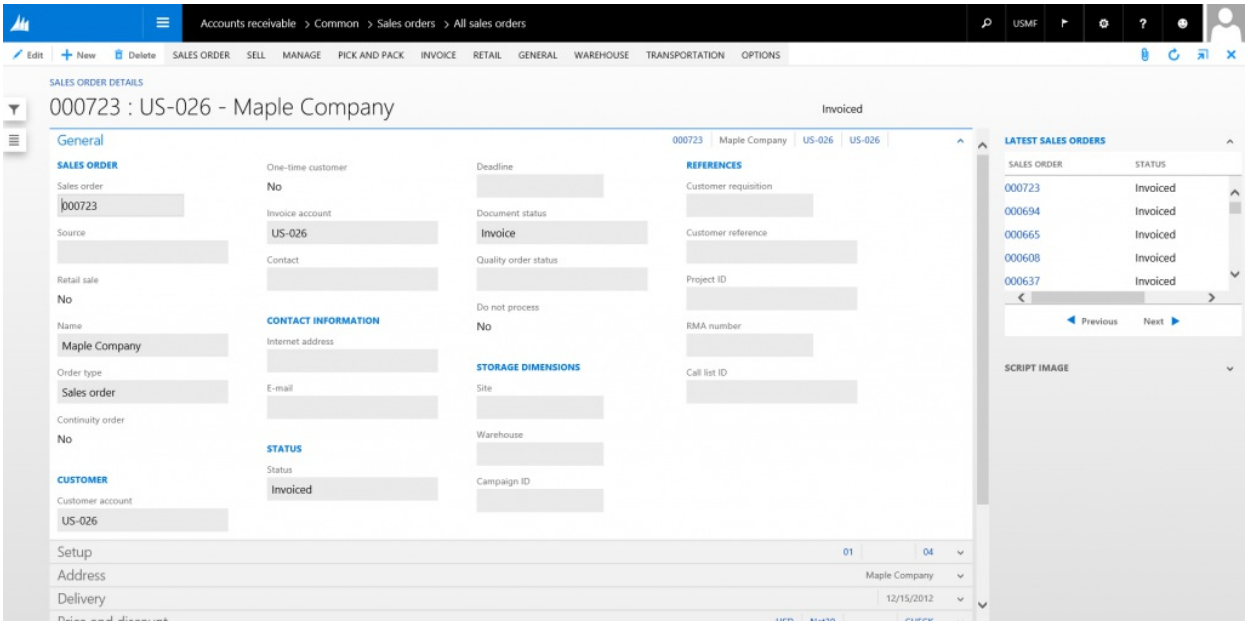
Example

Form: **SalesTable**

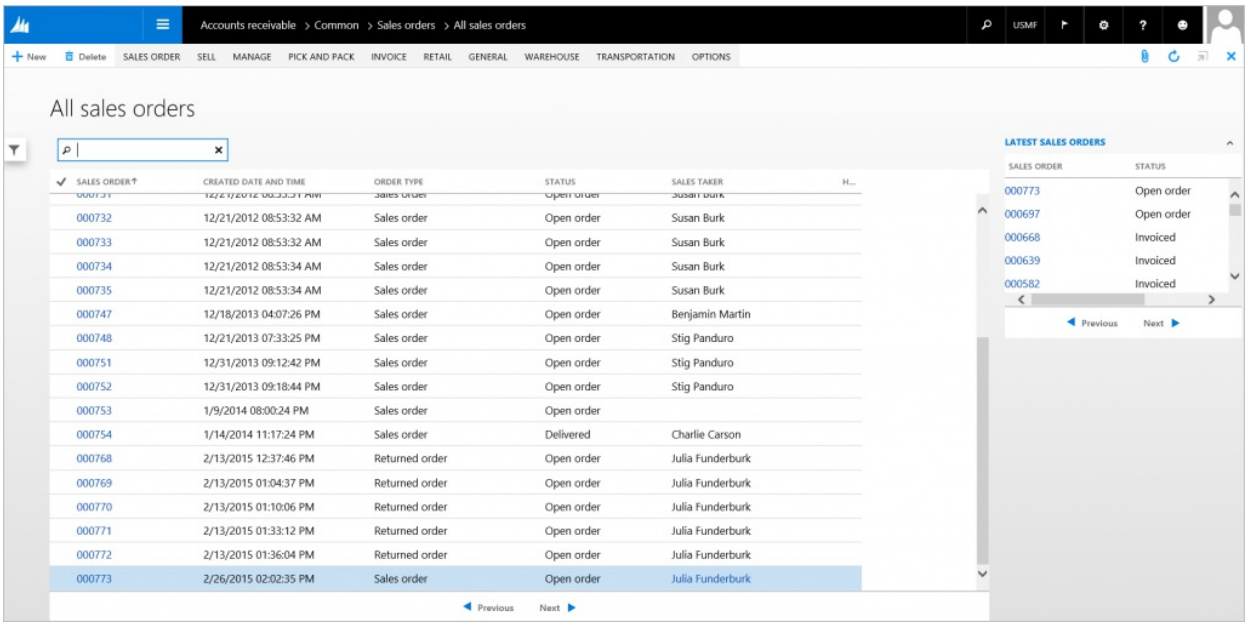
Line view



Header view



Grid view



Appendix

Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

- **Why is the Header view compulsory?**
  - The Header view is compulsory for the Details Transaction pattern. Initially, the Header view might not have more than the Line view header summary information. However, over time, it will be extended by application teams, internationalization teams, partners, and customers. It's important that the Header view be available for future modifications. In addition, a consistent and dependable form structure has benefits for usability and upgrade reasons.

Open issues

None currently.

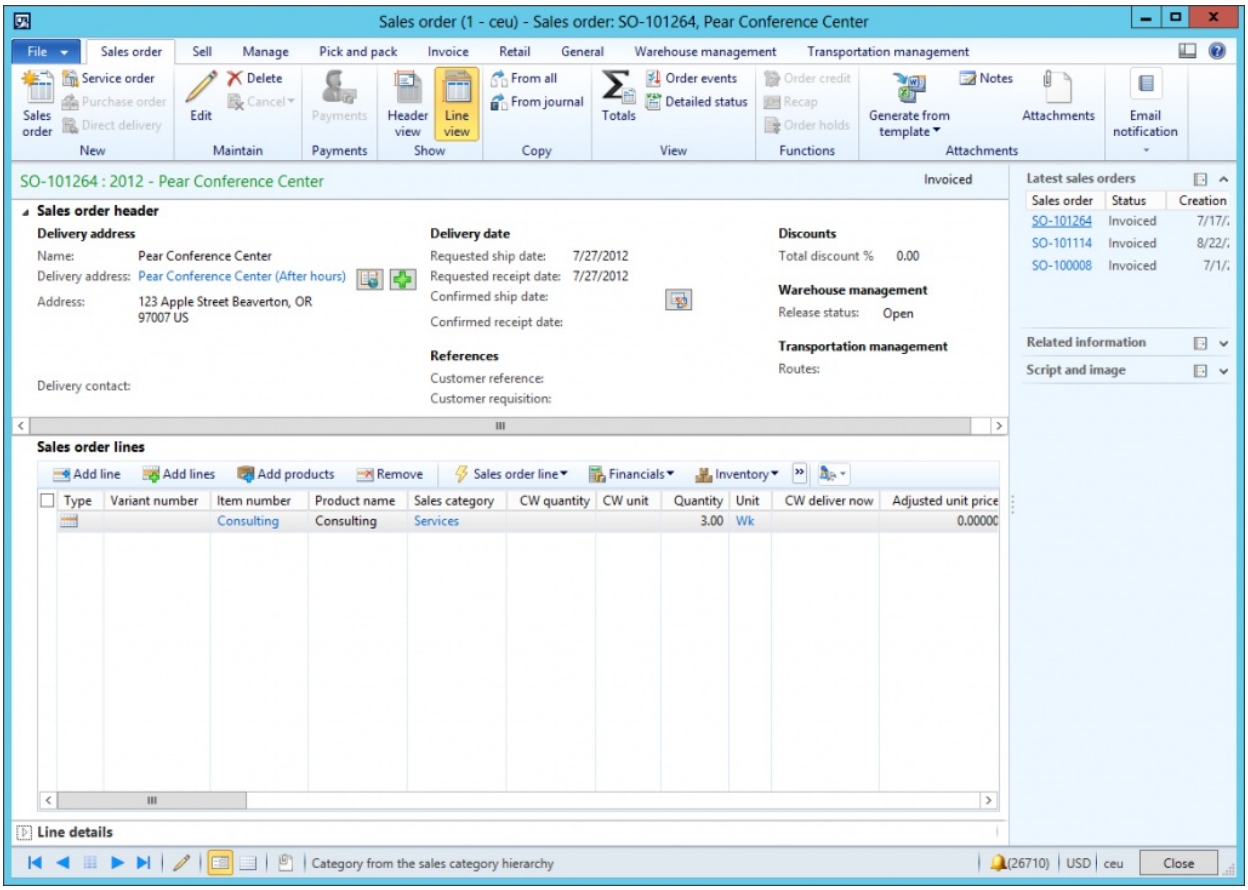
AX 2012 content

AX 2012 links

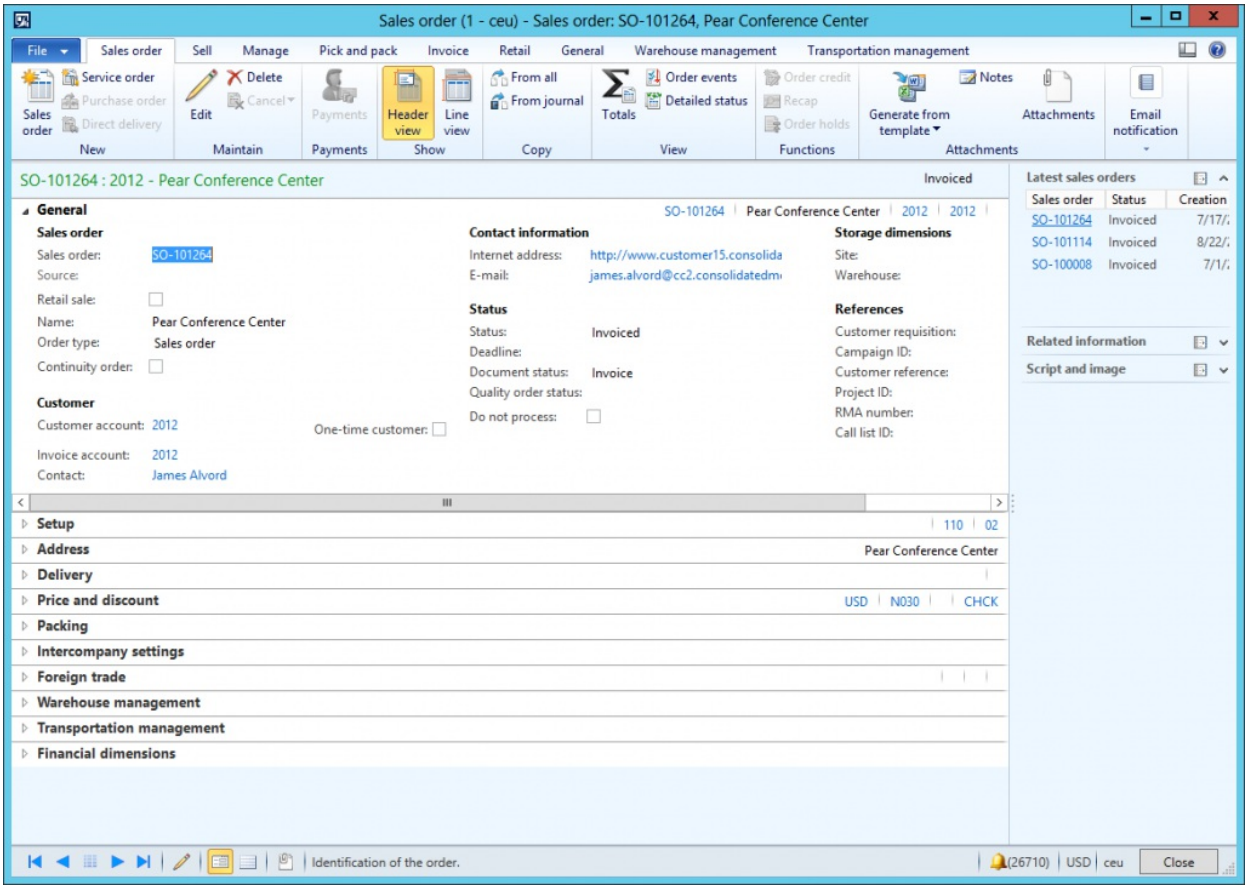
- MSDN Details Form with Lines User Experience Guidelines [AX 2012]
- MSDN Details Form [AX 2012]

AX 2012 example

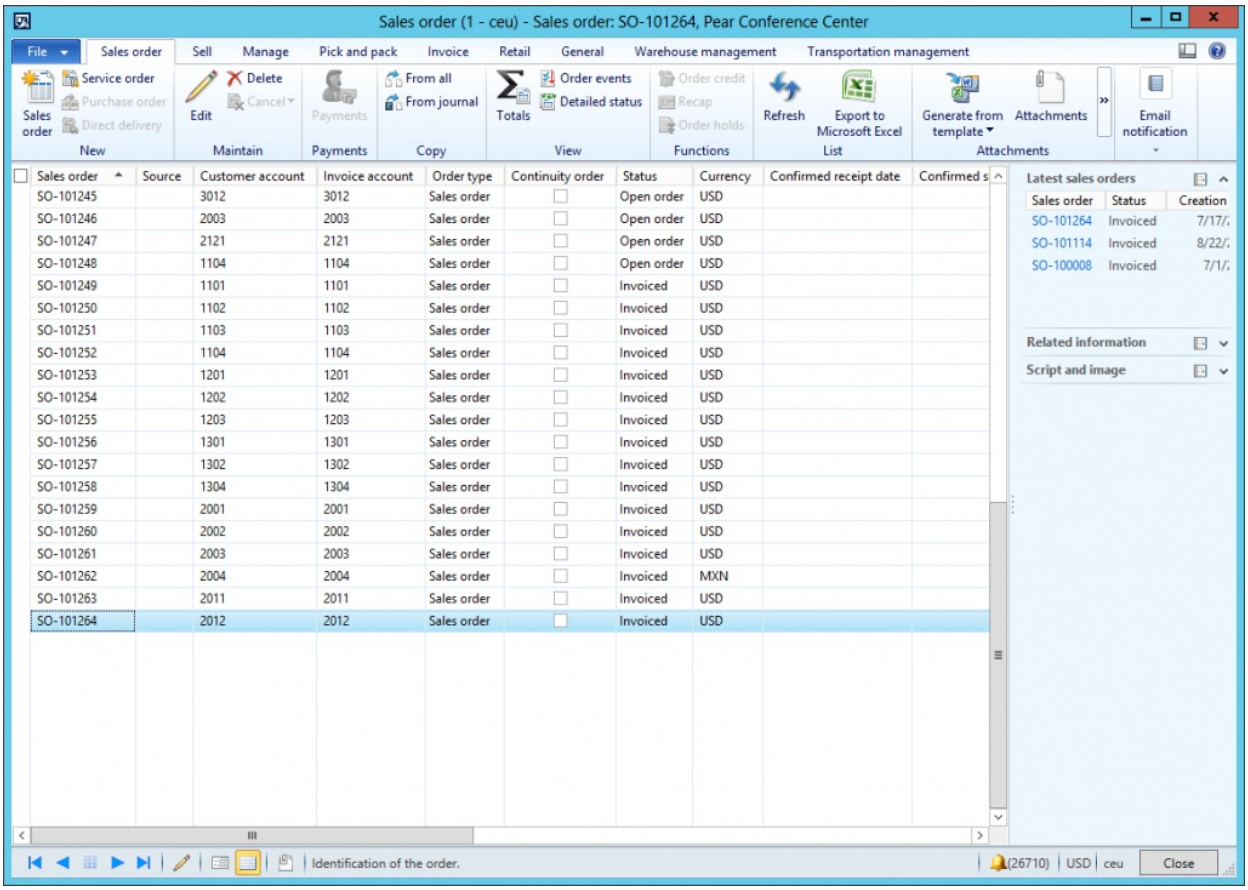
Line view



Header view



Grid view



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Form Part Section List form patterns

2/18/2021 • 3 minutes to read • Edit Online

This topic provides information about the Form Part Section List form patterns. These workspace-specific patterns have been developed to show filtered lists inside workspaces.

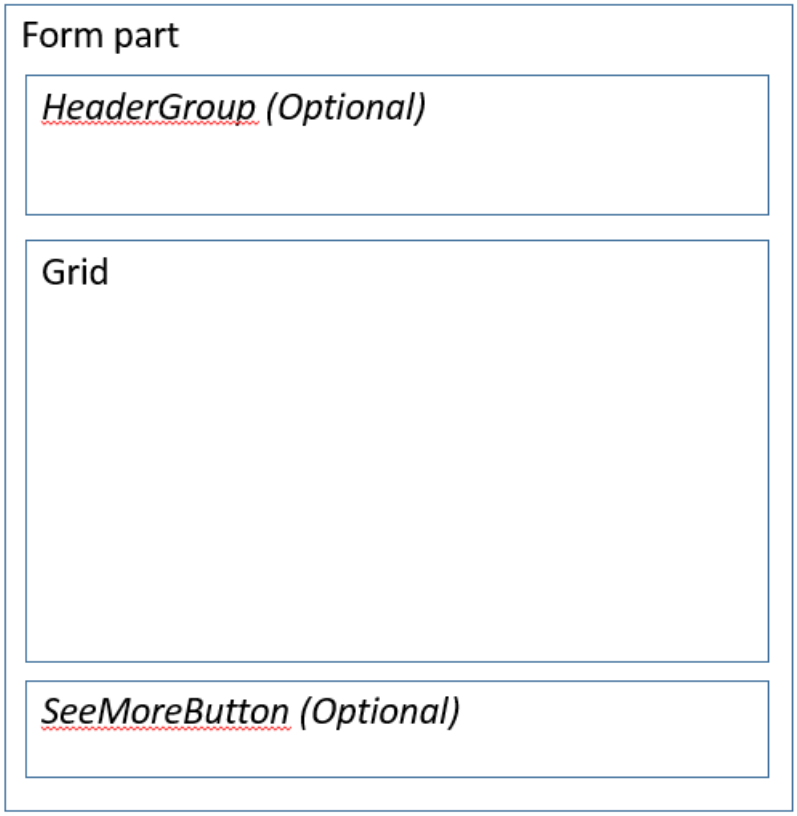
Usage

The Form Part Section List form patterns are workspace-specific patterns that are used to show filtered lists. The tabbed section of the workspace contains a set of vertical tabs. Each tab contains a Form Part Control that points to a form that contains one of the Form Part Section List patterns. Two patterns are described in this article:

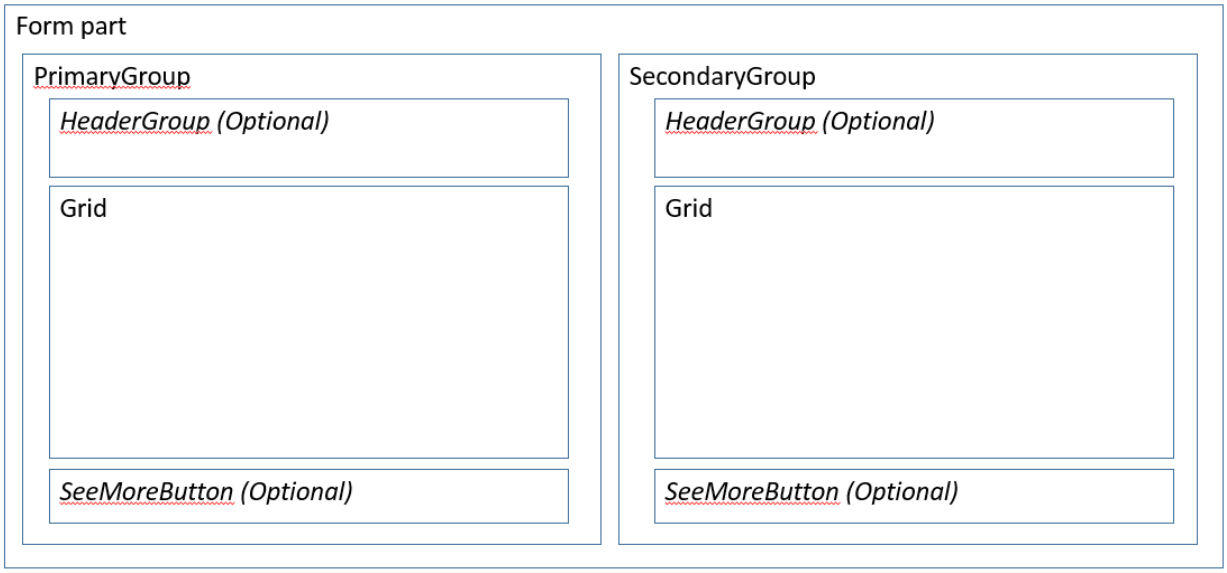
- **Form Part Section List** – This is the default Section pattern. It allows for a single list of data, together with an optional header group that contains filters and/or actions. Most content areas in the tabbed section of a workspace will use this pattern.
- **Form Part Section List - Double** – This variant enables a second list of data to appear to the right of the primary list. By default, the secondary list is hidden. To show it, the user clicks a button on the Toolbar above the primary list.

Wireframe

Form Part Section List



Form Part Section List - Double



Pattern changes for Finance and Operations

These patterns did not exist for Microsoft Dynamics AX 2012.

Model

Form Part Section List: High-level structure

- Design | Container
  - *Header (Group) [Optional]* – This must use one of the Filters and Toolbar subpatterns.
  - Grid
  - *GridDefaultAction (Button) [Optional]*
  - *SeeMoreButton (Button) [Optional]*

Form Part Section List - Double: High-level structure

- Design | Container
  - PrimaryGroup (Group)
    - *Header (Group) [Optional]* – This must use one of the Filters and Toolbar subpatterns.
    - Grid
    - *GridDefaultAction (Button) [Optional]*
    - *SeeMoreButton (Button) [Optional]*
  - SecondaryGroup (Group)
    - *Header (Group) [Optional]* – This must use one of the Filters and Toolbar subpatterns.
    - Grid
    - *GridDefaultAction (Button) [Optional]*
    - *SeeMoreButton (Button) [Optional]*

Core components

1. Apply the appropriate Form Part Section List pattern on **Form.Design**.
2. In the backing Operational workspace form, set the Form Part control on the corresponding vertical tab to point to a menu item that points to this form.

Related container patterns

- Section Tabbed List

- Filters and Toolbar

UX guidelines

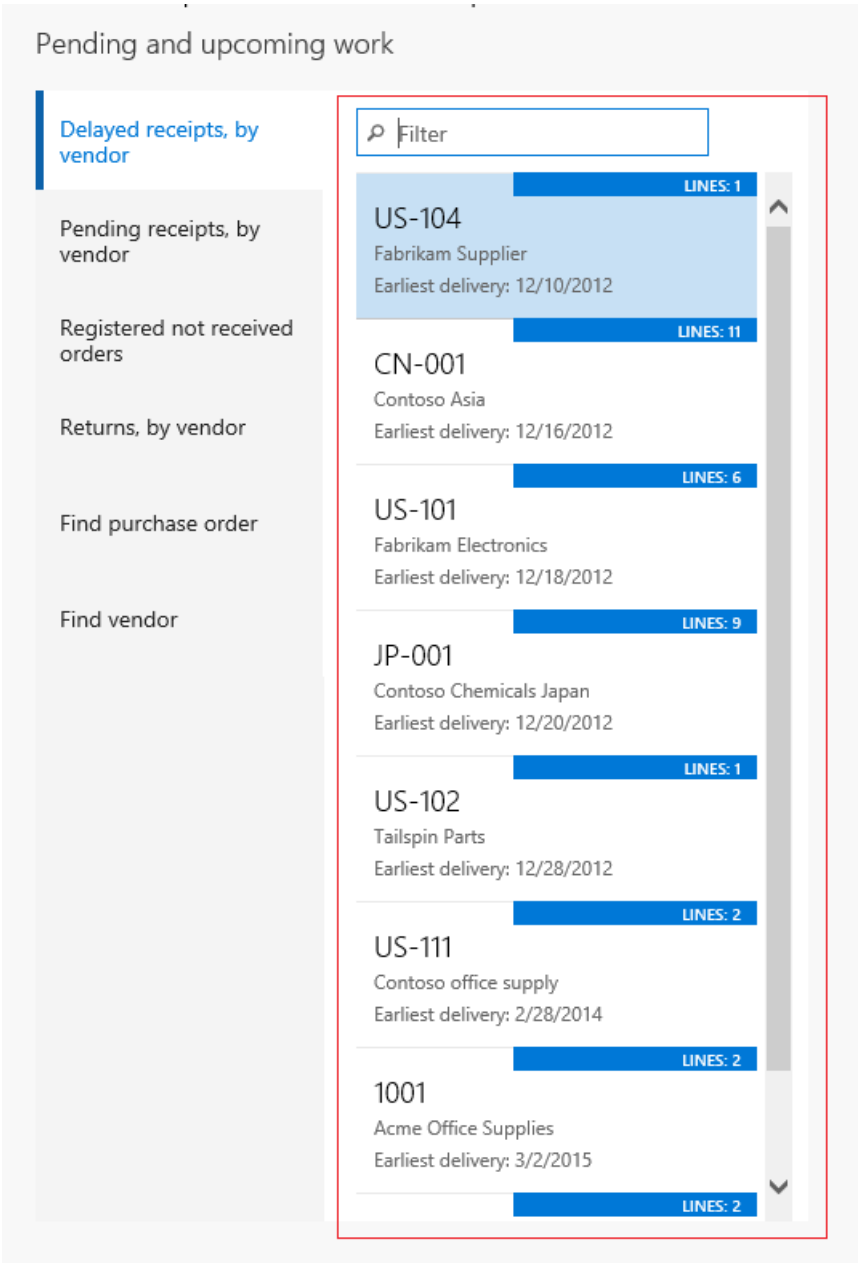
The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn't include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps.

- **General form guidelines**
  - Standard form guidelines have been consolidated into the General Form Guidelines document.
- **Pattern-specific guidelines**
  - If a backing form exists, and especially if not all the records are shown in the list, a **See more** button should appear at the bottom of the list, so that the user can see the full list.
  - Up to two important filters exist above the list.
  - Up to three frequently used actions exist above the list.
- **Grid**
  - Lists are filtered down to an interesting, relatively small set of data.
  - List grids have no more than three lines of data per row.
  - Card grids show no more than four fields (not including an image).
  - Tabular grids show no more than eight fields.
- **Form Part Section List - Double guidelines**
  - If both lists have actions and/or filters, both list must use the same Filters and Toolbar subpattern (either the Stacked variant or the Inline variant).

Examples

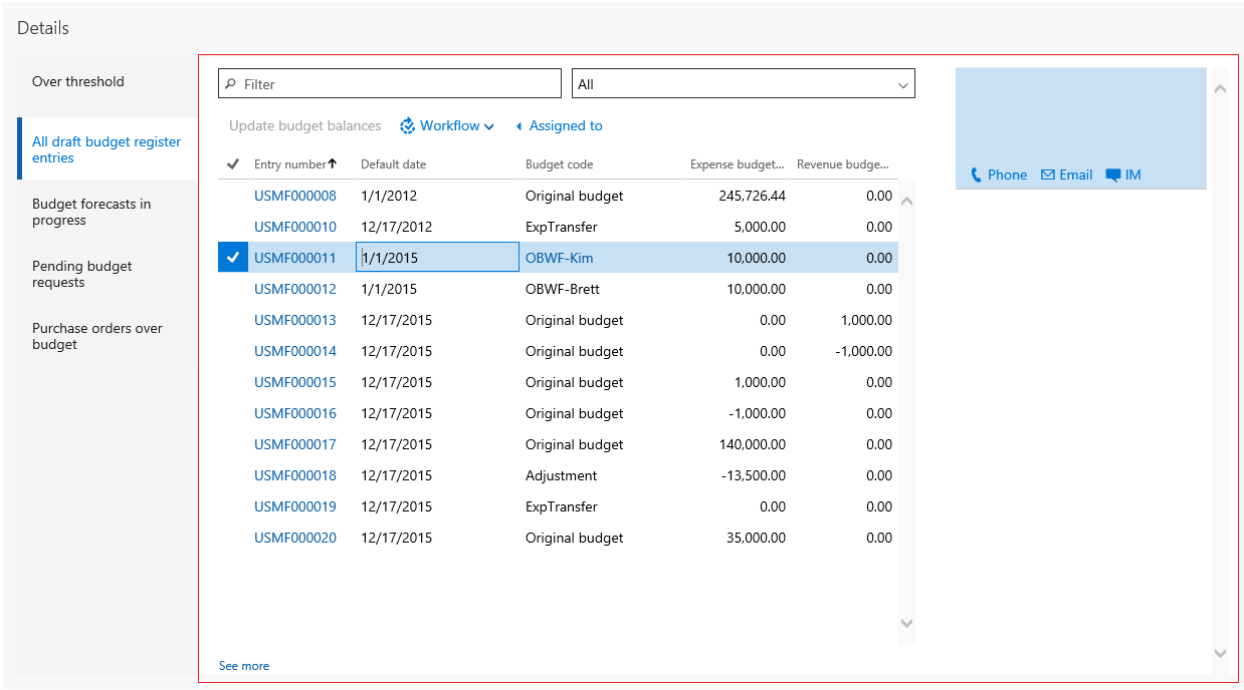
Form Part Section List

Form: **PurchOrderProcessReceiptsWorkspace** > **PurchOrdersWithDelayedReceiptsPart** (**All workspaces** > **Purchase order receipt and follow-up**)



Form Part Section List - Double

Form: **BudgetTrackingWorkspace** > **BudgetTransactionPart** (**All workspaces** > **Ledger budgets and forecasts**)



Appendix

Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues

None

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

List Page form pattern

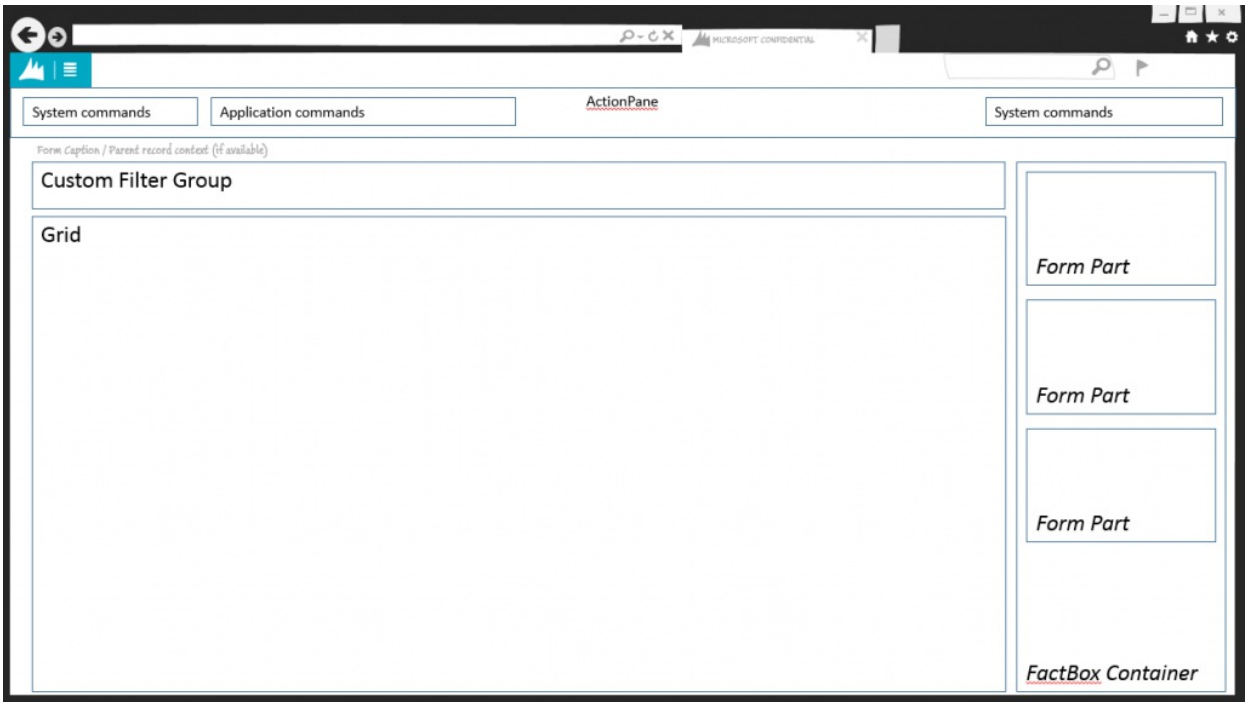
2/18/2021 • 4 minutes to read • Edit Online

This article provides information about the List Page form pattern. A list page presents a set of data on a UI that is optimized for browsing records, so that you can find and work with a specific record.

Usage

A list page presents a set of data on a user interface that is optimized so that you can browse records, find the right record, and then take an action upon that record. The list page lets the user search, filter, and sort the data. FactBoxes on the right side of the grid show related data for the active record. Actions that are relevant to the record are located on the ActionPane at the top of the page. The use of this pattern is now discouraged when there is a 1:1 correspondence between the List Page and Details page. Current guidance is to use this pattern only in other situations, such as when list pages have no backing details pages or have multiple backing details page (for example, when project quotations and sales quotations are shown together in the same List Page).

Wireframe



Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- FormTemplate/InteractionClass is now optional when you build new pages.
- List Page and Details Master/Details Transaction are merged into a single form when there is a 1:1 correspondence between the List Page and Details Page.
  - Improves performance when the user moves between the list and details.
  - Allows for bulk editing in the initial list.
- The **Preview** pane has been eliminated.

Model

High-level structure

- Design
  - ActionPane (ActionPane)
  - Custom Filter (Group)
    - Quick Filter (Quick Filter)
    - *OtherFilters ($Field) [0..N]*
  - Grid (Grid)

Core components

1. Apply the ListPage pattern on **Form.Design**.
2. Address BP Warnings:
   a. **Design.Caption** isn't empty.
   b. The form must be referenced by at least one menu item.
   c. **TabPage.Caption** isn't empty.
   d. **TabPage.DataSource** isn't empty.
   e. The primary data source has **AllowEdit**=**No**, **AllowCreate**=**No**, and **AllowDelete**=**Yes**.
   f. **Grid.DefaultAction** references the button that opens the child form.
   g. **Grid.DefaultLabelAction** references a label to show in the grid context menu.

Related patterns

- Details Master
- Details Transaction
- Simple List

Commonly used subpatterns

- Custom Filter Group

UX guidelines

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn't include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps. **Standard form guidelines:**

- Standard form guidelines have been consolidated into the General Form Guidelines document.

List Page guidelines:

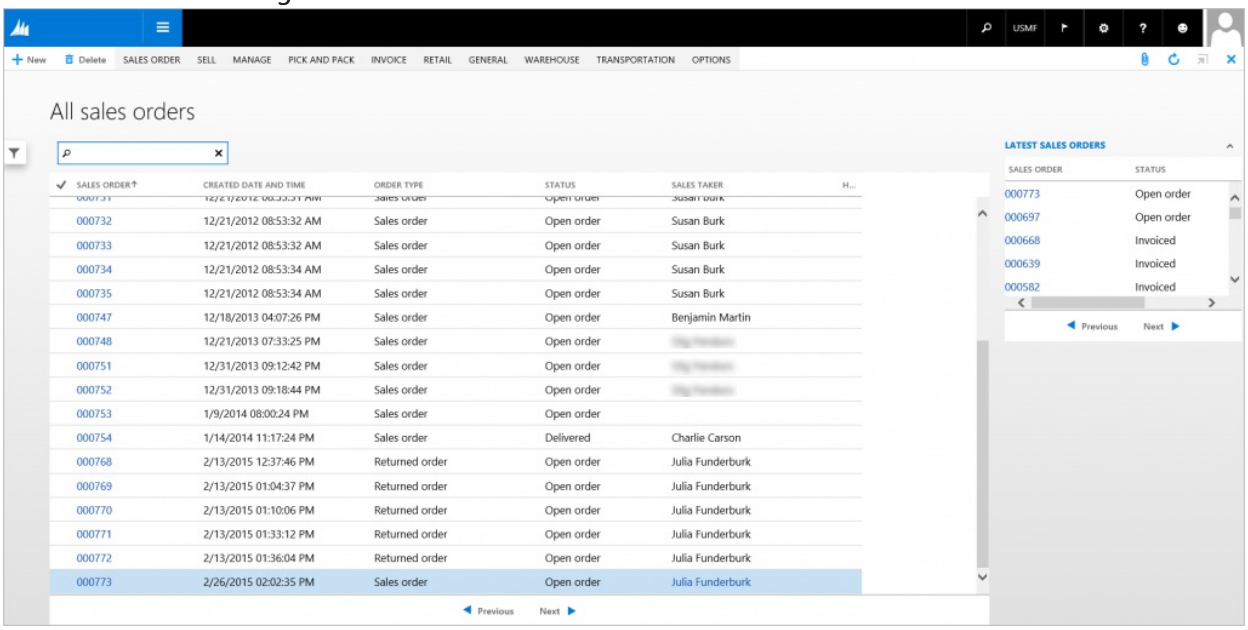
- Have fewer than 15 fields in the grid.
- The first textual/data column should be displayed as a link that goes to the appropriate details form. To do this, make sure that the grid has a default action to enable the hyperlink for the first column.
- A Quick Filter should appear above the list. By default, the QuickFilter should use the most likely field for a filter scenario.
- There should not be any duplicate **New** and **Delete** buttons.
- A link to the List page should be provided in the Main Menu.
- Focus should be in the Quick Filter when the list page is opened.
- **Page title area**:
  - The page title should be in a plural form.
  - For primary list pages, the title should be the name of the entity.
  - For secondary list pages, the title should reflect an activity or status.
- **Grid**:

- For transactional entities, the **ID** field should be the first column, followed by the master entity **ID** and **Name** fields.
- For master entities, the **Name** field should be the first column, followed by the **ID** field.

- **ActionPane** guidelines have been consolidated into the Dynamics AX General Form Guidelines document in the ActionPane guidelines section.
- **FactBox** guidelines have been consolidated into the FactBox Form Patterns document.

Examples

Form: **SalesTableListPage**



Appendix

Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

- **What do I do with the Preview pane when I migrate the form?**

  You can do one of the following:

  - Remove the **Preview** pane altogether if it no longer makes sense.
  - Remove the large header, and leave the **Preview** pane as is.
  - Split the **Preview** pane into multiple logical FactBoxes if the current one is too tall.
    - For a transaction preview, the lines would go into their own FactBox and should be limited to five lines.
    - For a transaction preview, rework the lines into a FactBox card pattern, where the lines are summarized into a count and a lines grid is shown in an enhanced preview when the user hovers over the count value.

Open issues

- **How to handle secondary list pages**
  - Stay in navigation (no app changes needed).
  - Create role-tailored views (after future framework support is added).

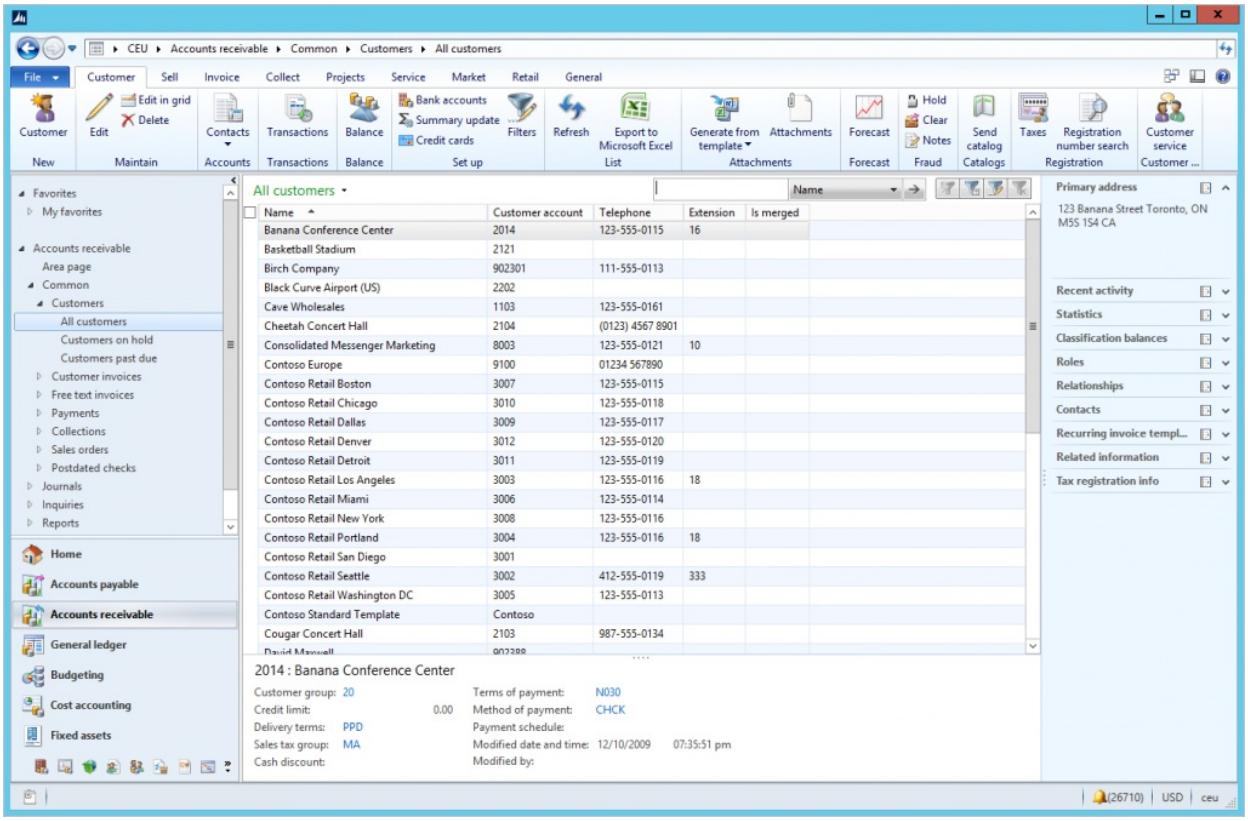
AX 2012 content

AX 2012 links

- MSDN AX 2012 List Page User Experience Guidelines

- MSDN AX 2012 List Page Forms

AX 2012 example



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Simple Details form pattern

2/18/2021 • 2 minutes to read • Edit Online

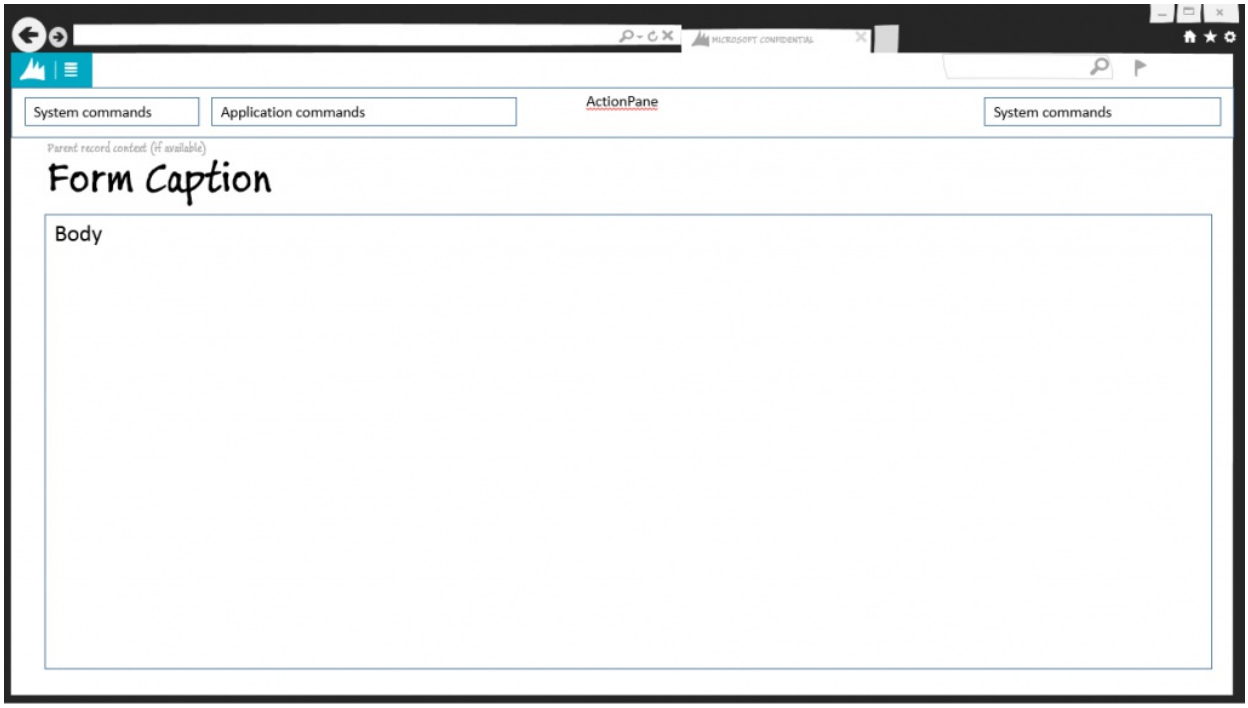
This article describes the Simple Details form pattern. This pattern is used when only a simple set of fields must be presented to the user.

Usage

The Simple Details pattern is used when only a simple set of fields must be presented to the user. Examples include the display of totals and customer balances. Typically, view mode is used for the Simple Details pattern. However, in cases where the form provides editable information, the edit mode should be synced to the parent form. Four patterns are described in this document:

- **Simple Details w/Toolbar and Fields** – This is the basic Simple Details pattern, in which several fields are displayed in the form. The fields can optionally appear inside Groups.
- **Simple Details w/Fast Tabs** – This is the Simple Details pattern that should be used when fields are organized into FastTabs.
- **Simple Details w/Standard Tabs** – This is the Simple Details pattern that should be used when fields are organized into traditional tabs.
- **Simple Details w/Panorama** – This is the Simple Details pattern that should be used when information is intended to be displayed in a panorama format.

Wireframe



Pattern changes

There are no planned changes for the use of this pattern in the current version of Microsoft Dynamics AX.

Model

Simple Details w/Toolbar and Fields – High-level structure

- Design
  - ActionPane (ActionPane)
  - Body (Group) – **Note:** A field subpattern is used.

Simple Details w/FastTabs – High-level structure

- Design
  - ActionPane (ActionPane)
  - *HeaderGroup (Group) [Optional]*
  - Body (Tab, Style=FastTabs)
    - BodyTabPages (TabPage repeats 1..N)
  - *FooterGroup (Group) [Optional]*

Simple Details w/Standard Tabs – High-level structure

- Design
  - ActionPane (ActionPane)
  - *HeaderGroup (Group) [Optional]*
  - Body (Tab, Style=Tabs)
    - BodyTabPages (TabPage repeats 1..N)
  - *FooterGroup (Group) [Optional]*

Simple Details w/Panorama – High-level structure

- Design
  - ActionPane (ActionPane)
  - Body (Tab, Style=Panorama)
    - BodyTabPages (TabPage repeats 1..N)
  - *FooterGroup (Group) [Optional]*

Core components

1. Apply the SimpleDetails pattern on **Form.Design**.
2. Address BP Warnings:
   a. **Design.Caption** isn't empty.
   b. The form must be referenced by at least one menu item.
   c. **TabPage.Caption** isn't empty.
   d. MainMenu must not contain menu items that reference a SimpleDetails form.

Commonly used subpatterns

- Fields and Field Groups
- Toolbar and Fields
- Tabular Fields
- Toolbar and List

UX guidelines

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This

checklist doesn't include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps. **Standard form guidelines:**

- Standard form guidelines have been consolidated into the Dynamics AX General Form Guidelines document.

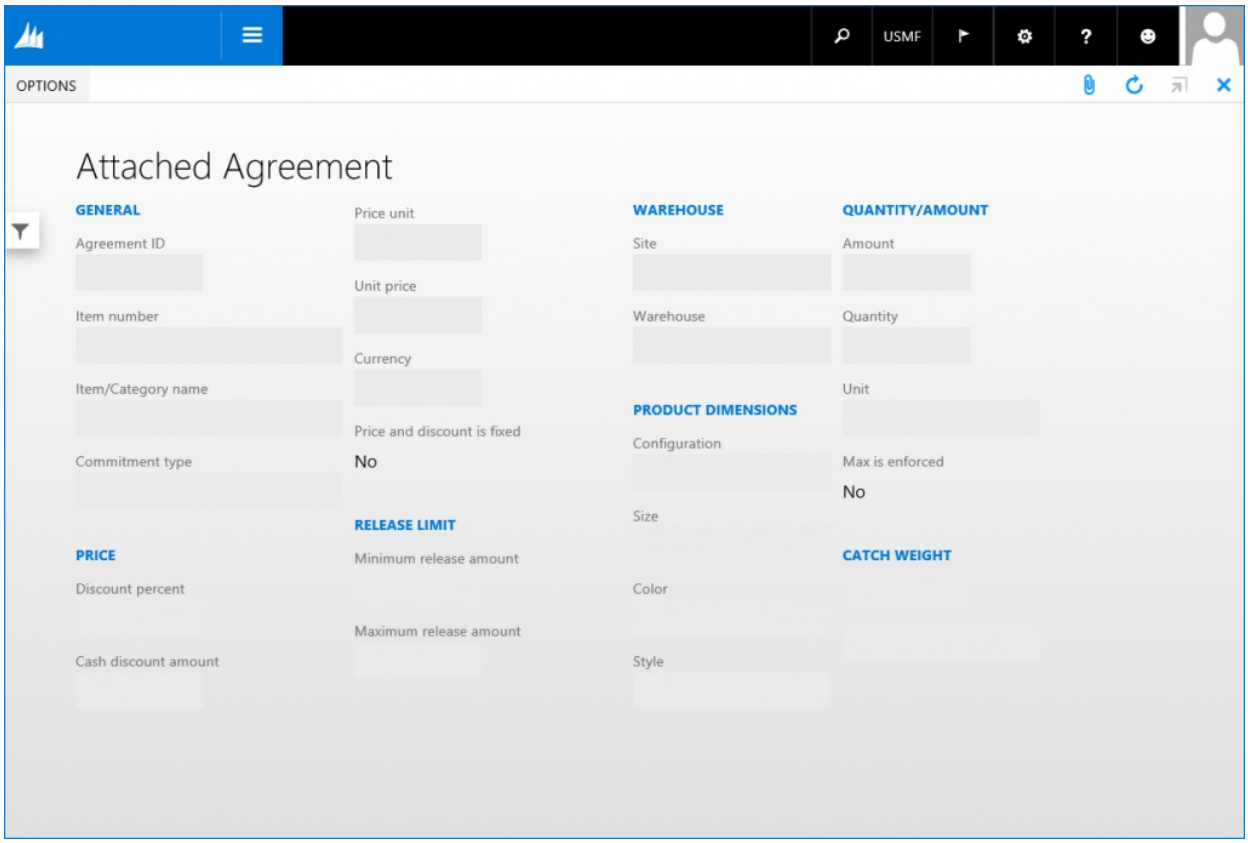
Simple Details guidelines:

- The form page should display a Form Caption that accurately describes the entity.
  - The Form Caption should be in a singular form.

Examples

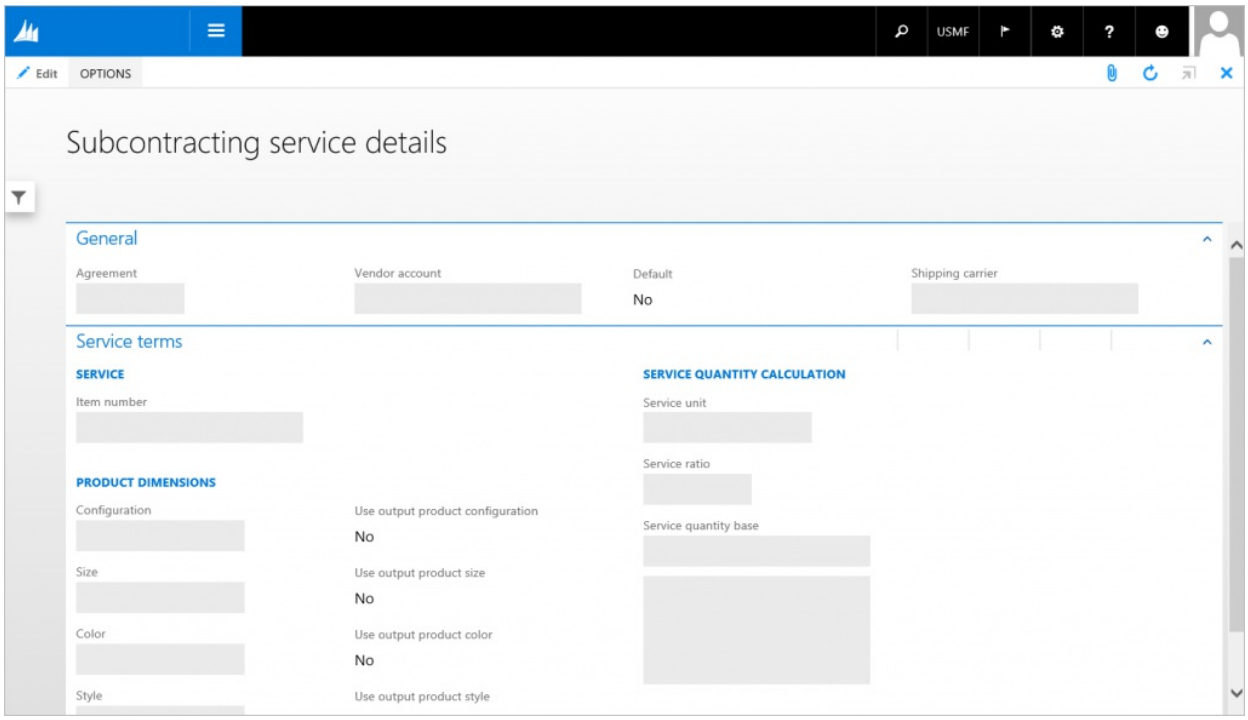
Simple Details w/Toolbar and Fields

Form: **AgreementLine**



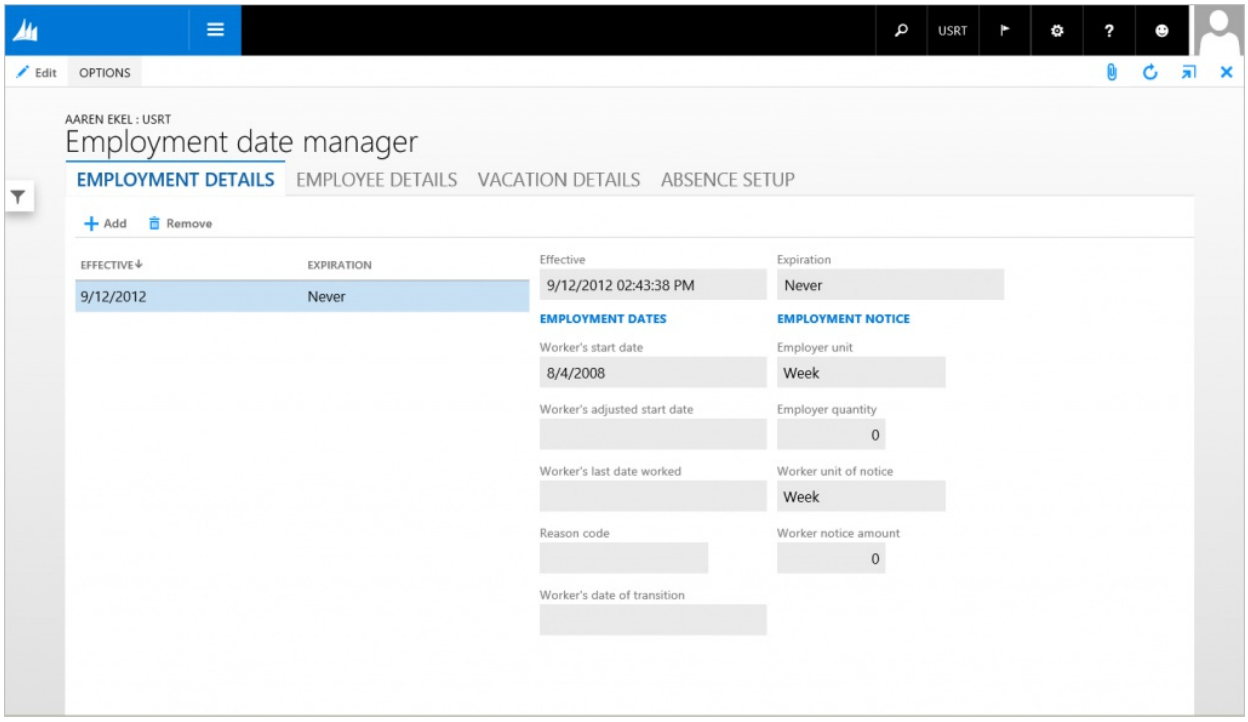
Simple Details w/FastTabs

Form: **PlanActivityServiceDetails**



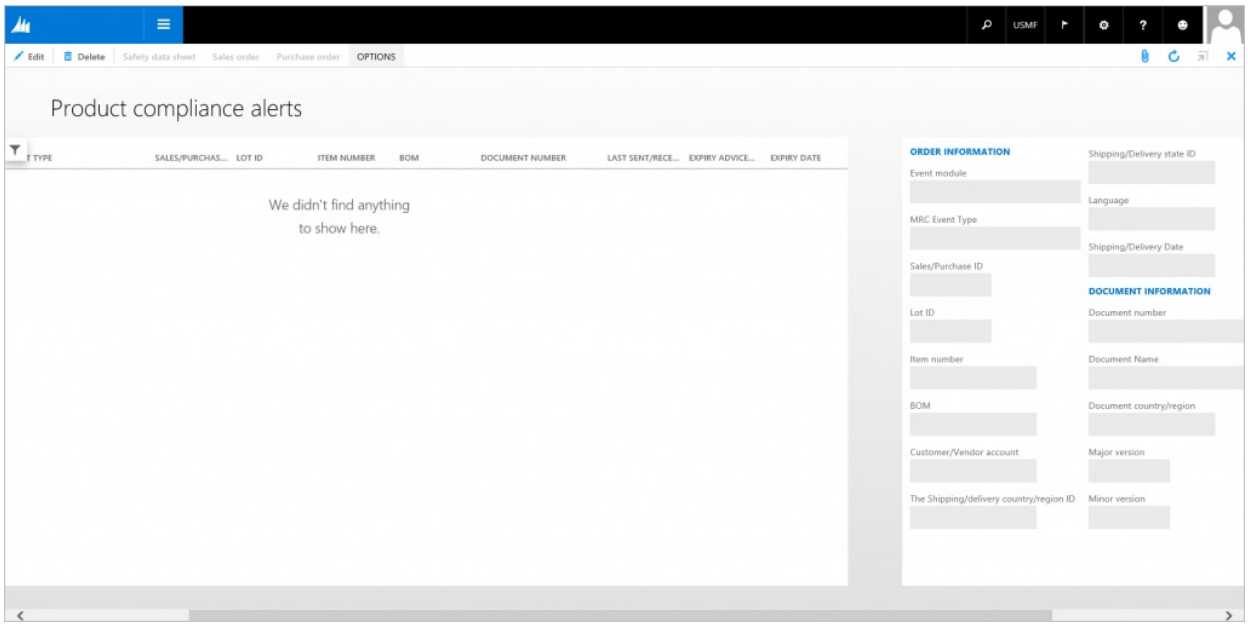
Simple Details w/Standard Tabs

Form: **HcmEmploymentDateManager** (Click **Human Resources** > **Common** > **Workers** > **Workers**, click **General** > **Versions** > **Employment History**, and then click **Date Manager**.)



Simple Details w/Panorama

Form: **PdsMRCEventTracker**



Appendix

Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues

- Investigate whether Simple Details forms that show a small amount of related content should have a different presentation than a full-page form.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Simple List and Details form pattern

2/18/2021 • 5 minutes to read • Edit Online

This topic provides information about the Simple List and Details form pattern. This pattern is used to maintain data for entities of medium complexity.

Usage

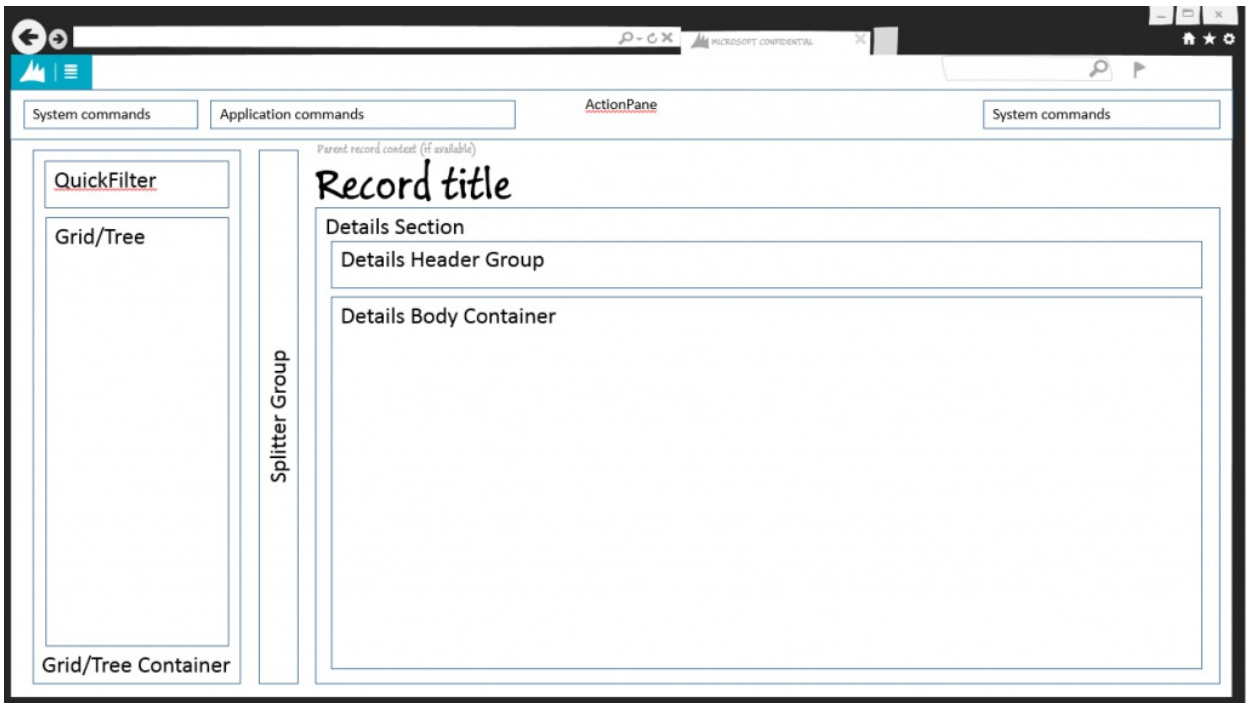
The Simple List and Details (SL+D) pattern is used to maintain data for entities of medium complexity. Entities of medium complexity are those entities that have six or more fields. The Simple List pattern should be used for simple entities that have fewer than six fields. There are some exceptions where entities that have up to 15 fields are still considered simple entities. The Simple List and Details pattern is prescribed when these conditions are met:

- The underlying data has more than six fields.
- There are between zero and five child data collections.

Three patterns are described in this document:

- **Simple List and Details – List Grid** – This is the basic SL+D pattern. This is the pattern that should be used by default.
- **Simple List and Details – Tabular Grid** – This is the SL+D pattern that should be used if the number of fields in the "simple list" part of the form is larger than expected (see the "Pattern changes" section later in this article).
- **Simple List and Details – Tree** – This is the SL+D pattern that should be used if the "simple list" part of the form is actually a tree.

Wireframe



Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- The top ActionPane strip control has been converted to a standard ActionPane.
- **New**, **Delete**, and **Edit** buttons are provided by the framework.
- View mode is used by default.
- A Quick Filter control has been added above the "list" part of the form.
- Whenever possible, use the list-style grid for the "list" part of the form. A tabular grid is an acceptable alternative in some situations, such as when these conditions are met:
  - Multiple fields of the same type (for example, three date fields) would not be distinguishable in the list-style grid.
  - The user's task is to compare a sequence of dates/numbers across rows in the list (for example, date effective dates or route step numbers).
  - The number of fields in the grid is larger than expected (if each row takes up more than three lines in the list-style grid).
- Fields in the header group are arranged horizontally instead of vertically.
- FactBoxes are allowed.
- The form structure has been simplified (the BodyGroup container has been removed).

Model

High-level structure

- Design
  - ActionPane
  - NavigationList (Group)
    - Quick Filter
    - *CustomFilterGroup (Group) [Optional]*
    - ListStyleGrid (Grid) | Tree | TabularGrid (Grid)
  - VerticalSplitter (Group) *[only allowed for Tree or TabularGrid variants]*
  - DetailsHeader (Group)
  - DetailsTab (Tab)

Core components

1. Apply one of the SimpleListDetails patterns on **Form.Design**.
2. Resolve required BP checks:
   a. Set **Design.Caption** the same as the label that is used on the **Name** property of the table.
   b. Set **Design.Datasource** the same as **Grid.Datasource**.
   c. Set the primary data source to **InsertIfEmpty**=**No**.
   d. Set the primary **ActionPane.DataSource** the same as **Grid.Datasource**.
   e. Set **Grid.Datasource** to the primary data source.

Commonly used subpatterns

- Fields and Field Groups
- Toolbar and List
- Toolbar and Fields
- Nested Simple List and Details

UX guidelines

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn't include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps.

Standard form guidelines:

- Standard form guidelines have been consolidated into the Microsoft Dynamics AX General Form Guidelines document.

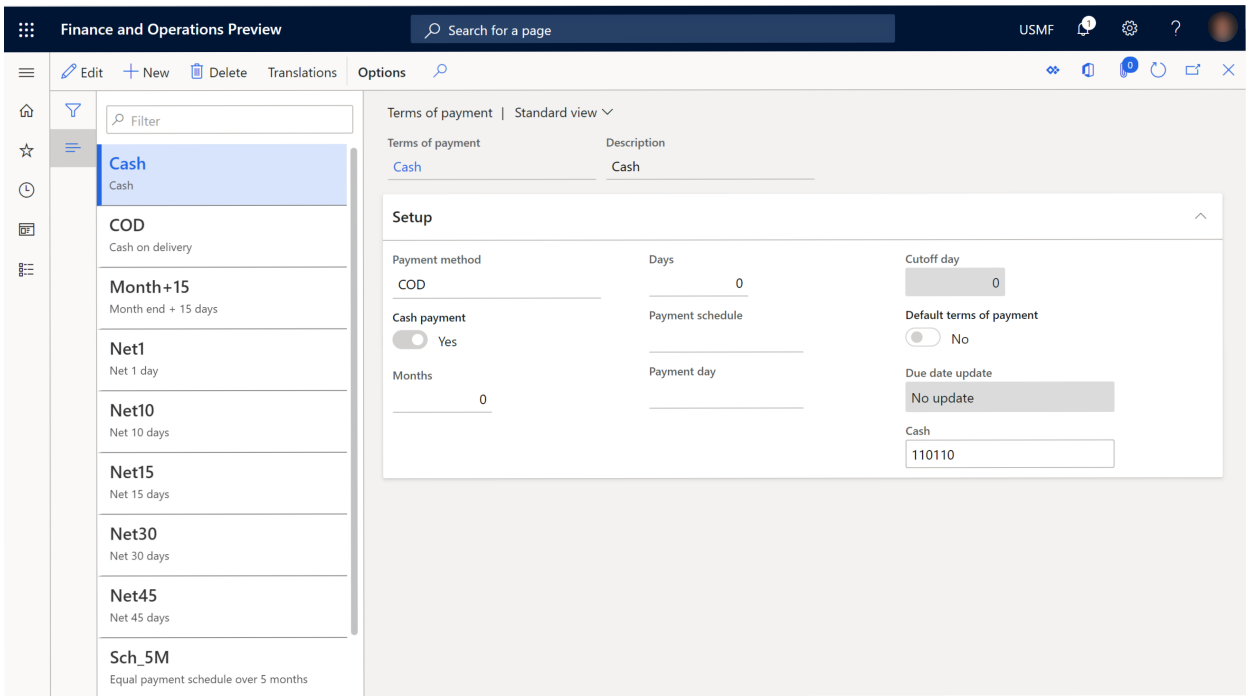
Simple list & detail guidelines:

- The page should display a Form Caption that accurately describes the entity.
  - The Form Caption should be in plural form.
- There should not be duplicate **New** or **Delete** buttons.
- By default, the Quick Filter should use the name or description column.
- Guidelines for custom filters have been consolidated into the Custom Filter Group subpattern document.
  - There should be no more than two custom filter fields in a SL&D form.
- There should be a tabular grid, a list-style grid, or a tree control on the left edge of the form.
  - List-style grids should display no more than three rows (lines) for each record in the List-style grid. Typically, just the ID and Description are sufficient.
  - Between two and five fields should be used for the list on the left.
  - A tabular grid can be used in some unique situations but isn't generally recommended.
    - If a tabular grid is used, it should **not** be editable.
  - When there is no data, the grid or tree control should not automatically add a new record.
- A **Details** section should be displayed on the right of the form:
  - The list fields (whether they are from a list, tabular grid, or tree) should be the first fields in Header Group. They should appear in the same order that they appear in the grid or tree, so that the user can edit and see the labels of the fields.
- Simple List and Detail forms must **not** have these elements:
  - Standard tabs to group fields

Examples

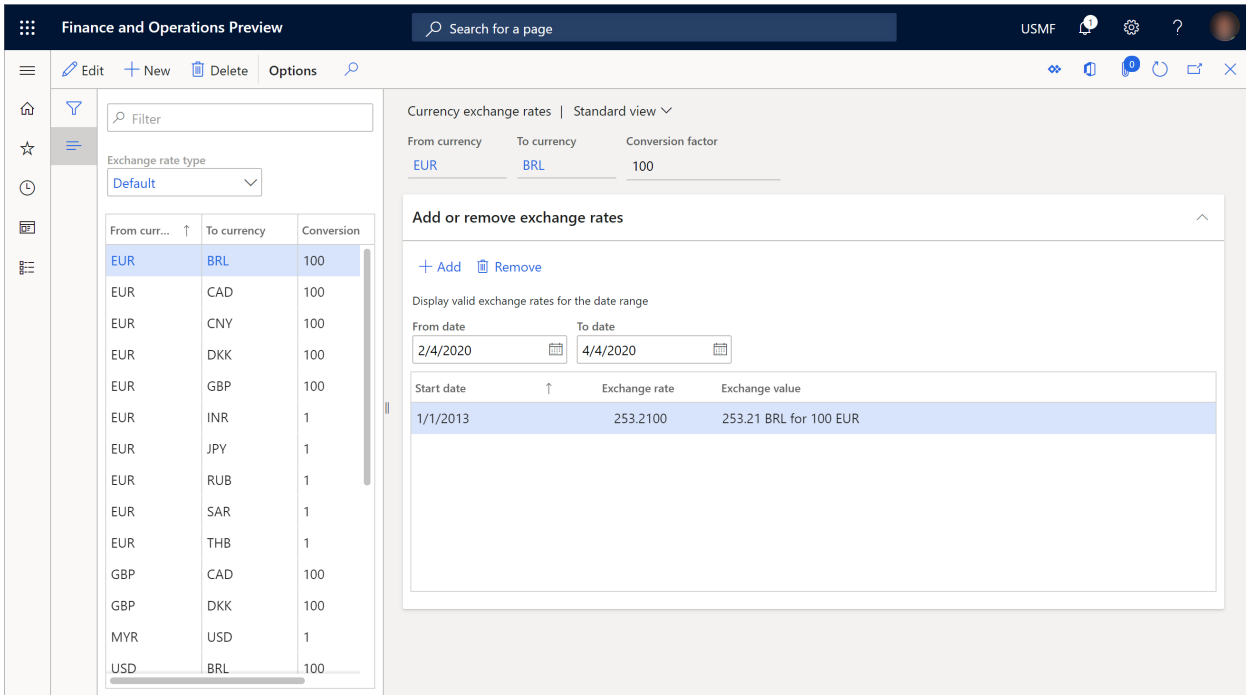
Simple List and Details – List Grid

Form: **PaymTerm**



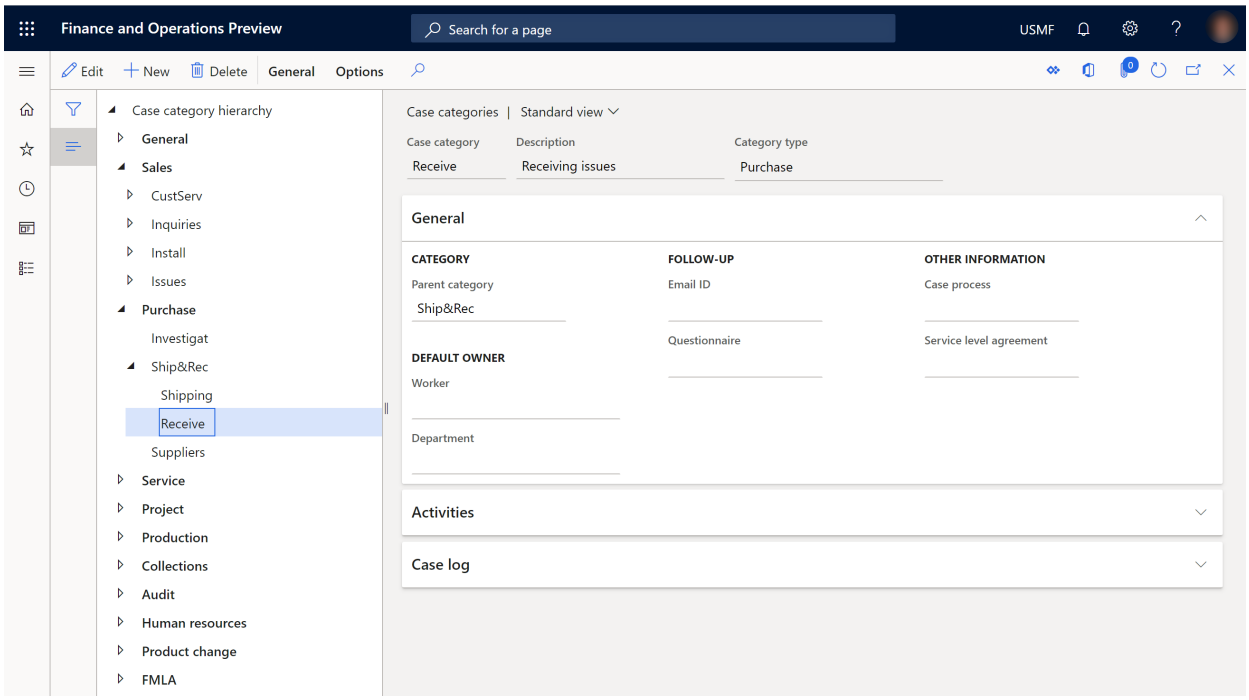
Simple List and Details – Tabular Grid

Form: **ExchangeRate**



Simple List and Details – Tree

Form: **CaseCategorySetup**



Appendix

Frequently asked questions

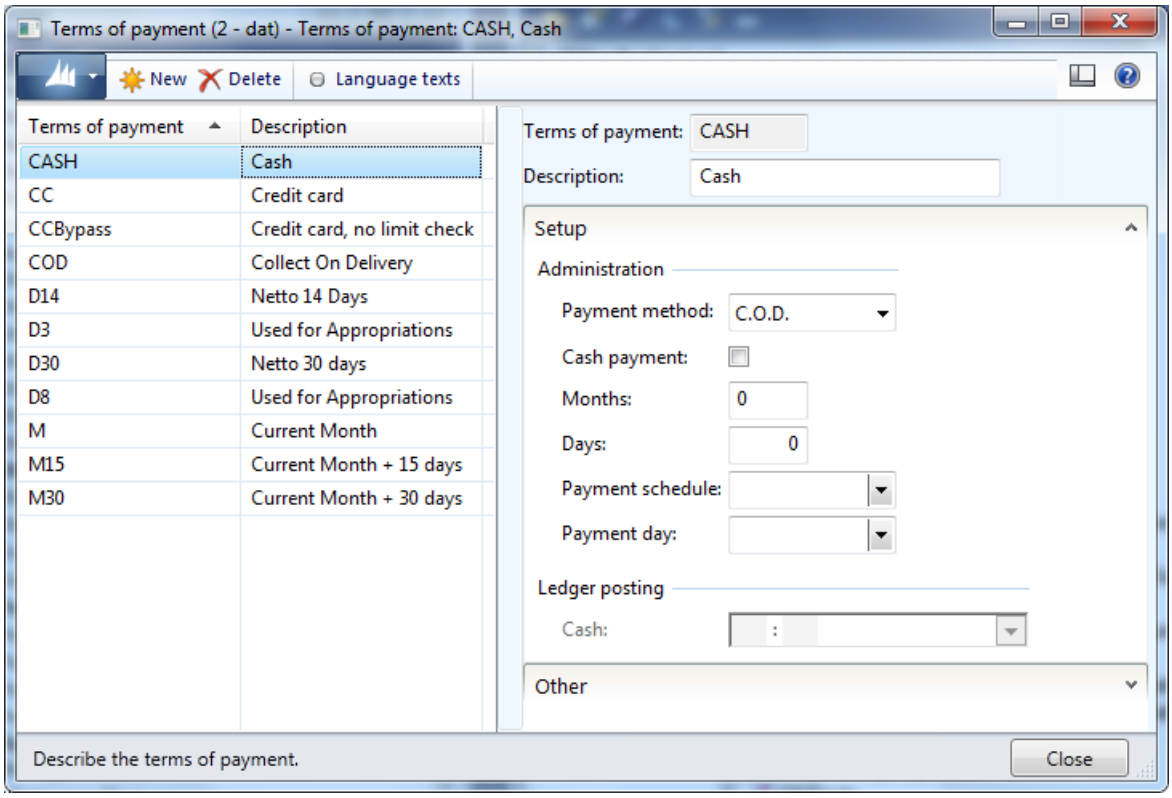
This section will have answers to frequently asked questions that are related to this guideline/pattern.

- **When do I use icons on actions in the toolbars?**
  - See the Button Image Guidelines in the General Form Guidelines document.

Open issues

- **How can a developer move between the ListStyleGrid and the TabularGrid patterns?**
  - Currently, developers must manually move between the patterns.
- **Are we going to allow modeling without FastTabs in the details body?**
  - Although we require FastTabs in the details body, we plan to eventually hide the FastTab header if only one FastTab is visible.
- **How do we allow for exceptions to the FastTab rule for legacy situations such as the Interest form?**
  - Whenever possible, refactor the form to fit the SL&D pattern (as the **Interest** form has done). Otherwise, use custom containers.
- **How do we prevent hyperlinks on fields in the UI?**
  - For some fields, you can **set IgnoreEDTRelation=Yes** to prevent hyperlinks in the UI. Regardless (as of Platform update 17), you can set **EnableFormRef=No** on an input control to disable a hyperlink.

AX 2012 content



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Simple List form pattern

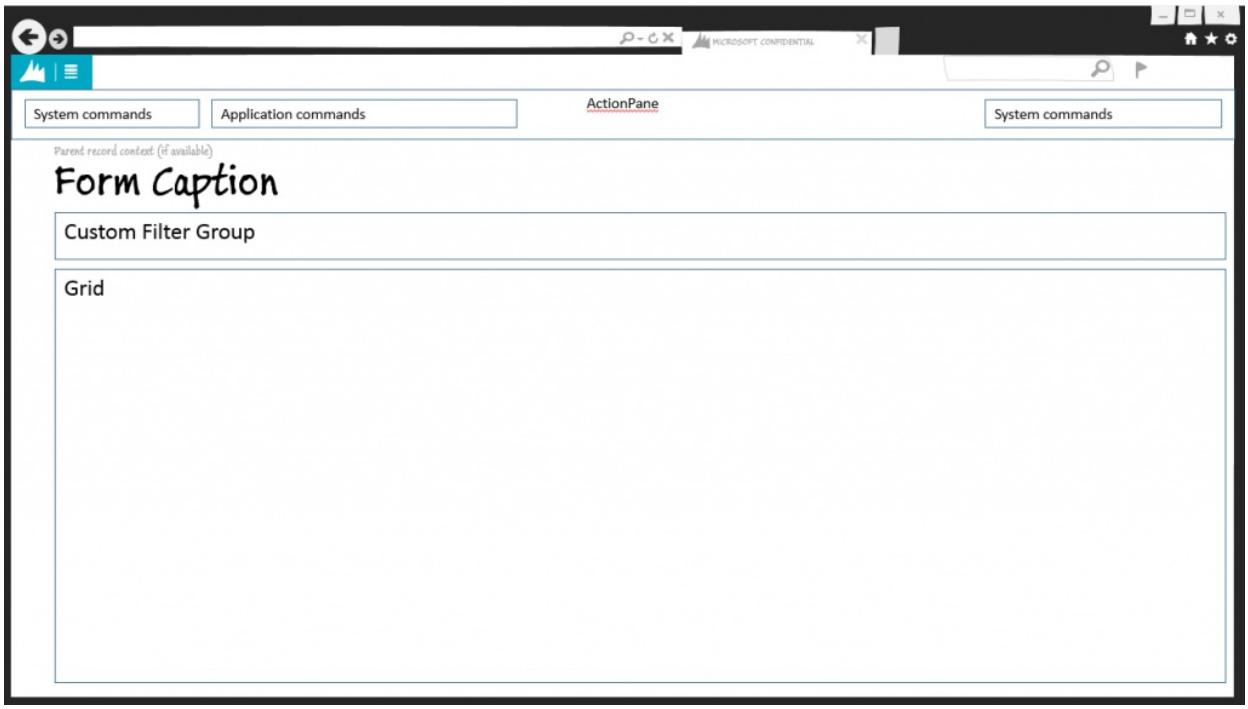
2/18/2021 • 2 minutes to read • Edit Online

This article provides information about the Simple List form pattern. This pattern is used to maintain data for simple entities.

Usage

The Simple List pattern is used to maintain data for simple entities. Simple entities are entities that have six or fewer fields and no parent/child relationships. There are some exceptions where entities that have up to 15 fields are still considered simple entities.

Wireframe



Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- The top ActionPane strip control has been converted to a standard ActionPane.
- **New**, **Delete**, and **Edit** buttons are provided by the framework.
- View mode is used by default.
- A Quick Filter has been added above the grid.
- When the form is used as a dependent form, the parent form record context is automatically shown above the form caption.
  - The page title group for dependent form usage was removed, because it will be provided by the framework.
- The pattern allows for multiple selections in the grid.

Model

High-level structure

- Design
  - ActionPane (ActionPane)
  - Custom Filter (Group)
    - Quick Filter (Quick Filter)
    - *OtherFilters ($Field) [0..N]*
  - TabularGrid (Grid)
  - *Footer (Group) [Optional]*

Core components

1. Apply the SimpleList pattern on **Form.Design**.
2. Address BP Warnings:
   a. **Design.Caption** isn't empty.
   b. **Design.DataSource** isn't empty.
   c. **Grid.Datasource** must be set.
   d. The form must be referenced by at least one menu item.
   e. **Design.Datasource** is set the same as **Grid.Datasource**.
   f. The primary key field of the primary data source's table has **IgnoreEDTRelation**=**Yes**.
   g. The grid must not contain more than 15 fields.

Commonly used subpatterns

- Custom Filter Group

UX guidelines

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn't include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps.

Standard form guidelines:

- Standard form guidelines have been consolidated into the Microsoft Dynamics AX General Form Guidelines document.

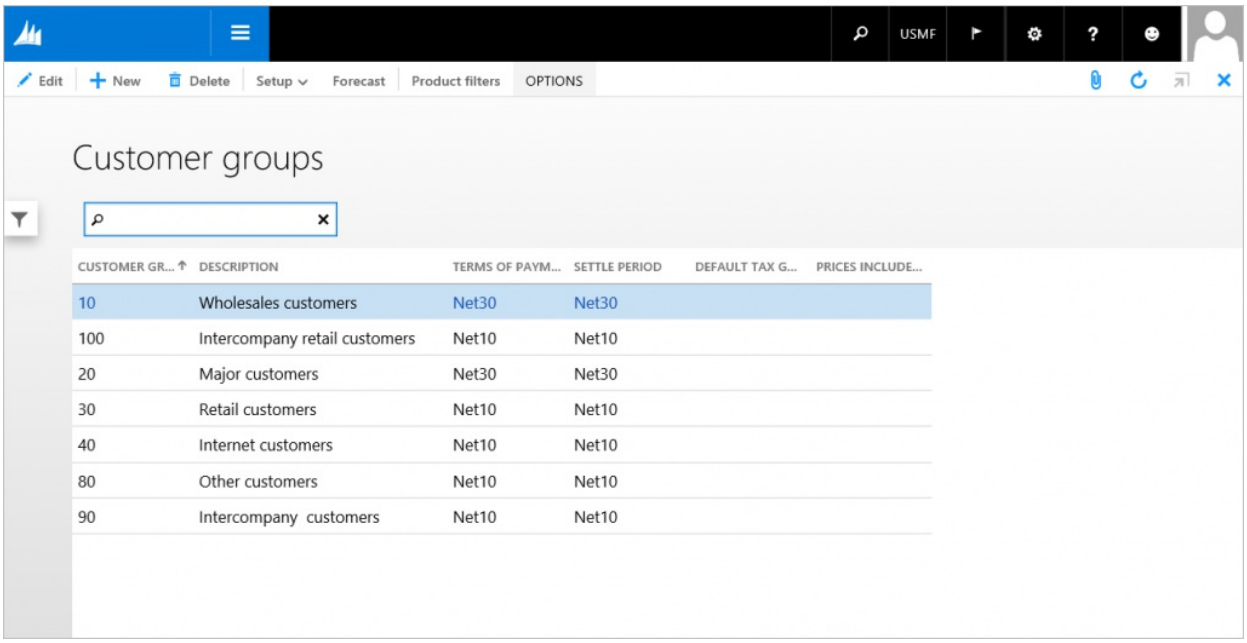
Simple list guidelines:

- By default, the Quick Filter should use the name or description column.
- The list can display up to 15 columns.

  Note: This guideline has been relaxed from AX 2012.
- There should not be any duplicate **New** or **Delete** buttons.
- The page title should be in a plural form.
- When there is no data, the grid should not automatically add a new record.

Examples

Form: **CustGroup**



Note: We plan to extend the grid lines to the right and bottom edges in a future client deliverable.

Appendix

Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues

None at this time.

AX 2012 content

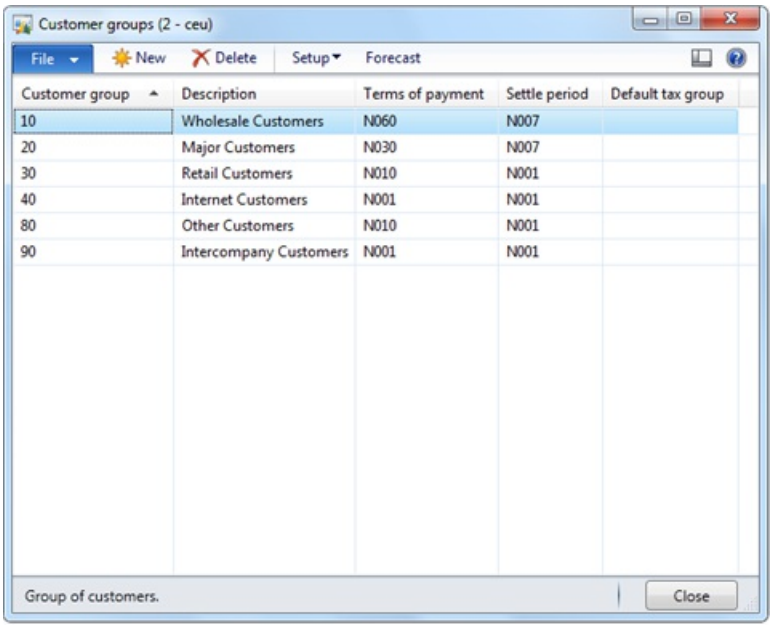


Table of Contents form pattern

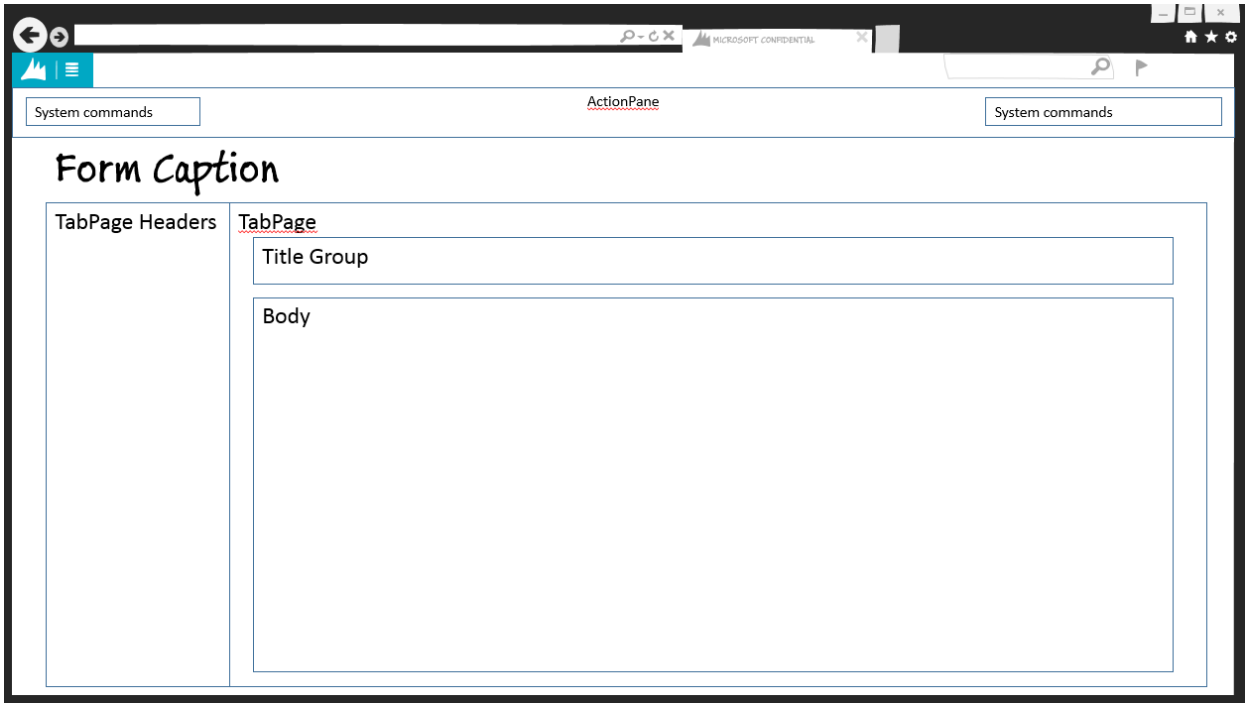
2/18/2021 • 3 minutes to read • Edit Online

This article provides information about the Table of Contents form pattern. This pattern should be used when two or more logically related forms are required for setup configuration.

Usage

The Table of Contents pattern should be used when two or more logically related forms are required for setup configuration. The vertical arrangement of tabs implies the order of completion. This form pattern is also used for collections of unrelated items, such as tab pages that have a different root entity per tab. This form pattern contains a collection of smaller content regions, each of which follows a container subpattern such as Toolbar and List, Nested Simple List and Details, or Fields and Field Groups.

Wireframe



Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- The Content Body child container uses dynamic columns for a responsive layout.
- An optional secondary instruction has been added under the Title Group.

Model

High-level structure

- Design
  - Tab (Style=VerticalTabs)
    - TabPage *[repeats 1..N times]*

    - Title (Group)
        - MainInstruction (StaticText)
        - *SecondaryInstruction (StaticText) [Optional]*
    - Body (Group) | FastTabContent (Tab)

Core components

- Apply the TableOfContents pattern on **Form.Design**.
- Address BP Warnings:
    - **Design.Caption** isn't empty.
    - The form must be referenced by at least one menu item.
    - **TabPage.Caption** isn't empty.
    - **TabPage.DataSource** isn't empty.
    - **StaticText.Text** isn't empty.

Commonly used subpatterns

Each BodyGroup will use one of the following container patterns for the content in the Table of Contents section:

- Fields and Field Groups
- Toolbar and List
- Toolbar and Fields
- Nested Simple List and Details
- Tabular Fields
- List Panel

UX guidelines

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn't include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps.

Standard form guidelines:

- Standard form guidelines have been consolidated into the Microsoft Dynamics AX General Form Guidelines document.

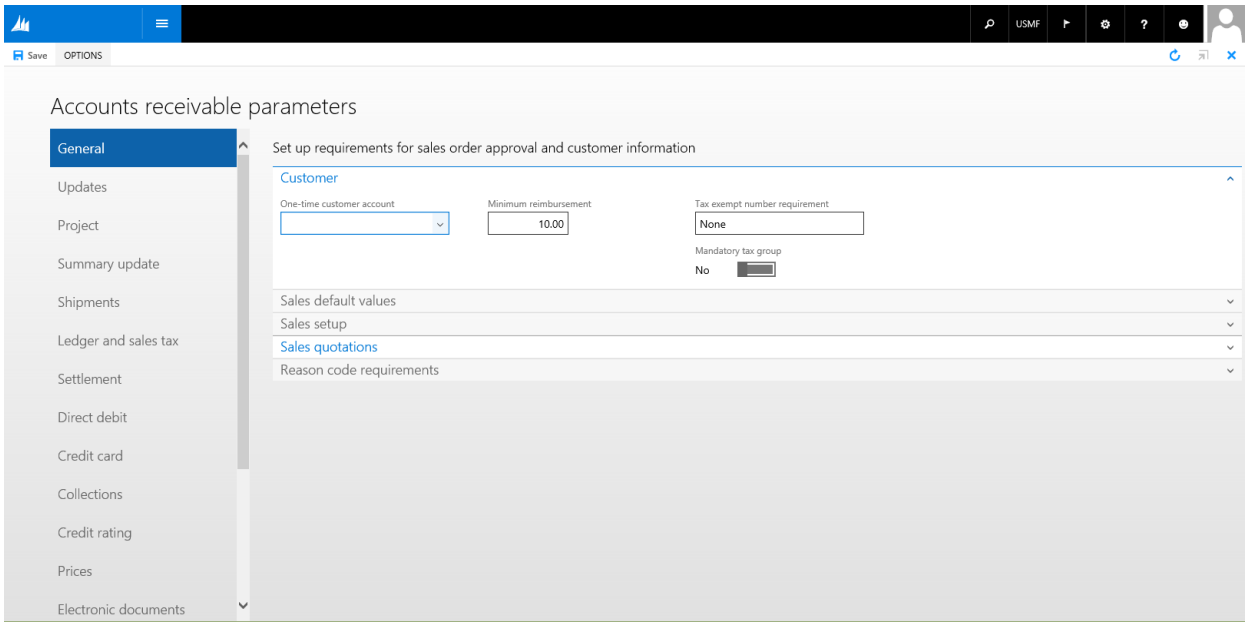
Table of contents guidelines:

- The supplemental instruction, if it's shown, is composed of a complete, concise sentence in sentence case and has end punctuation.
- TOC tabs should appear in the same sequence that is typically used to enter information.
- The first tab in the list should be highlighted when the form is opened, unless the form is opened in the context of a specific task from another form.
- The **content area** for the TOC content should primarily be one of three patterns: Simple List, Simple List and Details, or Simple Details.
    - Simple List content should follow the subpattern guidelines.
    - Simple List and Details content should follow the Nested Simple List and Details subpattern guidelines.
    - Simple Details content should follow the Toolbar and Fields subpattern guidelines.
    - FastTabs should follow the FastTab guidelines in the Dynamics AX General Form Guidelines document.
    - Actions appearing on a Toolbar on a tab page.
- A TOC form should **not** have the following:

- Application actions on a standard ActionPane. (It should have only framework actions.)
- FactBoxes.
- Standard tabs on a TOC tab page.

Examples

Form: **CustParameters**



Appendix

Frequently asked questions

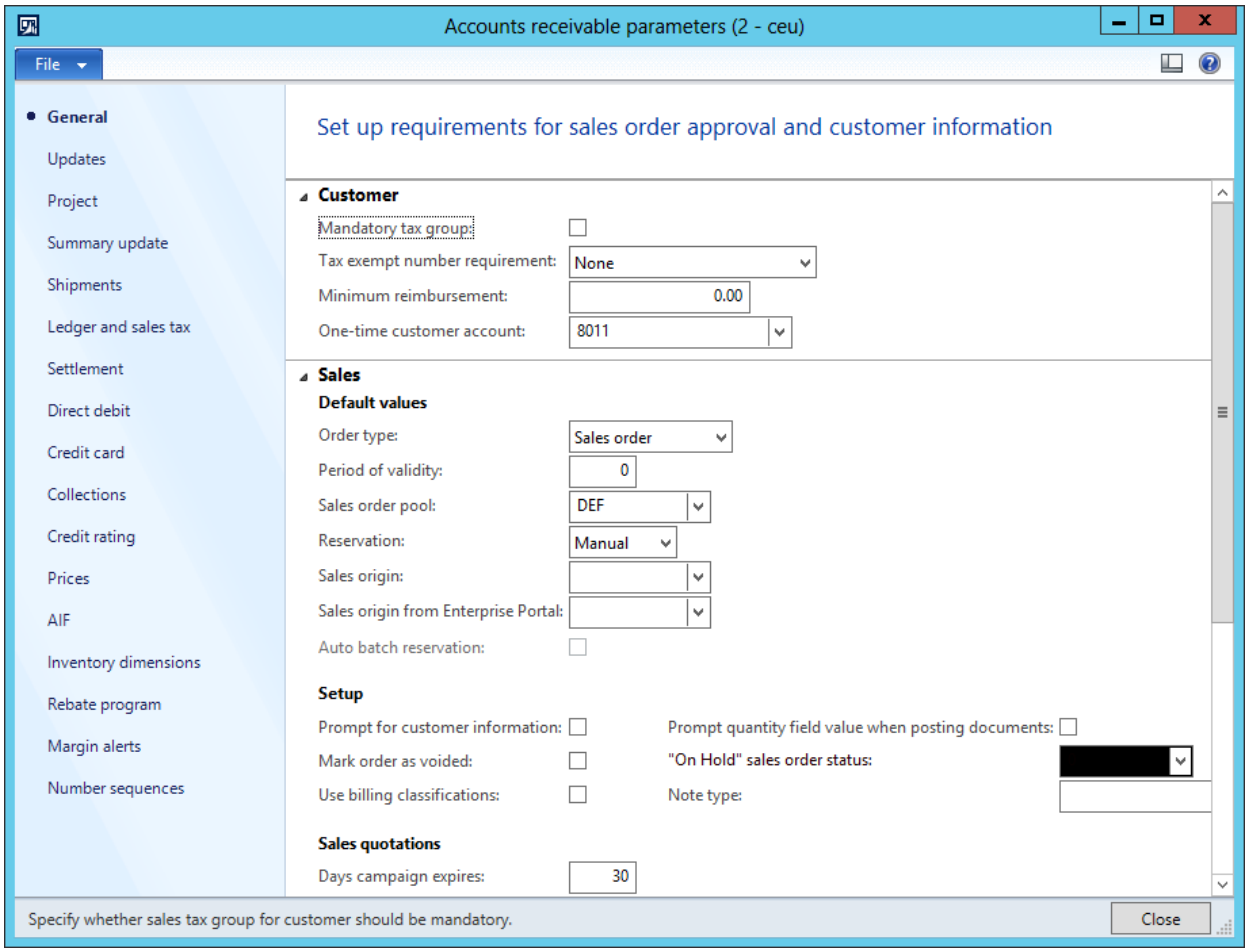
This section will have answers to frequently asked questions that are related to this guideline/pattern.

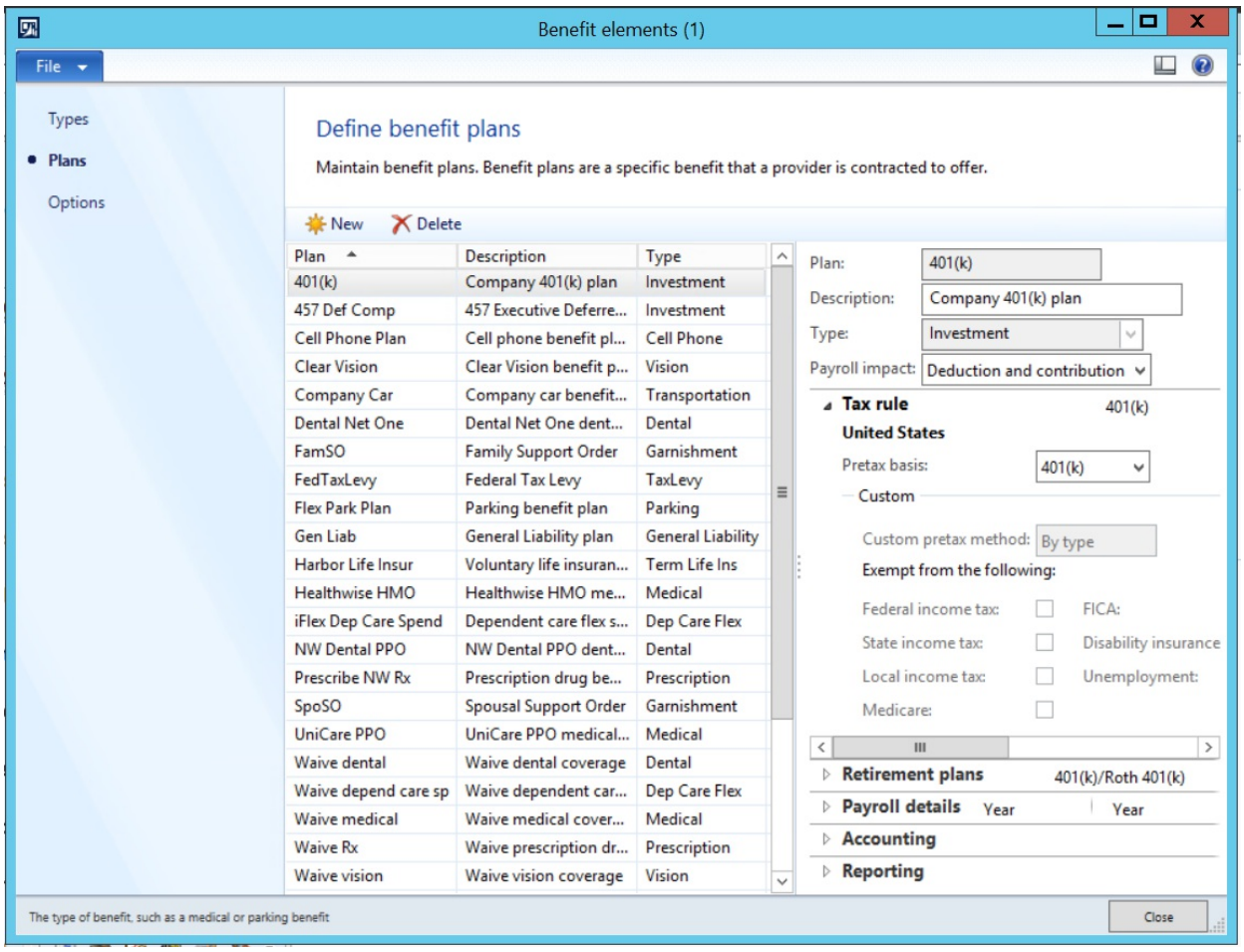
- **What do I do with 'Global' buttons?**
  - There have been several cases where a button is required in order to initialize data or sync information between services. Because we allow only system buttons on the standard Action Pane in this pattern, we recommend that these buttons go in one of two places:
    - On the tab page that the action is most closely related to.
    - If a place doesn't exist, on a toolbar on the first tab page of the pattern.

Open issues

- None

AX 2012 content


Accounts receivable parameters (2 - ceu)
File
General
Updates
Project
Summary update
Shipments
Ledger and sales tax
Settlement
Direct debit
Credit card
Collections
Credit rating
Prices
AIF
Inventory dimensions
Rebate program
Margin alerts
Number sequences
Set up requirements for sales order approval and customer information
Customer
Mandatory tax group:
Tax exempt number requirement:
None
Minimum reimbursement:
0.00
One-time customer account:
8011
Sales
Default values
Order type:
Sales order
Period of validity:
0
Sales order pool:
DEF
Reservation:
Manual
Sales origin:
Sales origin from Enterprise Portal:
Auto batch reservation:
Setup
Prompt for customer information:
Prompt quantity field value when posting documents:
Mark order as voided:
"On Hold" sales order status:
Use billing classifications:
Note type:
Sales quotations
Days campaign expires:
30
Specify whether sales tax group for customer should be mandatory.
Close


Benefit elements (1)
File
Types
Plans
Options
Define benefit plans
Maintain benefit plans. Benefit plans are a specific benefit that a provider is contracted to offer.
New
Delete
Plan
Description
Type
401(k) Company 401(k) plan Investment
457 Def Comp 457 Executive Deferre... Investment
Cell Phone Plan Cell phone benefit pl... Cell Phone
Clear Vision Clear Vision benefit p... Vision
Company Car Company car benefit... Transportation
Dental Net One Dental Net One dent... Dental
FamSO Family Support Order Garnishment
FedTaxLevy Federal Tax Levy TaxLevy
Flex Park Plan Parking benefit plan Parking
Gen Liab General Liability plan General Liability
Harbor Life Insur Voluntary life insuran... Term Life Ins
Healthwise HMO Healthwise HMO me... Medical
iFlex Dep Care Spend Dependent care flex s... Dep Care Flex
NW Dental PPO NW Dental PPO dent... Dental
Prescribe NW Rx Prescription drug be... Prescription
SpoSO Spousal Support Order Garnishment
UniCare PPO UniCare PPO medical... Medical
Waive dental Waive dental coverage Dental
Waive depend care sp Waive dependent car... Dep Care Flex
Waive medical Waive medical cover... Medical
Waive Rx Waive prescription dr... Prescription
Waive vision Waive vision coverage Vision
Plan: 401(k)
Description: Company 401(k) plan
Type: Investment
Payroll impact: Deduction and contribution
Tax rule 401(k)
United States
Pretax basis: 401(k)
Custom
Custom pretax method: By type
Exempt from the following:
Federal income tax:
FICA:
State income tax:
Disability insurance
Local income tax:
Unemployment:
Medicare:
Retirement plans 401(k)/Roth 401(k)
Payroll details Year Year
Accounting
Reporting
The type of benefit, such as a medical or parking benefit
Close

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Task Double form pattern

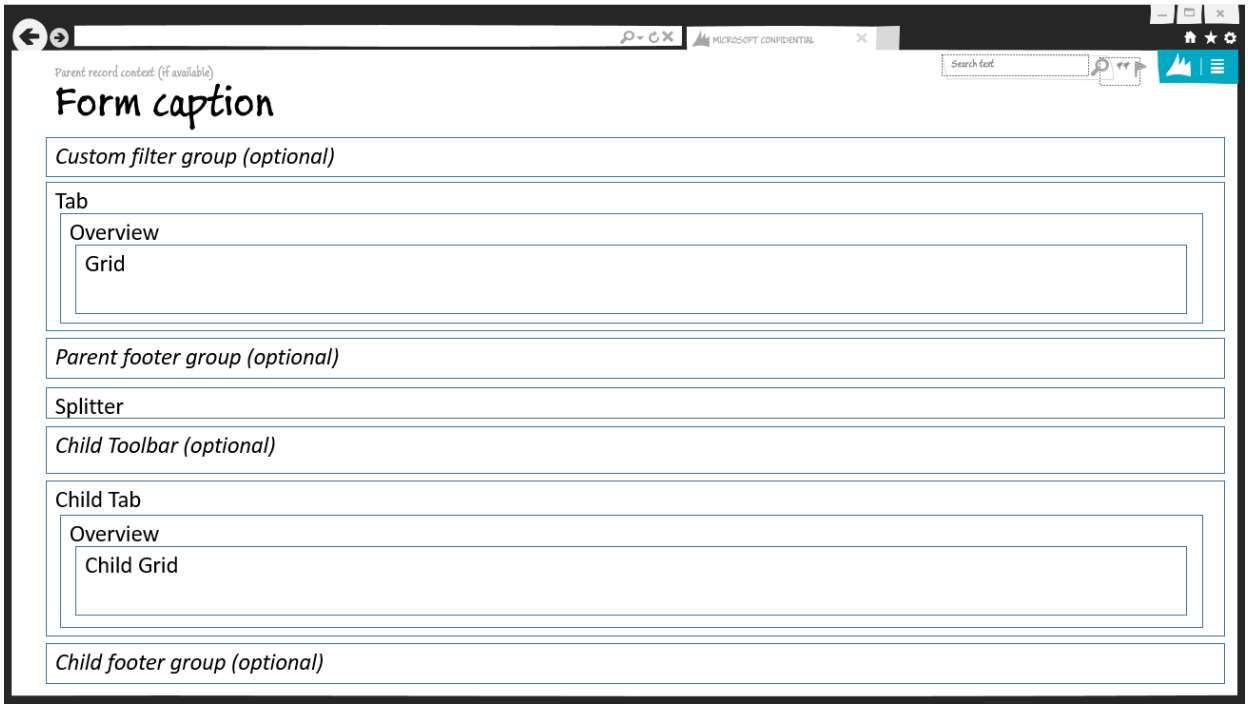
2/18/2021 • 2 minutes to read • Edit Online

This article provides information about the Task Double form pattern. This pattern was previously used to present a parent and child entity in the same form.

Usage

This type of form has previously been used when you wanted to present parent/child entities in the same form. This isn't a recommended pattern for new forms. No new forms should be created that use this pattern. This pattern will provide structure and stability for legacy forms, and will also provide a migration path to more modern form patterns.

Wireframe



]

Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- The form opens in view mode.
- The top ActionPane strip control has been converted to a standard ActionPane.
- The **Overview** label on the parent tab has been changed to **List**.
- The contents of the tab container use dynamic columns for a responsive layout.
- The label for the child tab's list should be **<x> list**, where **<x>** is replaced by an appropriate string, based on the entity. For example, if the child entity is usually called Charges, the label for the tab should be **Charges list**.
  - Exception: If the child entity is "lines" of some sort, the word "list" should not be added to the end.

Model

High-level structure

- Design
  - ActionPane (Action Pane)
  - *CustomFilter (Group) [Optional]*
  - ParentTab (Tab)
    - ParentList (TabPage) – **Note:** The Toolbar and List subpattern is used.
    - General (TabPage repeats 0..N)
  - *ParentFooterGroup (Group) [Optional]*
  - HSplitter (Group)
  - *ChildToolbar (ActionPane) [Optional]*
  - ChildTab (Tab)
    - ChildList (TabPage) – **Note:** The Toolbar and List subpattern is used.
    - General (TabPage, repeats 0..N)
  - *ChildFooterGroup (Group) [Optional]*

Core components

1. Apply the Task Double pattern on **Form.Design**.
2. Address BP Warnings:
   a. **Design.Caption** isn't empty.
   b. The form must be referenced by at least one menu item.
   c. **TabPage.Caption** isn't empty.
   d. **TabPage.DataSource** isn't empty.
   e. **StaticText.Text** isn't empty.

Related patterns

- Task Single

Commonly used subpatterns

- Custom Filter Group
- Fields and Field Groups
- Toolbar and List
- Toolbar and Fields

UX guidelines

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn't include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps.

Standard form guidelines:

- Standard form guidelines have been consolidated into the Microsoft Dynamics AX General Form Guidelines document.

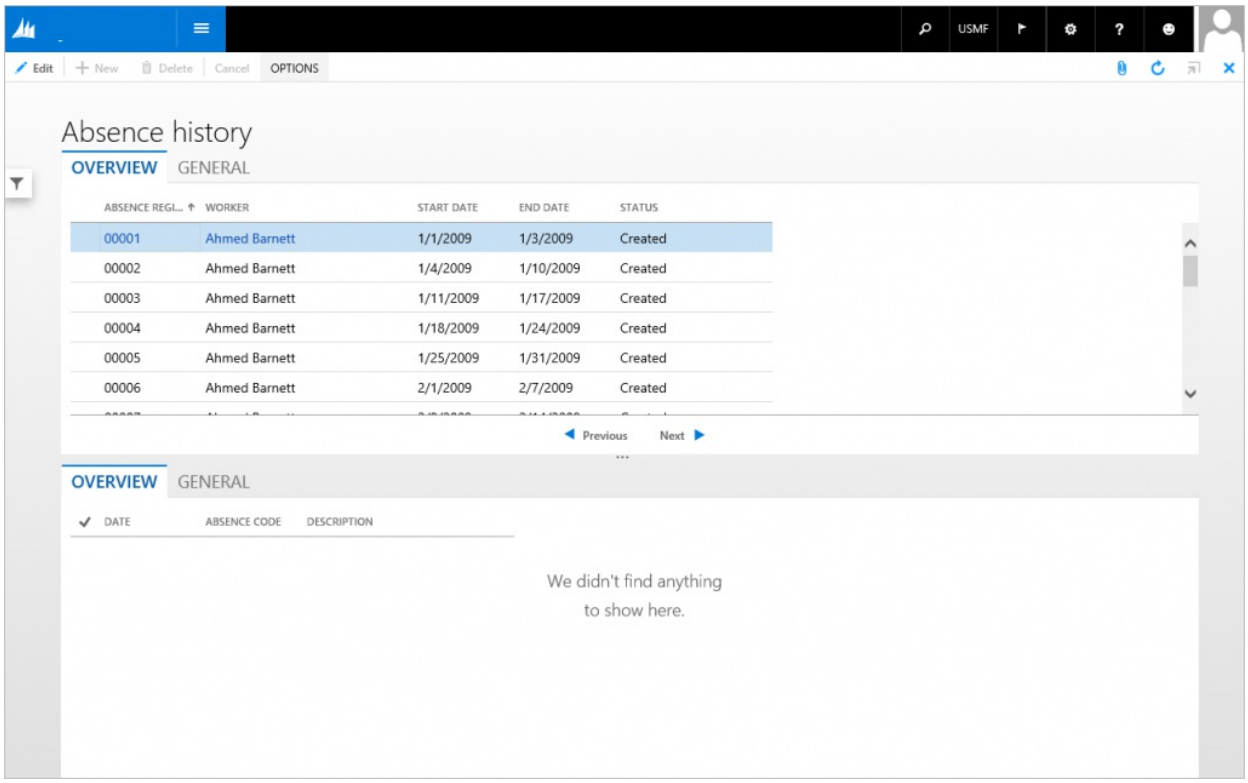
Task Double guidelines:

- The **Overview** tab is the first tab and is active when the form is opened.
- The first tab on a child tab control should be called **Lines list** or an appropriate variation.

- Selection in the parent grid will update content in the child grid.

Example

Form: **HRMAbsenceTableHistory**



Appendix

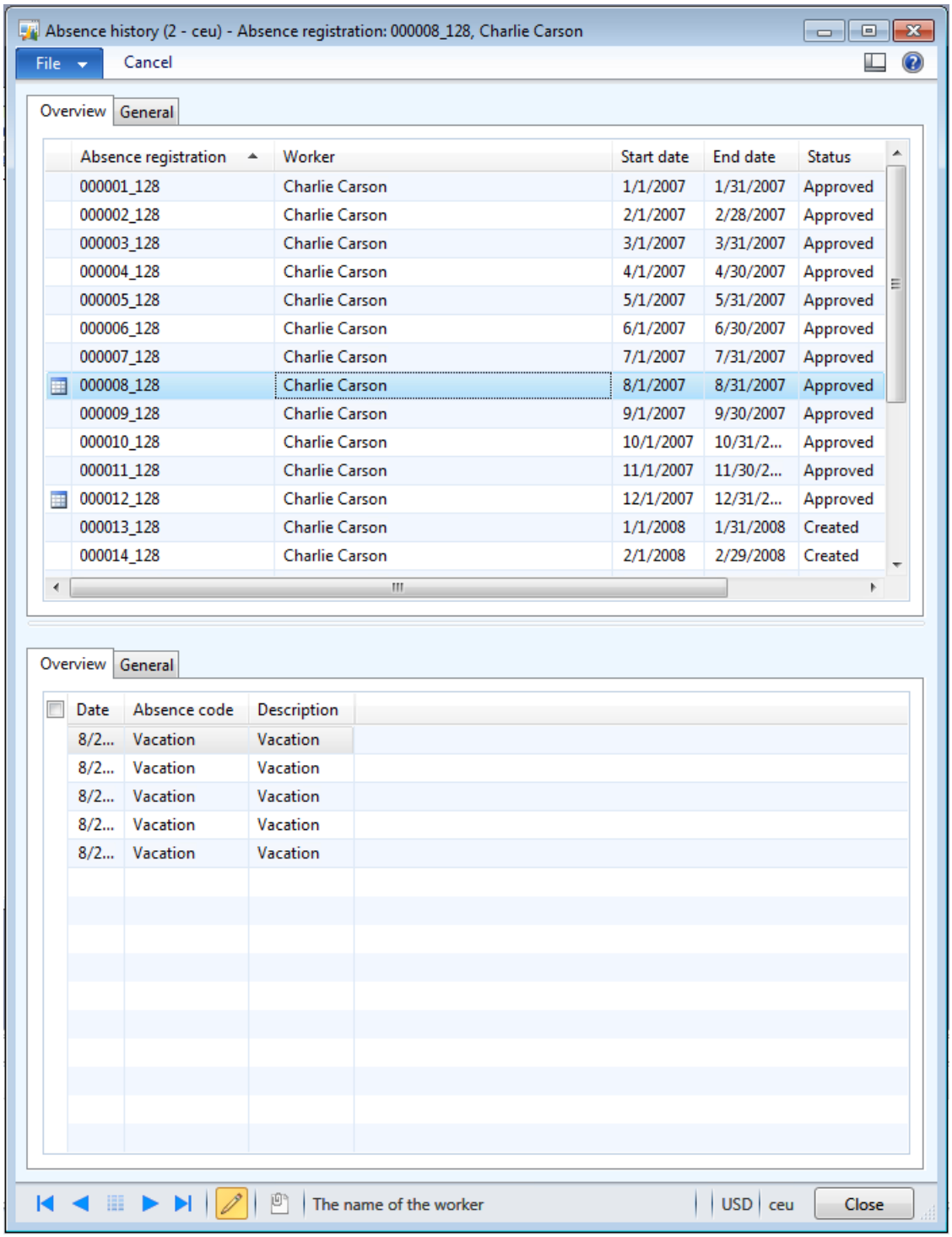
Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues

- None

AX 2012 content



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Task Single form pattern

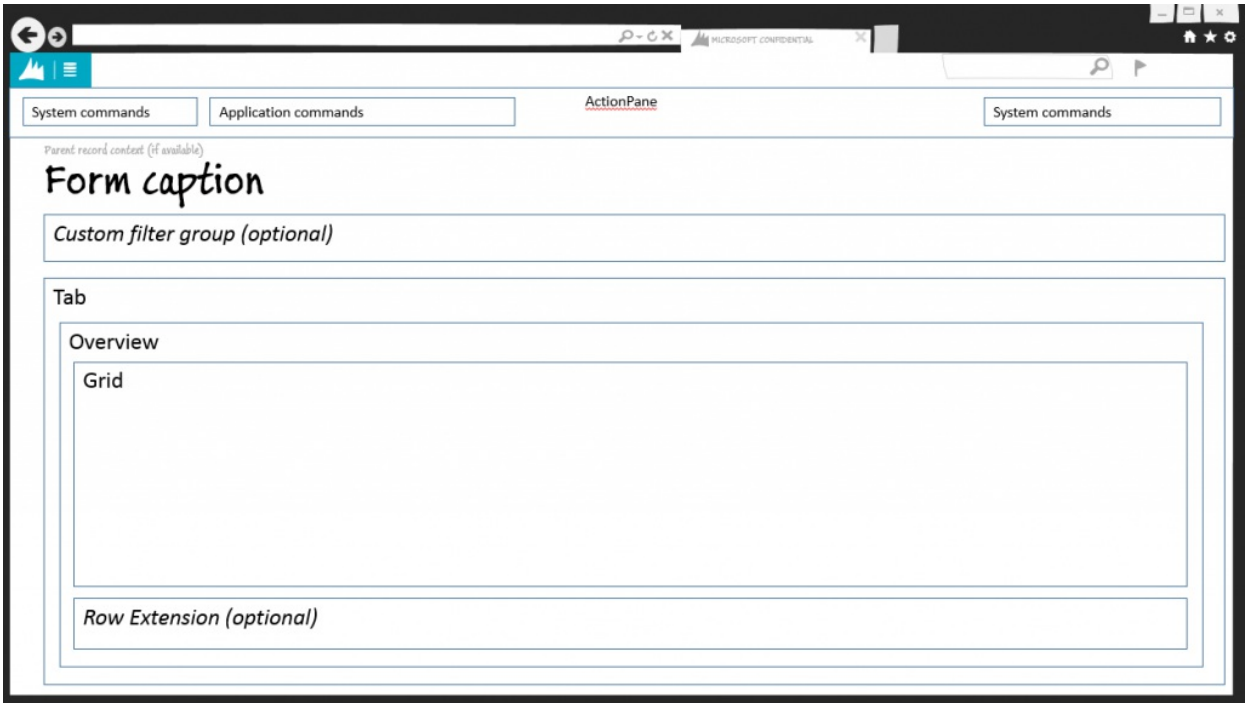
2/18/2021 • 2 minutes to read • Edit Online

This article provides information about the Task Single form pattern. This pattern was previously used to present data that users would perceive as originating from a single data source that had multiple records.

Usage

This type of form was used when you wanted to present data that users will perceive as originating from a single data source with multiple records. This isn't a recommended pattern for new forms. No new forms should be created that use this pattern. This pattern will provide structure and stability for legacy forms, and will also provide a migration path to more modern form patterns.

Wireframe



Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- The form opens in view mode.
- Commands have been moved to the standard ActionPane from a Toolbar (ActionPane strips).
- The **Overview** label on the first tab has been changed to **List**.
- The content of the tab container uses dynamic columns for a responsive layout.

Model

High-level structure

- Design
    - ActionPane (Action Pane)

- *CustomFilter (Group) [Optional]*
- Tab (Tab)
  - Overview (TabPage)
    - Grid (Grid)
    - *RowExtension (Group) [Optional]*
  - General (TabPage, repeats 0..N)
- *FooterGroup (Group) [Optional]*

Core components

1. Apply the TaskSingle pattern on **Form.Design**.
2. Address BP Warnings:
   a. **Design.Caption** isn't empty.
   b. The form must be referenced by at least one menu item.
   c. **TabPage.Caption** isn't empty.
   d. **TabPage.DataSource** isn't empty.
   e. **StaticText.Text** isn't empty.

Related patterns

- Task Double

Commonly used subpatterns

- Custom Filter Group
- Fields and Field Groups
- Toolbar and List
- Toolbar and Fields

UX guidelines

The verification checklist shows you the steps for manually verifying that the form complies with UX guidelines. This checklist doesn't include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps.

Standard form guidelines:

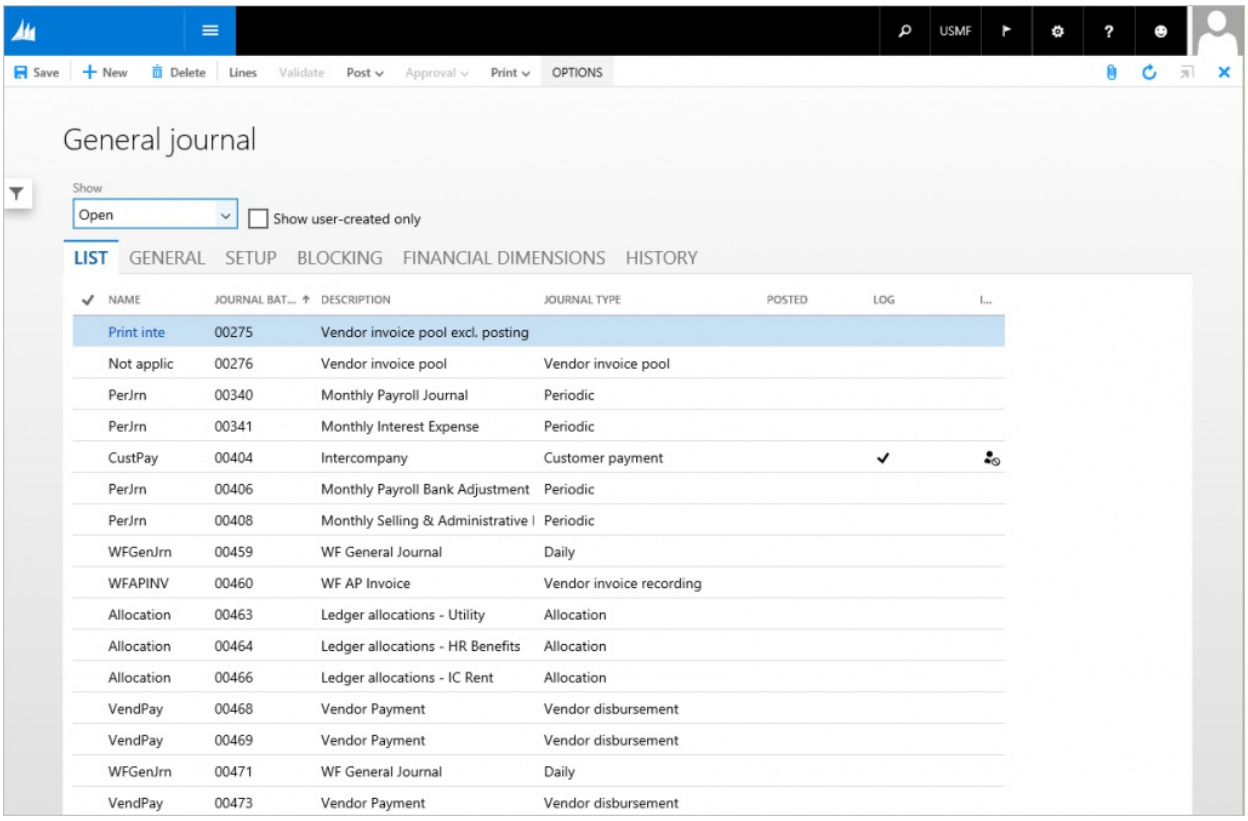
- Standard form guidelines have been consolidated into the Microsoft Dynamics AX General Form Guidelines document.

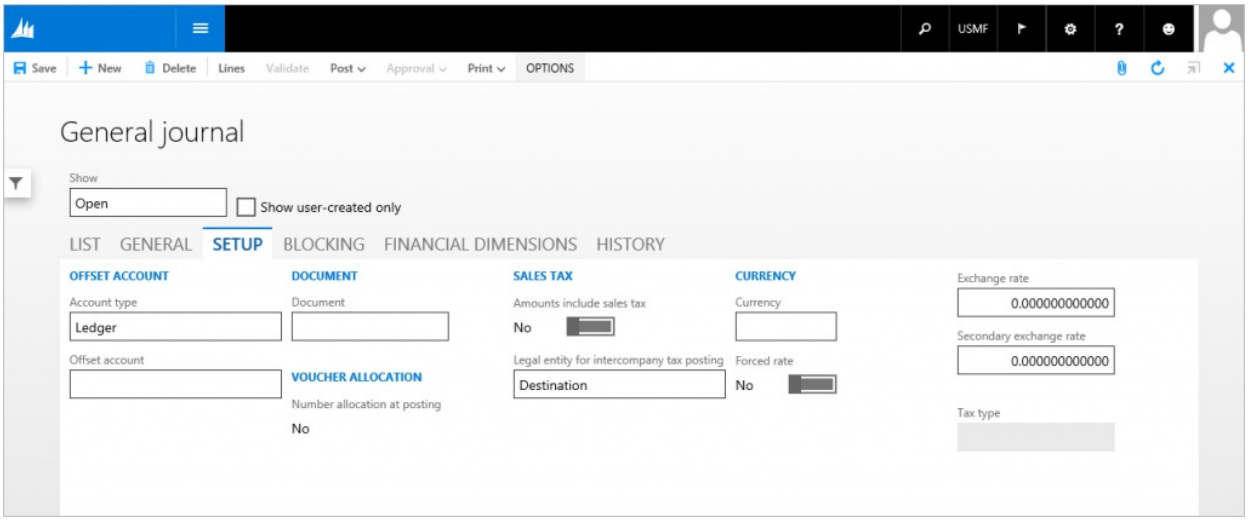
Task Single guidelines:

- The **Overview** tab is the first tab and is active when the form is opened.
- The **General** tab must be the second tab and must have the label **General**.

Examples

Form: **LedgerJournalTable**





Appendix

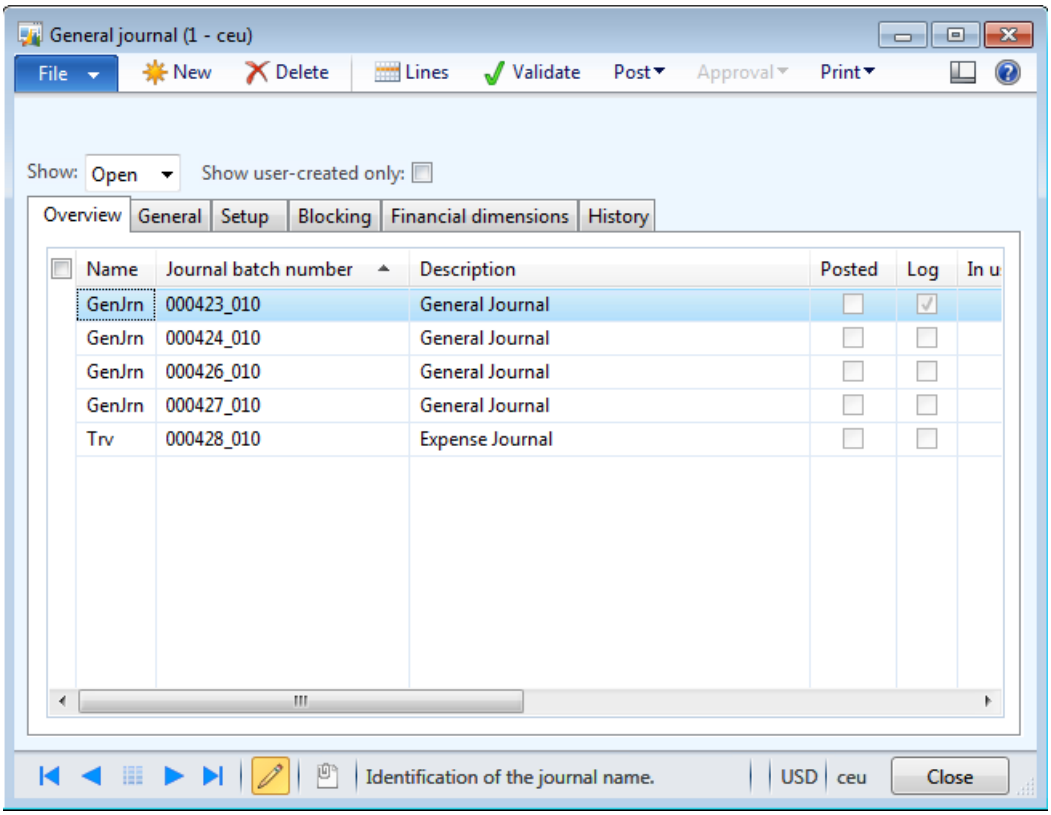
Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues

- None

AX 2012 content



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Wizard form pattern

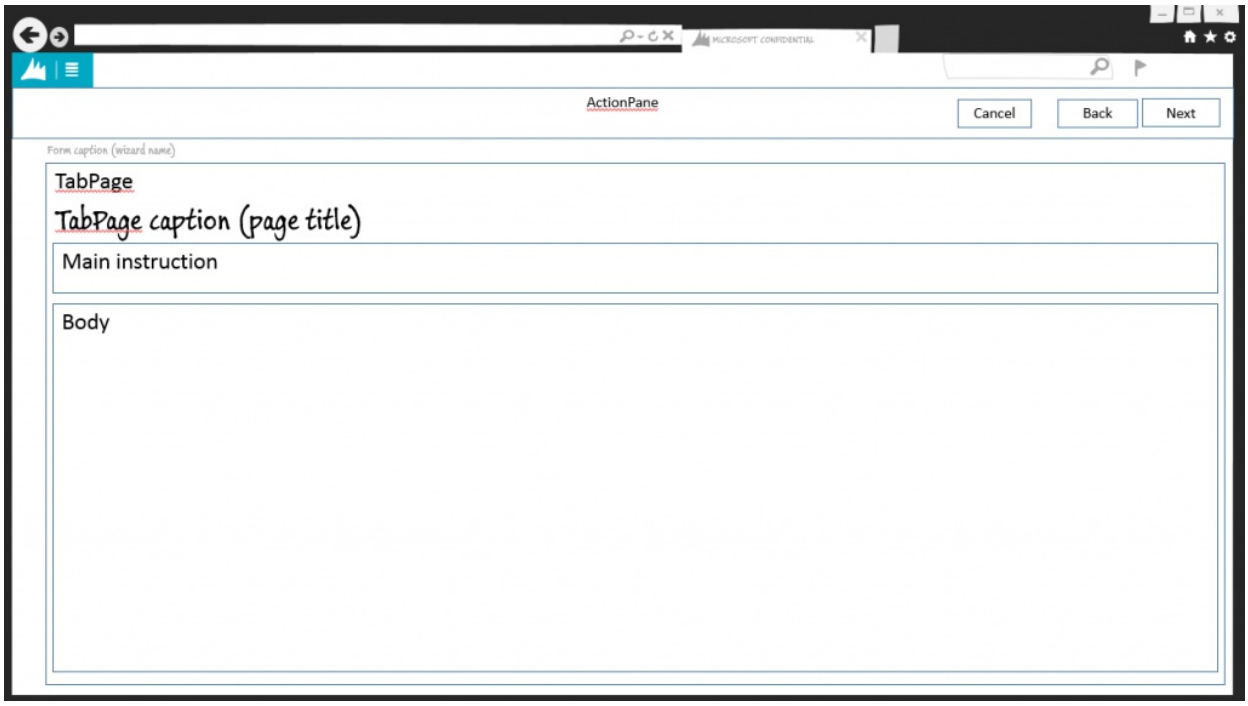
2/18/2021 • 2 minutes to read • Edit Online

This article provides information about the Wizard form pattern. A wizard is a special form of user assistance that takes the user through a task by using an ordered series of tab pages.

Usage

A wizard is a special form of user assistance that takes the user through a task by using an ordered series of tab pages. Wizards are especially useful for complex or infrequent tasks that the user might have difficulty learning or doing, or for tedious, frequently performed tasks.

Wireframe



Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- The secondary instruction for a wizard step was previously defined in the Help Text property of that step's Tab Page. This instruction will now be modeled on the Tab Page as a Static Text control.

Model

High-level structure

- Design (Style=Wizard; Caption=<wizard title>)
  - WizardContent (Tab)
    - WizardContentPage (TabPage) *[repeats 1..N times, can be named anything; Caption set to page title]*
      - MainInstruction (StaticText)

    - Body (Group)

Core components

1. Apply the Wizard pattern on **Form.Design**.
2. Address BP Warnings:
    a. **Design.Caption** isn't empty.
    b. The form must be referenced by at least one menu item.
    c. **TabPage.Caption** isn't empty (for all wizard content pages).
    d. **MainInstruction.Text** isn't empty (for all wizard content pages).

Commonly used subpatterns

- Fields and Field Groups
- Toolbar and List
- Toolbar and Fields
- List Panel

UX guidelines

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn't include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps.

Standard form guidelines:

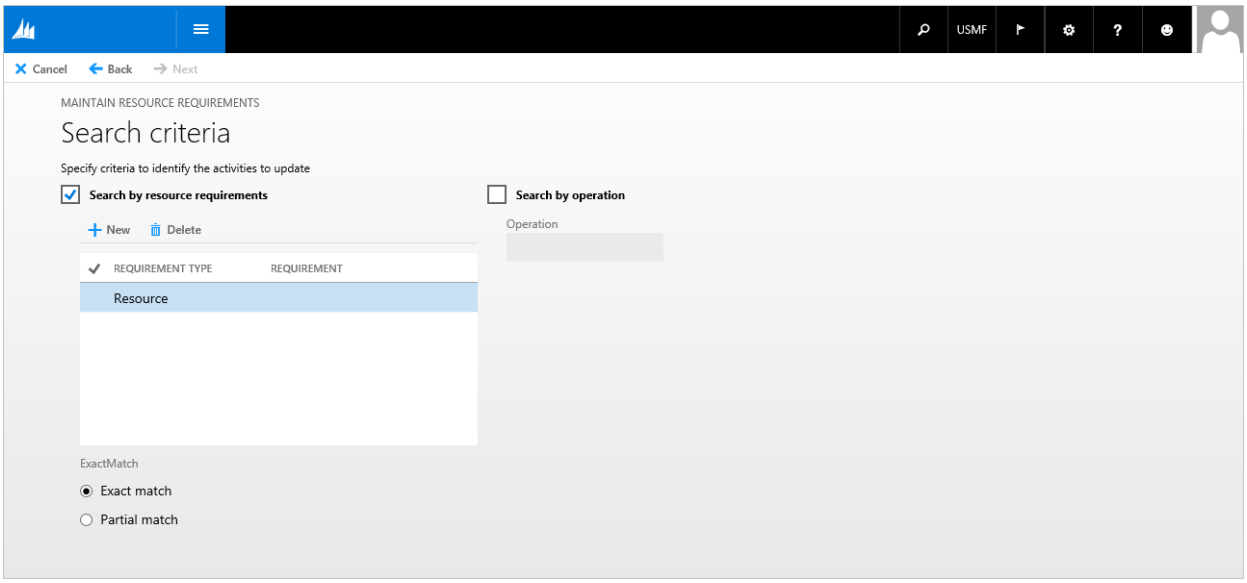
- Standard form guidelines have been consolidated into the Microsoft Dynamics AX General Form Guidelines document.

Wizard guidelines:

- Each tab page should have a title.
- Each tab page should have a main instruction.
- Content should be subdivided into logical groups per page.
- A wizard should have **<Next>** and **<Previous>** buttons on the appropriate pages.
- The user should also be able to cancel the wizard, and cancellation should return to the state that existed before the wizard was started.
- Only one question should be asked per wizard page (tab page).
- When a set of choices is presented to the user, radio buttons should be used to make the alternatives clear, even if a check box or combo box is otherwise acceptable.
- Wizard forms must **not** have these elements:
    - FactBoxes
    - FastTabs

Examples

Form: **WrkCtrBulkResReqEditWizard**



Appendix

Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues

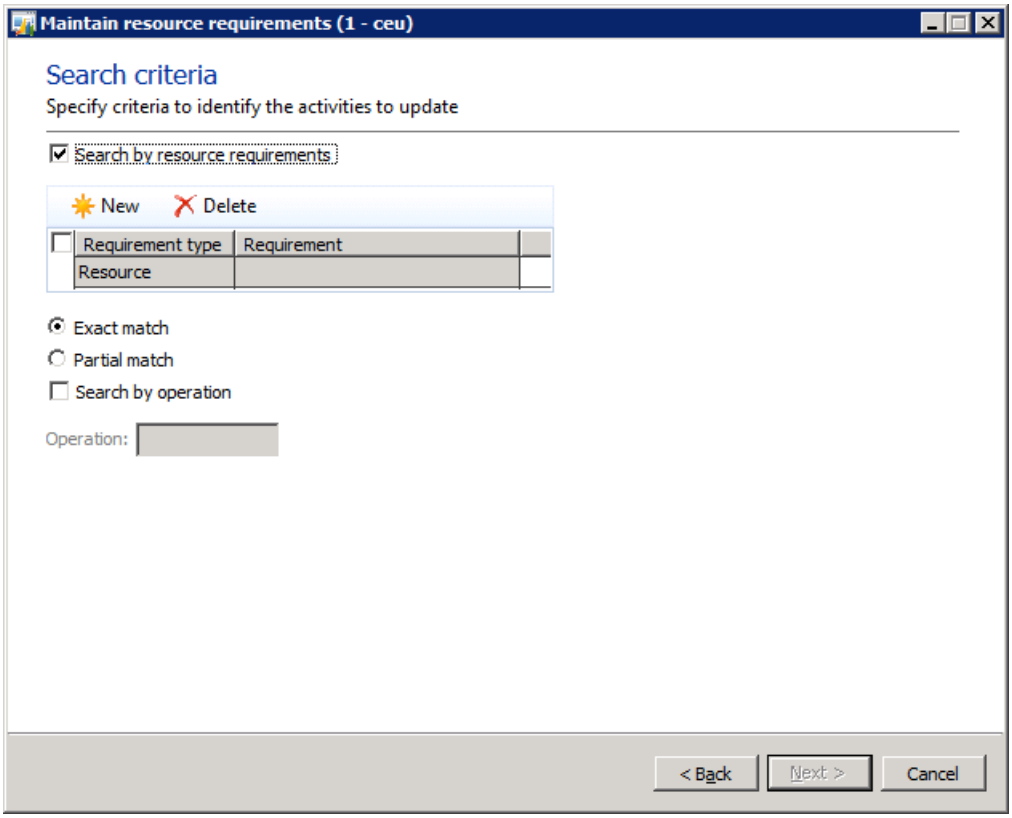
- None

AX 2012 content

AX 2012 links

- MSDN Wizards in Microsoft Dynamics AX [AX 2012]
- MSDN Guidelines for Wizard Development [AX 2012]

AX 2012 example



Workspace form pattern

2/18/2021 • 4 minutes to read • Edit Online

This topic discusses workspace form patterns. Workspaces are the primary way that users navigate to tasks and specific pages. A workspace should be created for every significant business activity that is supported.

Usage

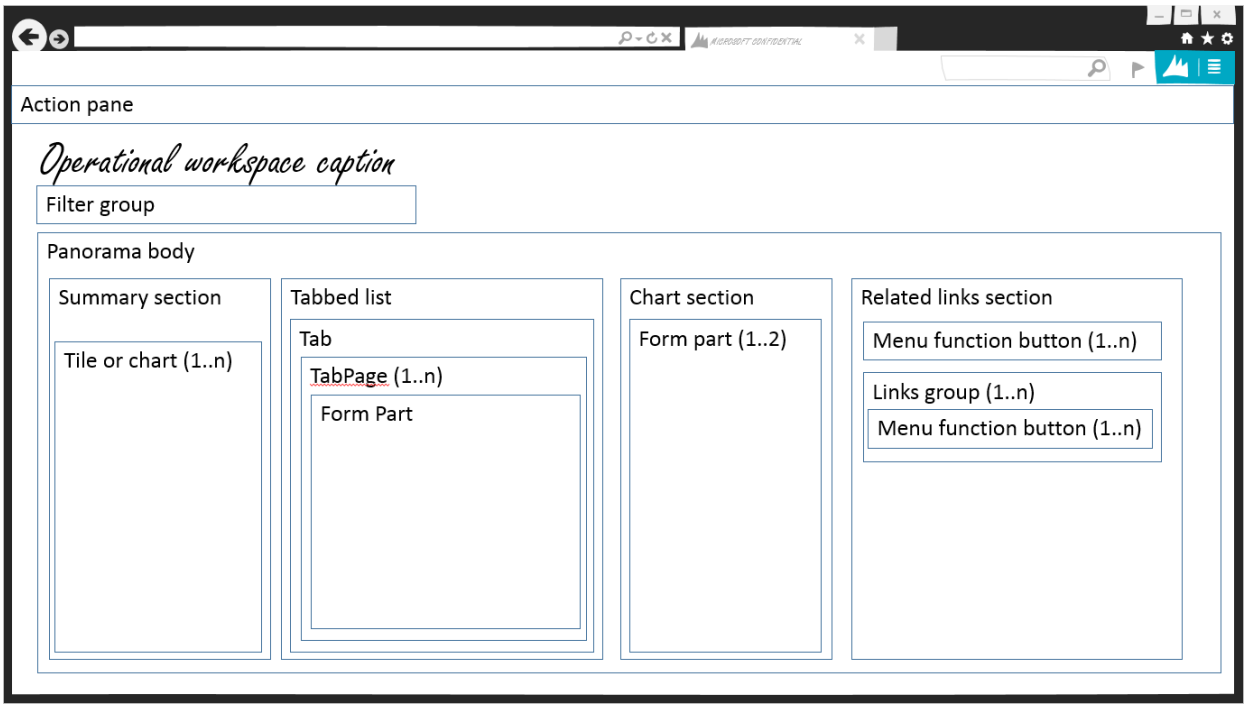
Workspaces are a new concept, and are meant to be the primary way that users navigate to tasks and specific pages. A workspace should be created for every significant business "activity" that you want to support. An "activity" is less granular than a task and more granular than a legacy "area page." A workspace is intended to provide a one-page overview of the activity, and to help users understand the current status, the upcoming workload, and the performance of the process or user. Users should be able to start the most typical tasks for the activity directly from the workspace. If possible, users should also be able to complete tasks directly in the workspace, based on the overview that they just received. Currently, there are two workspace patterns:

- **Tabbed workspace**: Instead of forcing a horizontally-scrolling panorama for content, this pattern uses standard tabs to allow the development of vertically-scrolling workspaces. This is particularly being used to embed Power BI reports into workspaces. Additional subpatterns to help define content inside these tabs will likely be provided in the future.
- **Operational workspace**: This is the standard pattern currently used for workspace development. Because of the set of components that are permitted in it, this pattern has superior performance over the deprecated "workspace" pattern. For this reason and to ensure visual and behavioral consistency with the other workspaces in the system, we recommend that you use this pattern.
- (Deprecated) **Workspace**: This pattern is only mentioned for the sake of completeness. Do **not** use this pattern. It will soon be removed from the product.

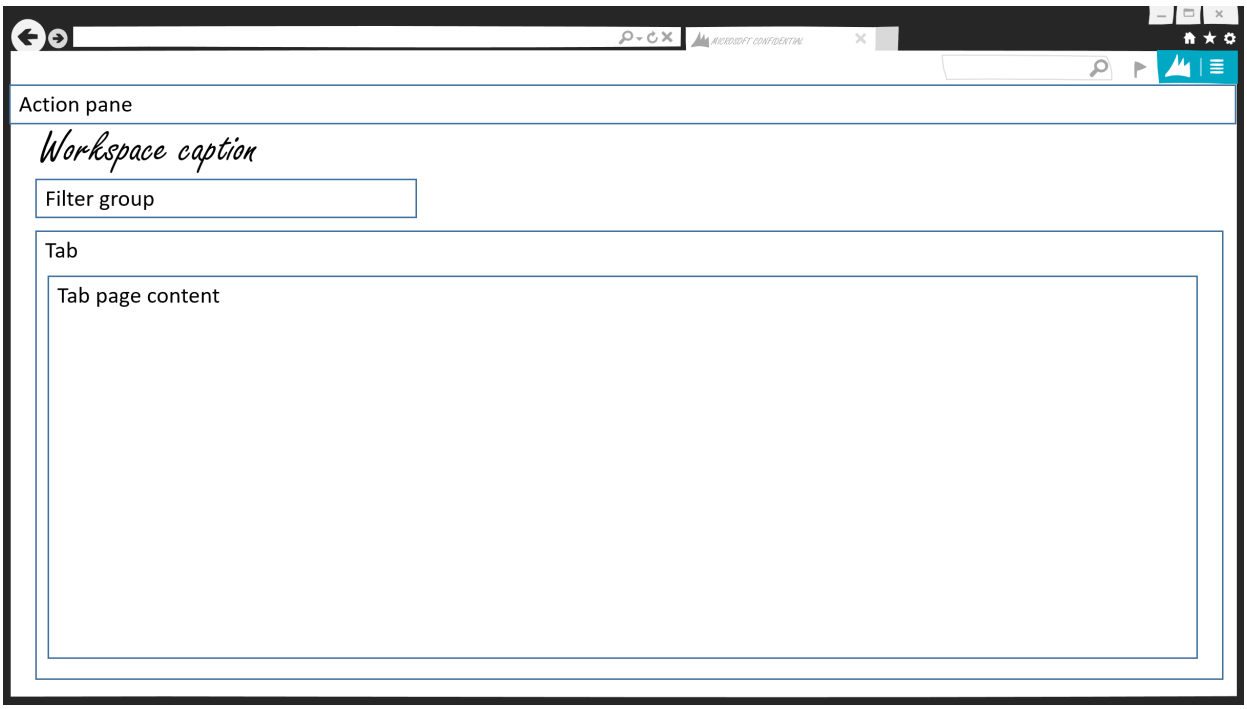
The rest of this topic will focus on the Operational workspace pattern and the Tabbed Workspace pattern, as the original Workspace pattern is deprecated and should not be used.

Wireframe

Operational workspace



Tabbed workspace



Pattern changes for Finance and Operations

The Microsoft Dynamics AX 2012 Role Center has been replaced by multiple activity-focused workspaces.

Model

Operational workspace – High-level structure

- Design
  - *Action pane (ActionPane) [Optional]*
  - *Workspace page filter group (Group) [Optional]* – This must use the Workspace Page Filter Group subpattern.
  - Panorama (Tab)

    - Section summary tiles (TabPage) – This must use the Section Tiles subpattern.
    - Section tabbed list (TabPage) – This must use the Section Tabbed List subpattern.
    - *Section charts (TabPage) [Optional]* – This must use the Section Stacked Chart subpattern.
    - *Section PowerBI (TabPage) [Optional]* – This must use the Section PowerBI subpattern.
    - Section related links (TabPage) – This must use the Section Related Links subpattern.

Tabbed workspace – High-level structure

- Design
    - *Action pane (ActionPane) [Optional]*
    - *Workspace page filter group (Group) [Optional]* – This must use the Workspace Page Filter Group subpattern.
    - StandardTab (Tab)
        - ContentTabPage (1..N)

Core components

- Apply the appropriate Workspace pattern on **Form.Design**.
- Address BP Warnings:
    - **Form** must be referenced by at least one menu item.
    - **TabPage.Caption** isn't empty (for all panorama sections).

Commonly used subpatterns

- Workspace Page Filter Group
- Section Tiles
- Section Tabbed List
- Section Stacked Chart
- Section PowerBI
- Section Related Links

Related patterns

- Form Part Section List
- Section Chart

UX guidelines

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn't include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps.

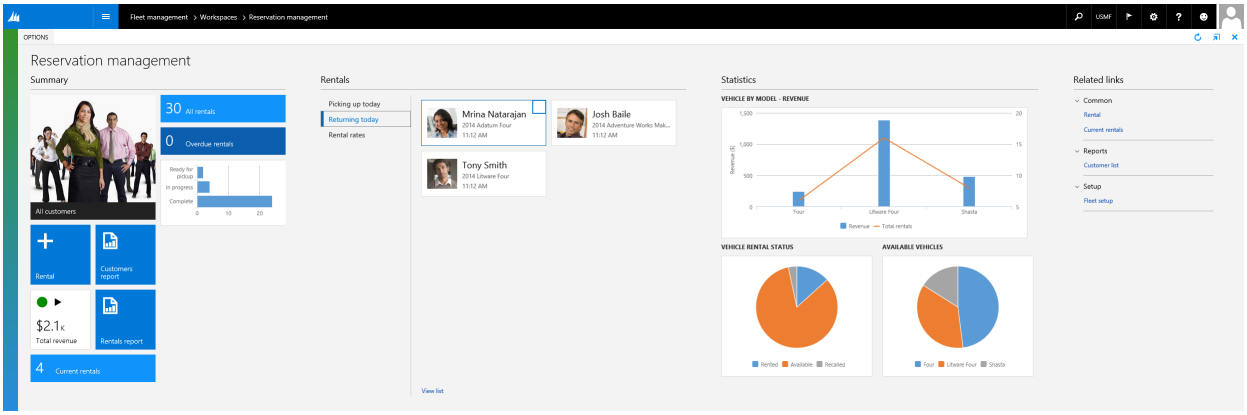
- Standard form guidelines
    - Standard form guidelines have been consolidated into the General Form Guidelines document.
- Workspace form guidelines
    - Use a noun phase for the page title, and avoid general words. The page title should not duplicate the title of an area page.
    - The page title should begin with the noun that users would have in mind.
    - All sections must have a title.
    - A section typically spans the width of two to four standard tiles.

    - Any section that uses a FormPartControl to display content should have **HeightMode** set to **SizeToAvailable** on the FormPartControl.
- Actions
    - Include only frequently used commands.
    - Actions on the Action Pane should be related to the whole workspace (not a specific section of it).
        - **Exception:** A single "New" action can be put as a tile in the **Summary** section if it's very frequently used.
    - Group variations of the same command on drop-down menus.
        - **Examples:** New sales quote, New sales order, New return order
- Filters
    - Zero to five filter fields are allowed on a workspace.
        - Only a single field can be put under the page title
        - The remaining filters must be in a workspace configuration dialog.

Example

Operational workspace

Form: **FMClerkWorkspace**



Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues

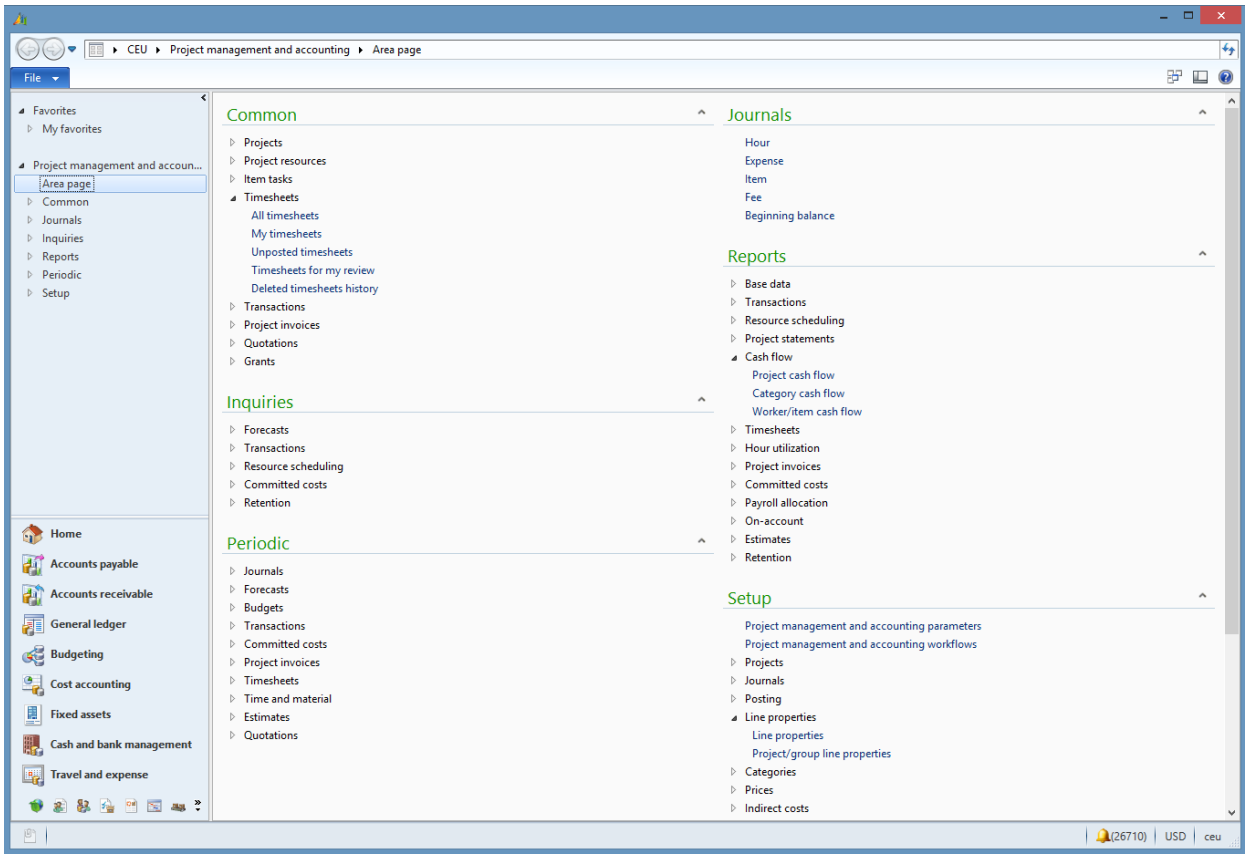
- None

AX 2012 content

AX 2012 links

- MSDN Role Center Page Reference [AX 2012]
- MSDN Role Center User Experience Guidelines [AX 2012]

AX 2012 example



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Advanced selection form pattern

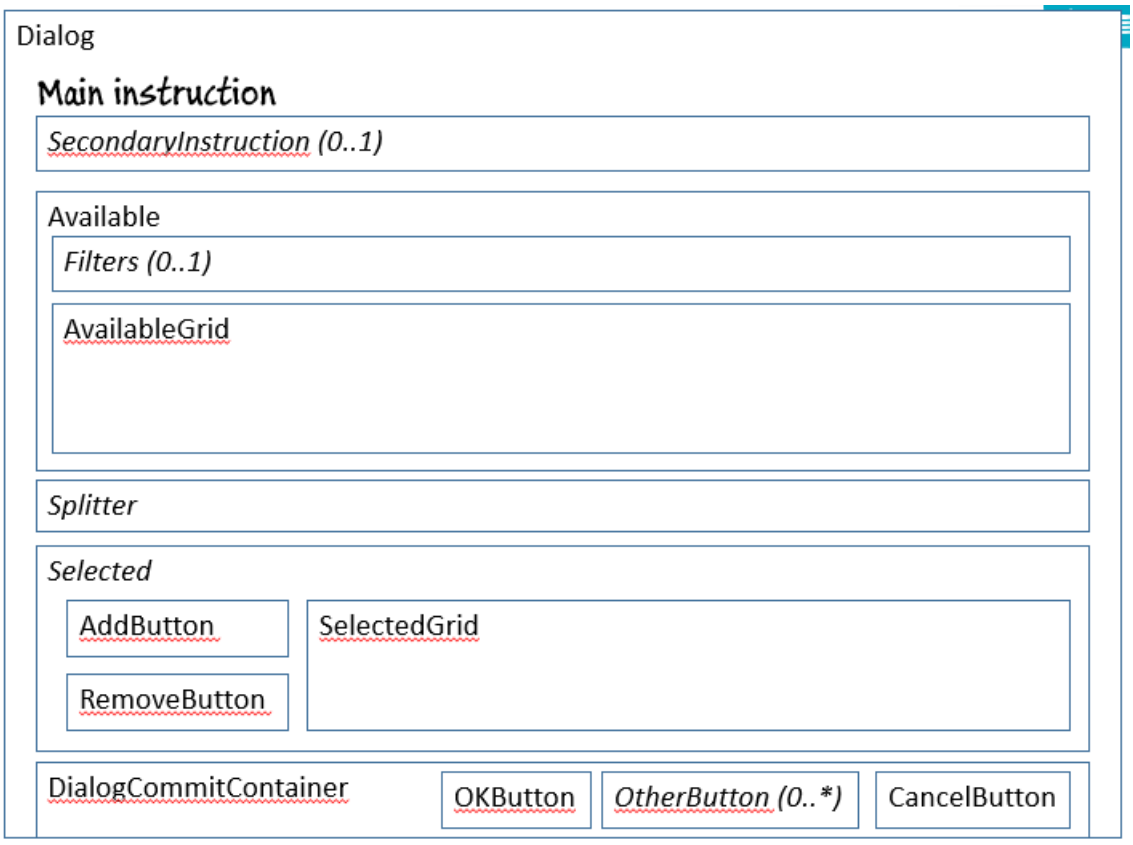
2/18/2021 • 2 minutes to read • Edit Online

This article provides information about the Advanced Selection form pattern. This Dialog form pattern lets users filter and select items from a large, wide list. Like the List Panel pattern, this pattern should be used when the primary user task is to select a set of items.

Usage

The Advanced Selection form pattern should be used when the primary user task is to select a set of items. This task is usually accomplished through a multi-select list. However, in many scenarios, users must select items that aren't contiguous and, at the same time, must see the set of items that they are selecting. This pattern resembles the List panel pattern, in that the user selects items in one list and adds them to another. However, this pattern allows for custom filters and a "wide" list on top, and uses most of the screen "real estate" of the page (typically, it's a Large dialog). Use this pattern when a user must be able to filter and select in a large, wide list.

Wireframe



Related patterns

- List Panel subpattern
- Dialog form pattern

UX guidelines

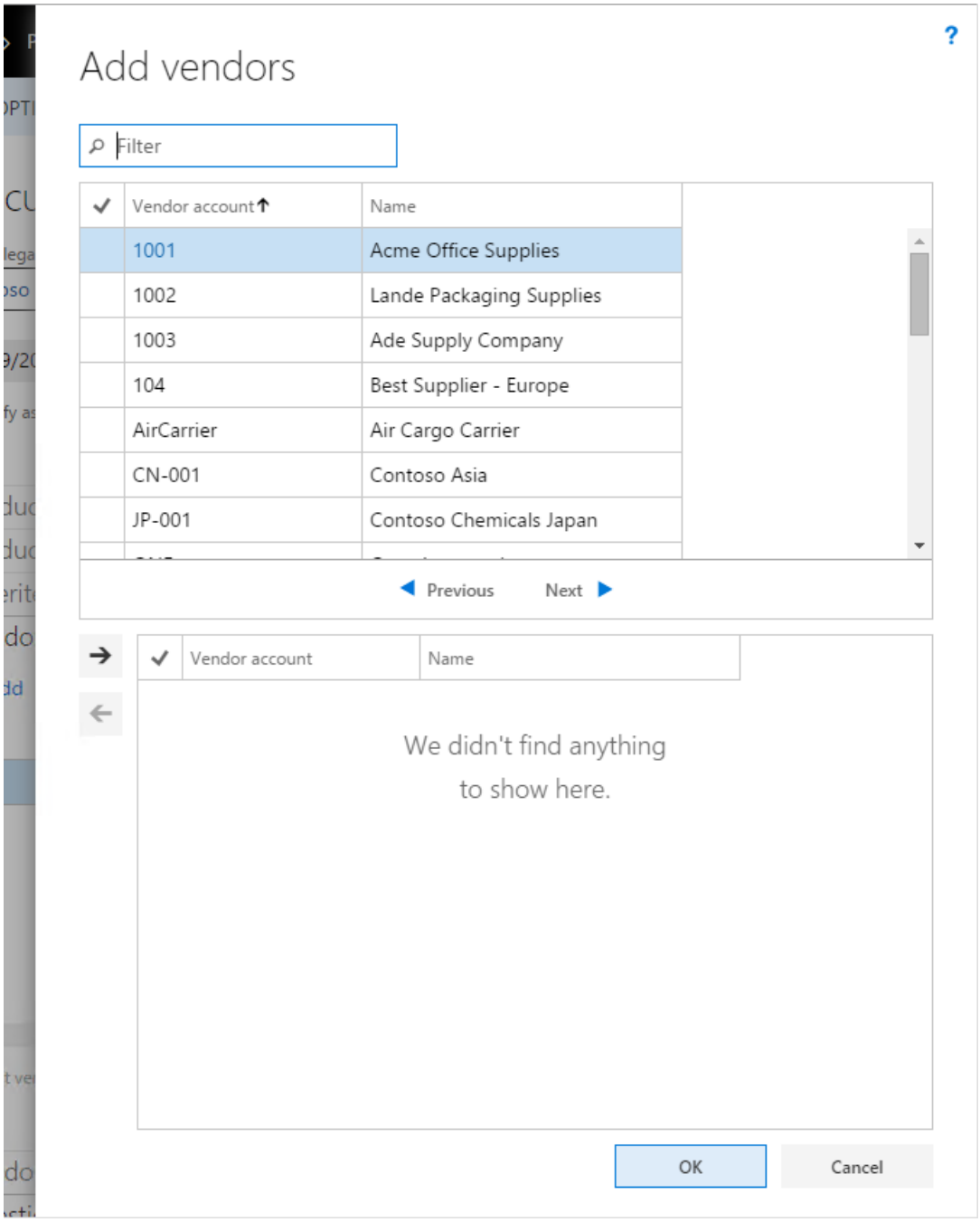
The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn't include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps.

- **Standard form guidelines:**
  - Standard form guidelines have been consolidated into the General form guidelines document.
- **Advanced selection guidelines:**
  - By default, the Quick filter should use the name or description column.
  - The list can display up to 15 columns. **Note:** This guidelines has been relaxed since Microsoft Dynamics AX 2012.
  - The main instruction should instruct users what they need to do.
  - When there is no data, the grid should not automatically add a new record.

Example

Form: **ProcCategoryAddVendor** (Click **Procurement and sourcing** > **Procurement categories**. On the **Vendors** FastTab, click **Add**.)

> [NOTE] This form no longer utilizes this pattern; however, the image shows an example of what a typical Advanced selection form pattern looks like.



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Dialog form pattern

2/18/2021 • 8 minutes to read • Edit Online

This topic provides information about the Dialog form pattern. A dialog box represents an action or activity that users can explicitly commit or cancel. It's used when a user initiates a specific task or process, and the system requires user input about how or whether to proceed.

Usage

A dialog box represents an action or activity that users can explicitly commit or cancel. It's used when a user initiates a specific task or process, and the system requires user input about how or whether to proceed. Dialogs are modal and require that users interact with the controls in the dialog before they can return to the parent page. Dialogs also can have multiple sizes. Selection of a dialog size is subjective, and will vary, depending on the form elements that you've modeled on the dialog. The sizes are as follows:

- **Small** – This size is a one-column-wide dialog. If your dialog contains a relatively small amount of content (all simple fields, and no wide tables or other wide elements), you can probably use this size.
- **Medium** – This size is a two-column-wide dialog. If your dialog contains more content than can comfortably fit within a small dialog, but a full-width dialog isn't required, you should use this size.
- **Large** – This size is a three-column-wide dialog. If your dialog contains more content than can comfortably fit within a medium dialog, but a full-width dialog isn't required, you should use this size.
- **Full** – A large dialog is nearly the full width of the browser viewport. Its size varies, depending on the viewport width, and it will always be the largest dialog size option. Use this size if your dialog has a lot of wide elements, or if it requires an unusually large amount of horizontal space.

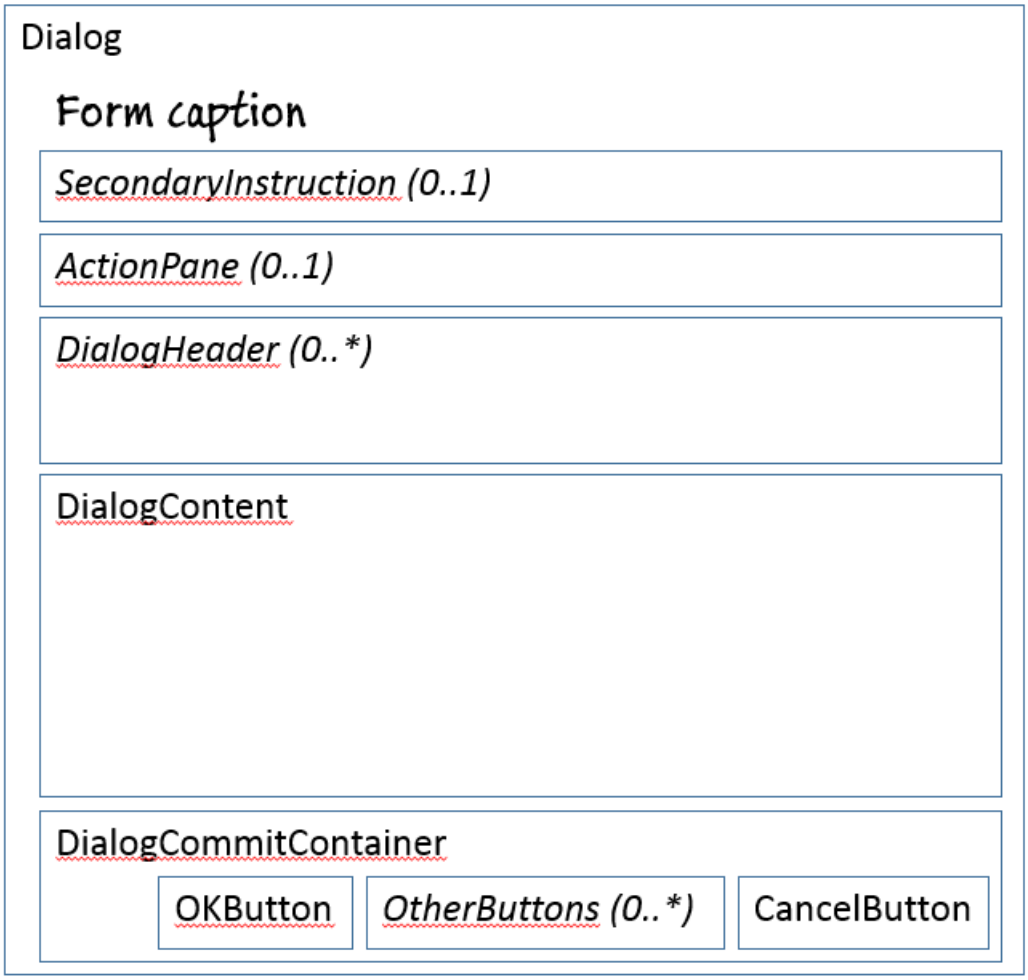
For more detail about the various dialog sizes, see the table in the appendix of this topic, under "Selecting the correct dialog size." We strongly recommend that you review that table. Five patterns are described in this document:

- **Dialog** – This is the basic dialog pattern. Use this dialog if you don't have a reason to use one of the other Dialog patterns.
- **Dialog w/tabs** – This is a more specific version of the Dialog pattern. It incorporates a Tab control in the dialog. You can also optionally provide a header for the Tab, and also a footer.
- **Dialog w/FastTabs** – This closely resembles the Dialog w/tabs pattern but uses FastTabs instead of regular tabs to organize the information.
- **Dialog w/double tabs** – This closely resembles the Dialog w/tabs pattern but has a second Tab control immediately after the first one.
- **Dialog (read only)** – This pattern is for informational forms that aren't editable. The user can still switch between tabs or a view selector, but direct manipulation of input fields isn't allowed. This dialog variation also includes a **Close** button instead of **OK** and **Cancel** buttons.

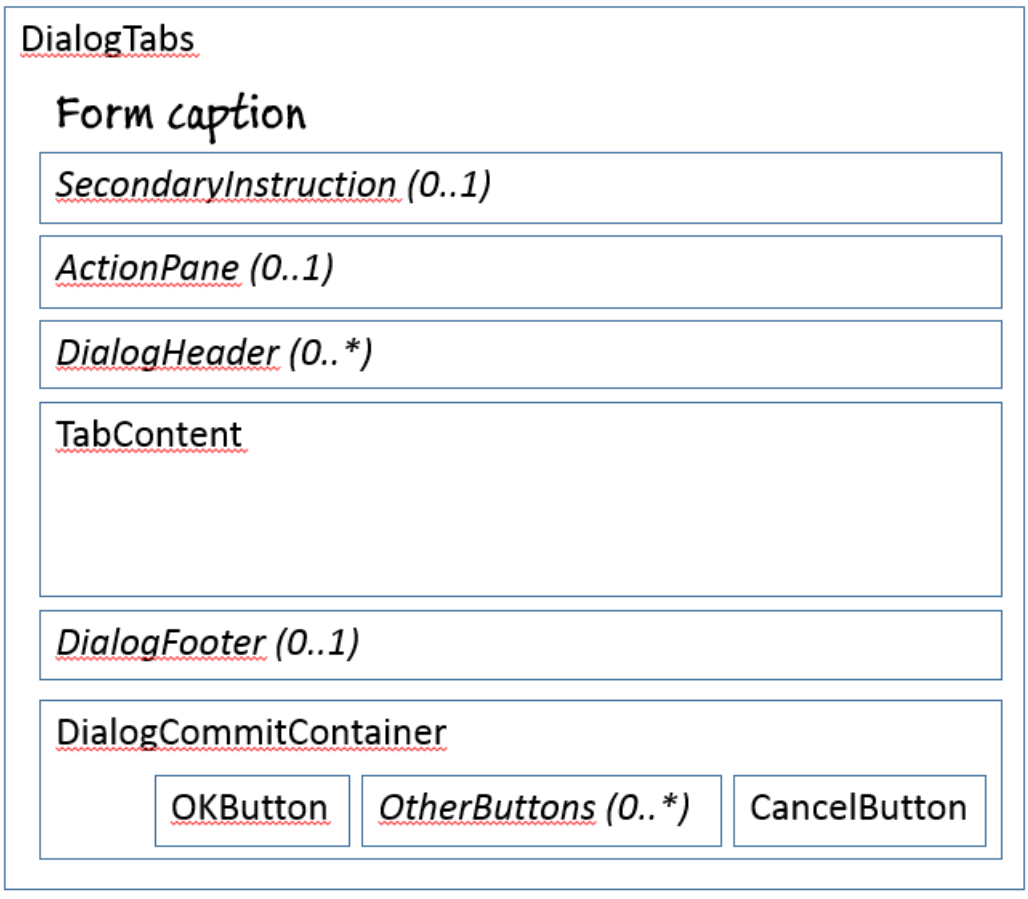
Wireframe

The following sections show the wireframes for the four dialog types that are included in this article.

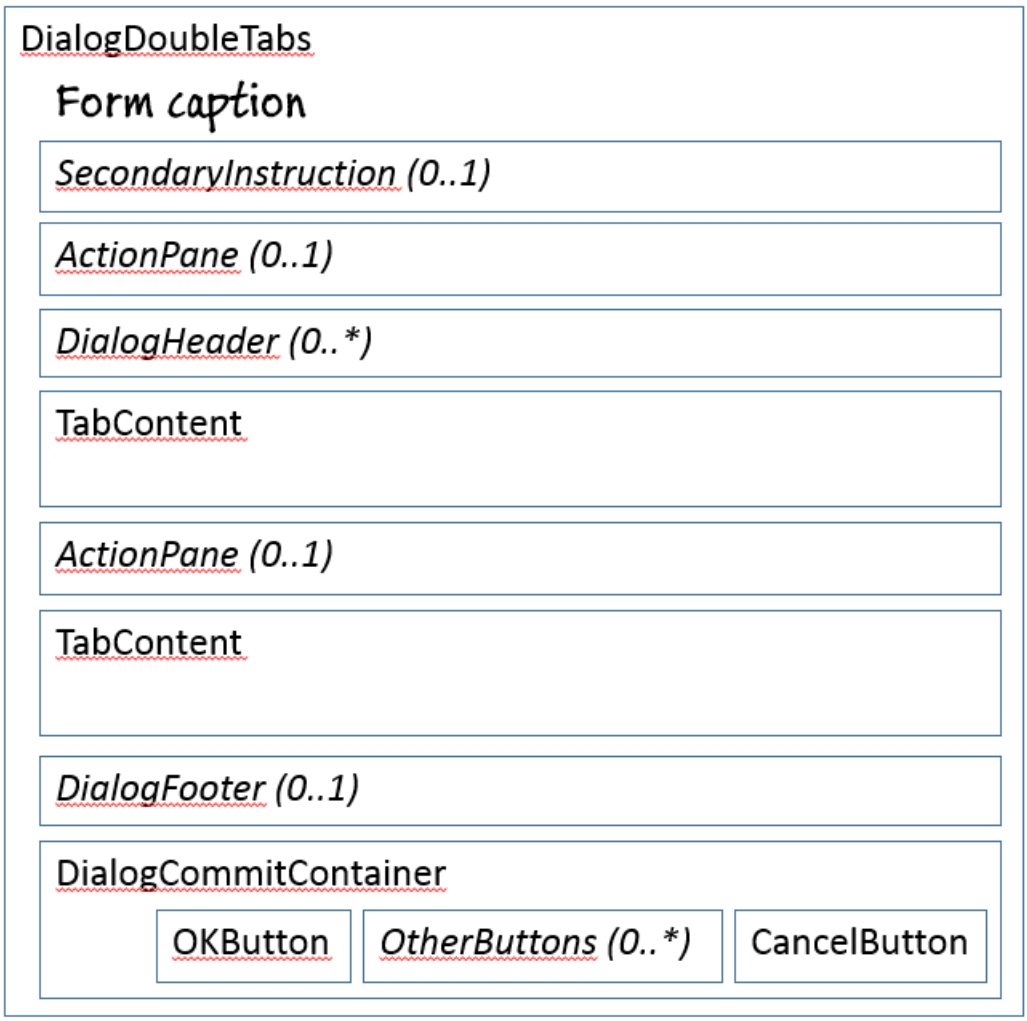
Dialog



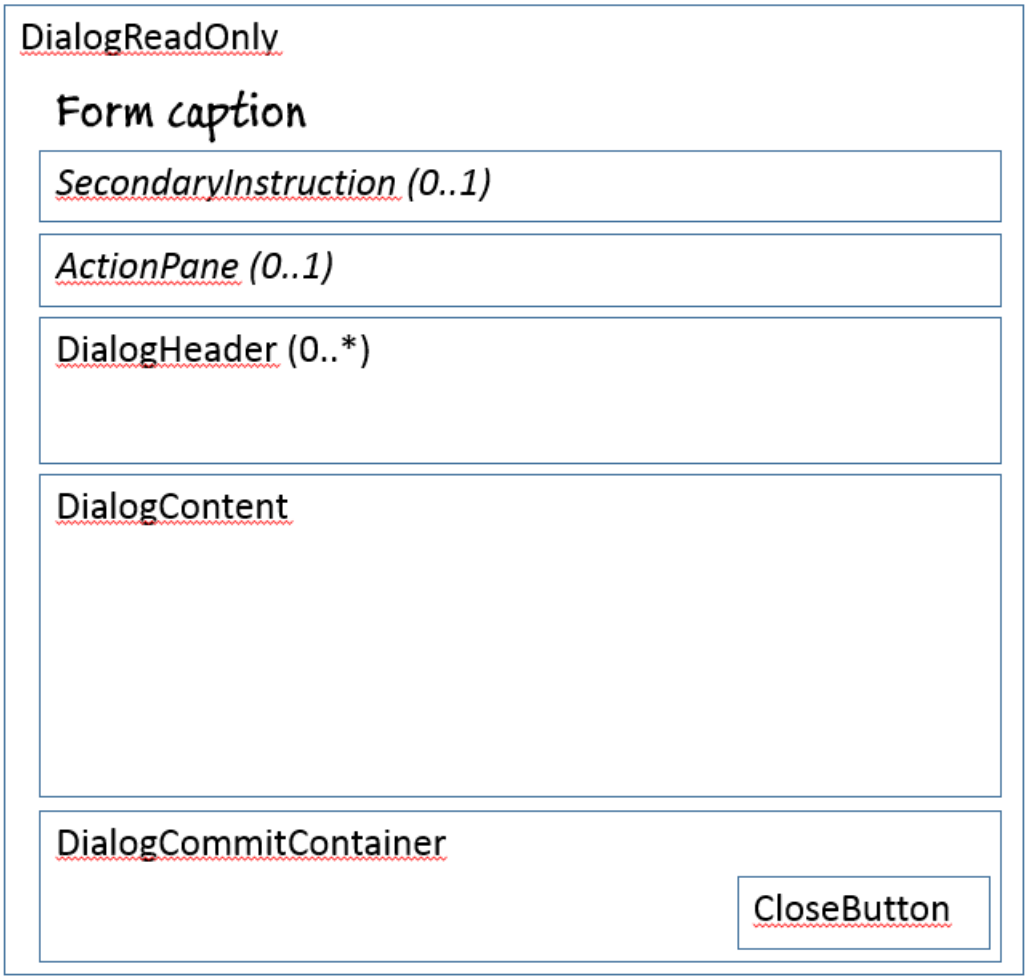
Dialog w/tabs and Dialog w/FastTabs



Dialog w/double tabs



Dialog (read only)



Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- The form caption serves as the MainInstruction. Therefore, a modeled MainInstruction control is no longer required.
- Manual handling of error messages is no longer required.
- In some cases, buttons will scroll off the bottom of the slider if the form content exceeds the available height.
- Slider dialogs have their own message bar (because the main message bar is obscured when a slider is open).

Model

Dialog (basic) – High-level structure

- Design
    - *SecondaryInstruction (StaticText) [Optional]*
        - *ActionPane (ActionPane) [Optional]*
        - *DialogHeader (Group, can repeat) [Optional]*
        - DialogContent (Group, repeats 1..N)
        - DialogCommitContainer (ButtonGroup)
            - OKButton ($Button)
            - *OtherButton ($Button, can repeat) [Optional]*
            - CancelButton ($Button)

Dialog w/Tabs and Dialog w/FastTabs – High-level structure

- Design
  - *SecondaryInstruction (StaticText) [Optional]*
  - *ActionPane (ActionPane) [Optional]*
  - *DialogHeader (Group, can repeat) [Optional]*
  - TabContent (Tab)
    - TabPage (TabPage, repeats 1..N)
  - *DialogFooter (Group) [Optional]*
  - DialogCommitContainer (ButtonGroup)
    - OKButton ($Button)
    - *OtherButton ($Button, can repeat) [Optional]*
    - CancelButton ($Button)

Dialog w/double tabs – High-level structure

- Design
  - *SecondaryInstruction (StaticText) [Optional]*
  - *ActionPane (ActionPane) [Optional]*
  - *DialogHeader (Group, can repeat) [Optional]*
  - TabContent (Tab)
    - TabPage (TabPage) [1..*]
  - TabContent (Tab)
    - TabPage (TabPage) [1..*]
  - *DialogFooter (Group) [Optional]*
  - DialogCommitContainer (ButtonGroup)
    - OKButton ($Button)
    - *OtherButton ($Button, can repeat) [Optional]*
    - CancelButton ($Button)

Dialog (read only) – High-level structure

- Design
  - *SecondaryInstruction (StaticText) [Optional]*
  - *ActionPane (ActionPane) [Optional]*
  - *DialogHeader (Group, can repeat) [Optional]*
  - DialogContent (Group, repeats 1..N)
  - DialogCommitContainer (ButtonGroup)

    CloseButton ($Button)

Core components

- Apply the Dialog pattern on **Form.Design**.
- Address BP Warnings:

- **Design.Caption** isn't empty.
- The form must be referenced by at least one menu item.
- **StaticText.Text** isn't empty.

Related patterns

- Drop Dialog

Commonly used subpatterns

- Fields and Field Groups
- Toolbar and List
- Toolbar and Fields
- Fill Text

UX guidelines

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn't include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps. **Standard form guidelines:**

- Standard form guidelines have been consolidated into the General Form Guidelines document.

Dialog guidelines:

- Focus should be in the first editable field in the dialog box when the dialog box is first opened.
  - **Exception:** If the dialog is read-only, focus should be on the **Close** button.
- A dialog must have a **main instruction** at the top.
  - A final period should not be included if the instruction is a statement. If the instruction is a question, a question mark should be included.
- A secondary instruction to the user can optionally be included, and it should present additional information that will help the user understand or use the dialog box. The secondary instruction should consist of a complete sentence in sentence case and should have end punctuation.
- A dialog must have a **content** area.
- For editable dialogs:
  - The content area should contain only the controls that are required in order to complete the task.
  - Constrained input controls, such as selection lists, check boxes, radio buttons, and command links, should be used to avoid validation errors.
  - Reasonable default values for each input should be provided whenever possible.
  - Controls should not cause another dialog to appear during validation.
    - Warning messages for validation issues should be displayed in a message bar as soon as possible.
- For read-only dialogs:
  - The content area should only contain controls that are non-editable or only allow the user to switch the data is displayed, such as a view selector.
  - Programmatically changing the value of a field should not cause validation errors.
  - If your dialog has multiple tabs, the tab that has the most content must define the selection of the dialog width.
- A dialog must have a **commit** button area:
  - For an editable dialog only, there is a commit button that starts the action that is implied by the main instruction.
  - The labels should make sense on their own and should be a response to the main instruction.
  - For both editable and read-only dialogs, the right-most button is a **Cancel** button that cancels the

operation without side-effects.

- There is a button that is marked as the default button for the dialog.
- The button that is selected as the default button should be the safest, most secure response to the task that the user is performing, such as the main instruction of a Dialog or Drop Dialog.
  - If safety and security aren't factors, the button that is most likely to be clicked or that is most convenient for the user should be selected as the default button.
  - **Exception:** Don't select a destructive response as the default unless there is an easy, obvious way to undo the command.

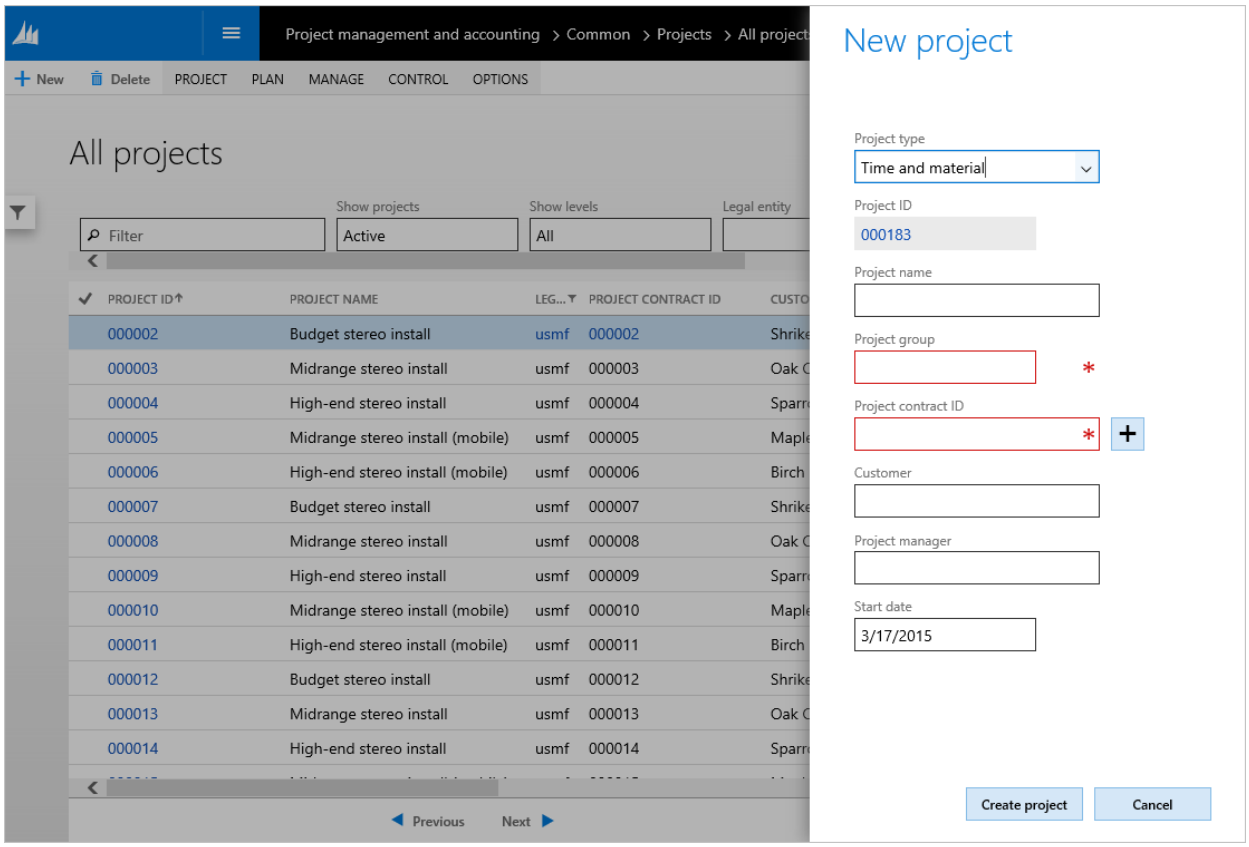
A dialog should **not** have these elements:

- FactBoxes

Examples

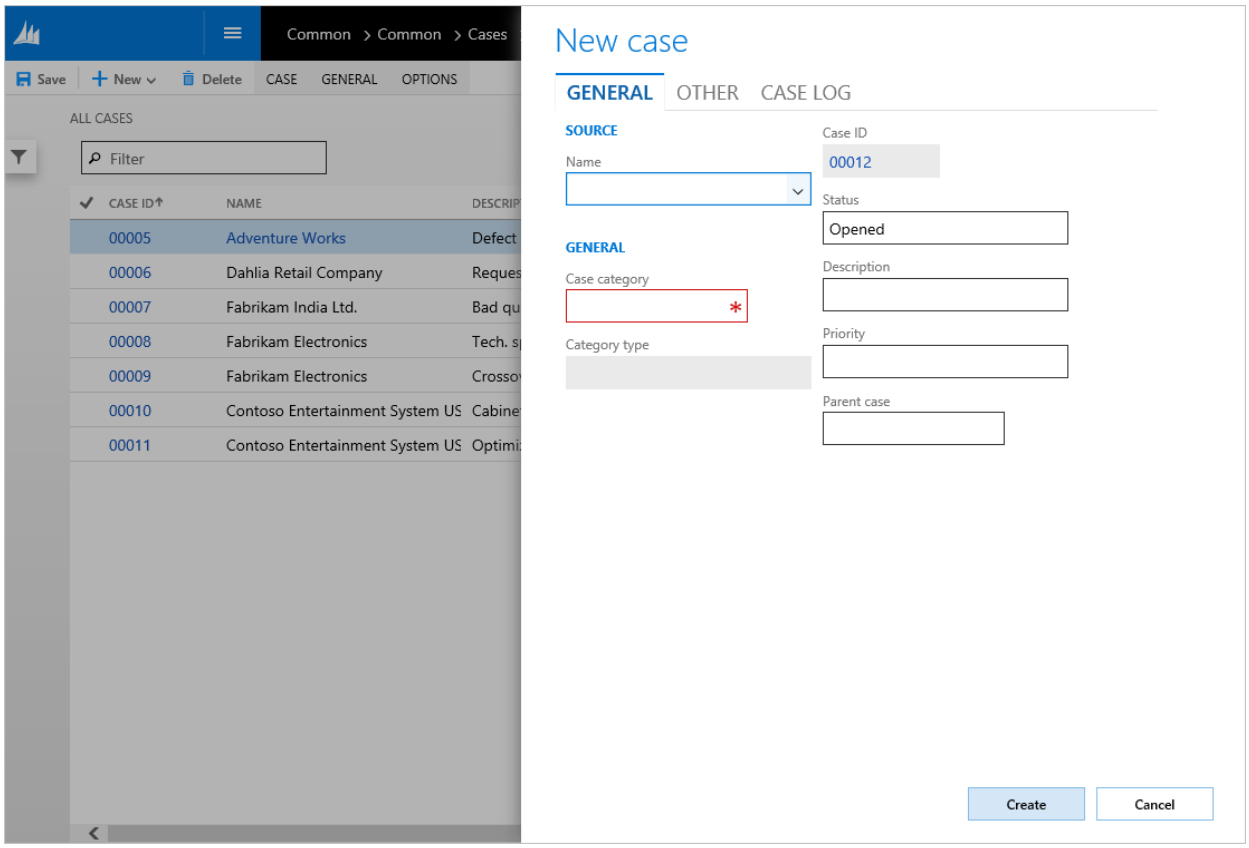
Dialog (basic)

Form: **ProjTableCreate** (Click **Project management and accounting** > **Common** > **Projects** > **All projects**, and then click **New**.)



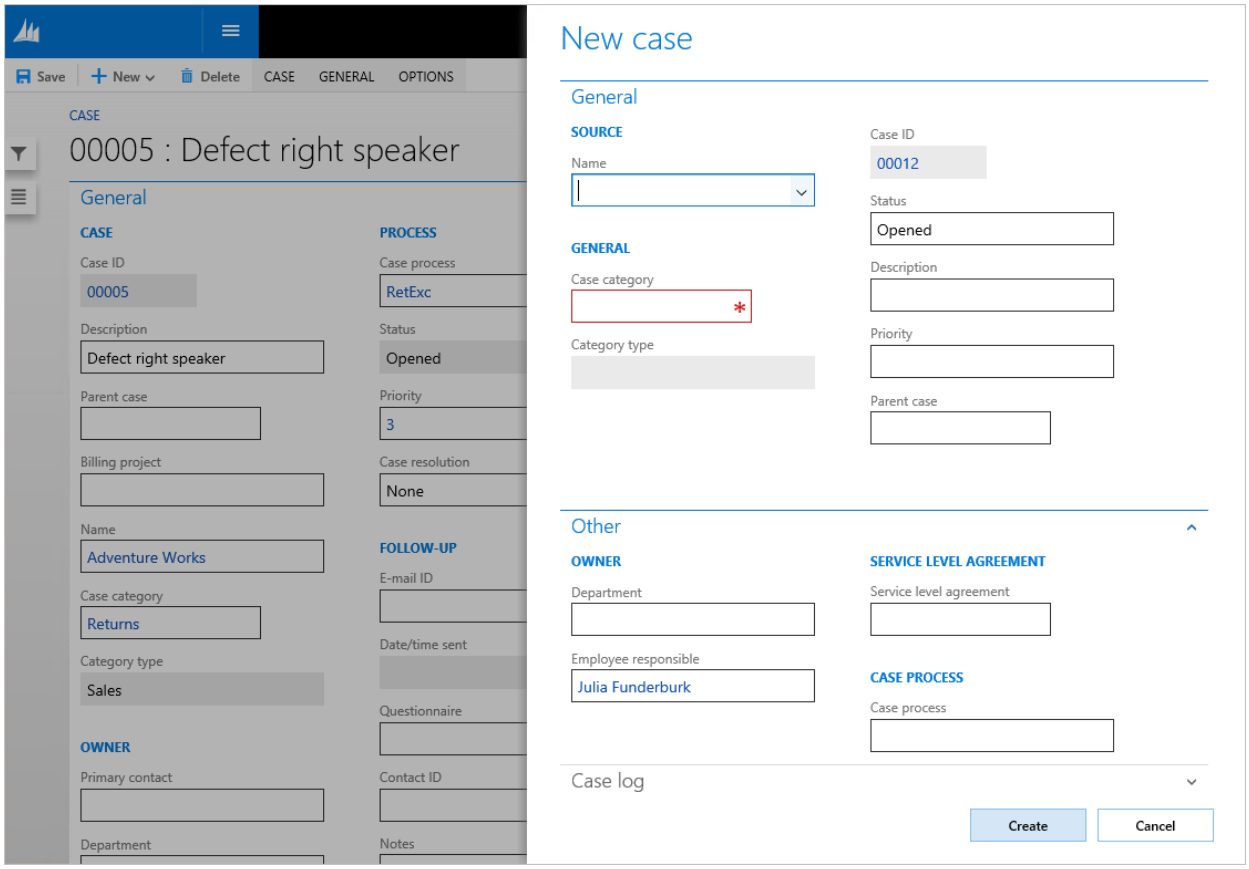
Dialog w/tabs

Form: **CaseDetailCreate** (Click **Common** > **Common** > **Cases** > **All cases**, and then click **New**.)



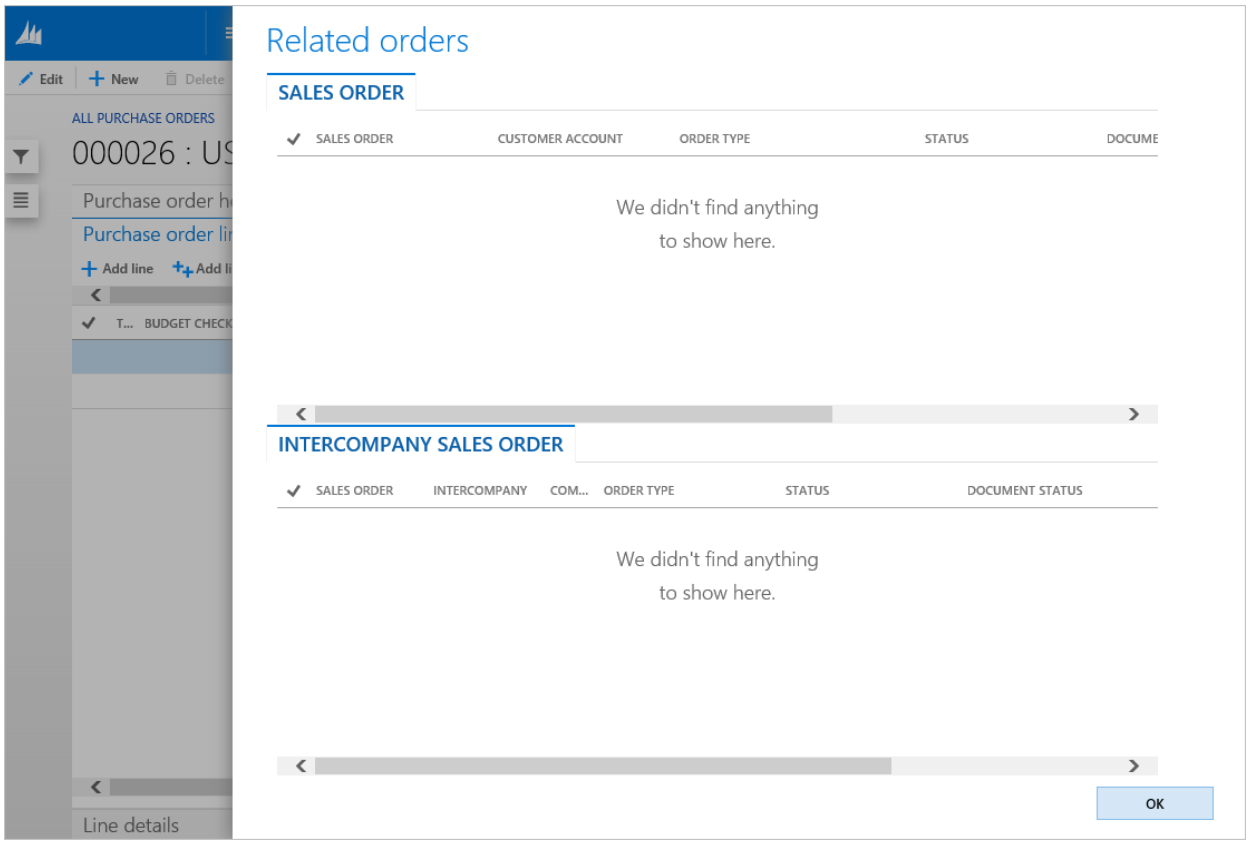
Dialog w/FastTabs

This example shows a modified version of the **CaseDetailCreate** form, because the product currently includes no examples of forms that use this pattern.



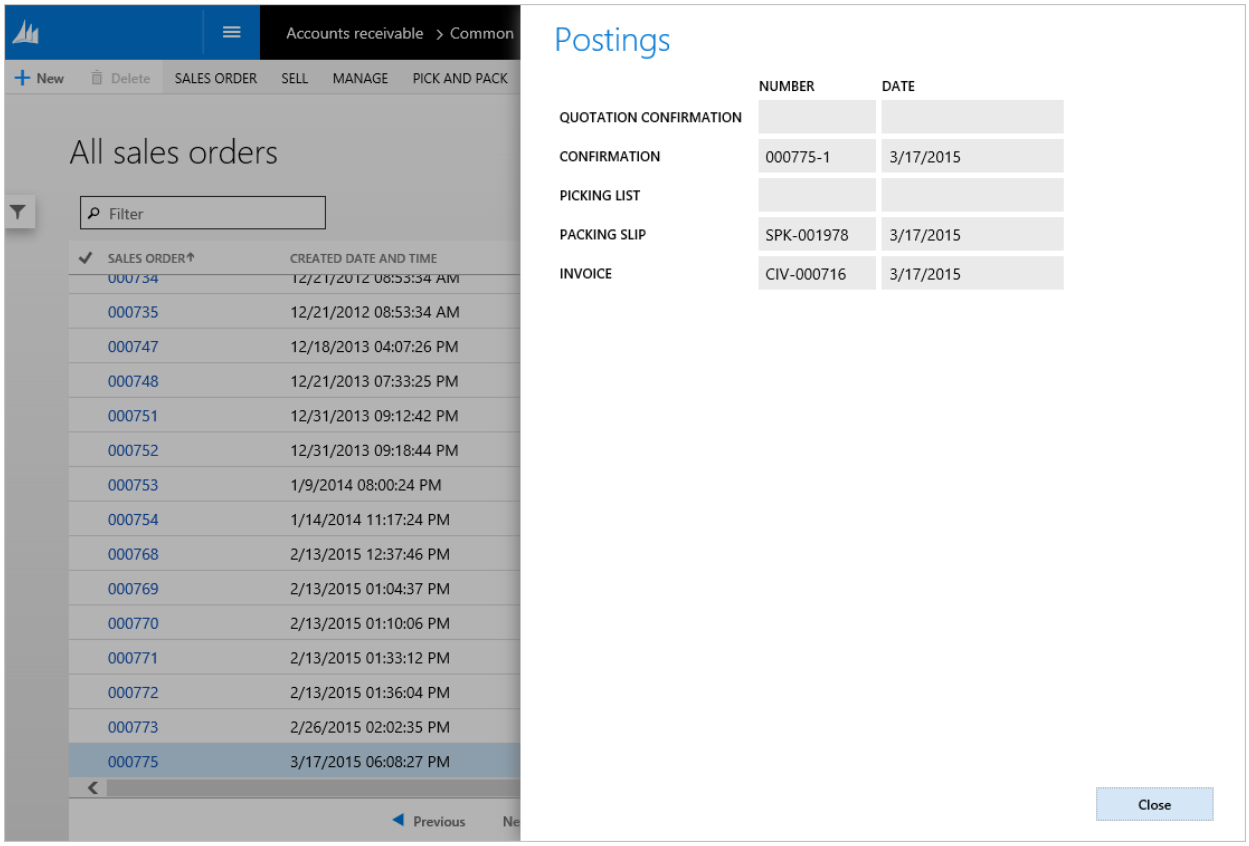
Dialog w/double tabs

Form: **PurchTableReferences** (Click **Accounts payable** > **Common** > **Purchase orders** > **All purchase orders**, and then click **General** > **Related information** > **Related orders**.)



Dialog (read only)

Form: **SalesTablePostings** (Click **Accounts receivable** > **Common** > **Sales orders** > **All sales orders**, and then click **General** > **Related information** > **Postings**.)



Appendix

Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues

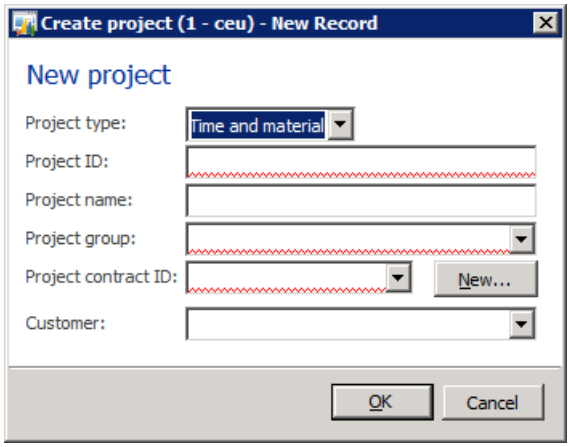
- **How does this pattern handle the More info link in dialogs?**
  - **More info** usage is assumed to be a custom pattern unless we have enough cases to justify the addition of a new pattern.
- **Should the pattern be modified to force OK/Cancel buttons to use CommandButtons instead of any button type?**
  - We will be looking at making this change in the future.

Selecting the correct dialog width

TYPE OF CONTENT	SMALL DIALOG	MEDIUM DIALOG	LARGE DIALOG	FULL DIALOG	NOTES
Columns of content	The slider fits one column of content.	The slider fits two columns of contents	The slider fits three columns of content.	The slider fits the viewport width minus peek.	The maximum number of columns depends on the width of the fields in the column. Therefore, the width is defined as *x* × 100% field size.
Horizontal scroll	No horizontal scrolling	Avoid horizontal scrolling.	Avoid horizontal scrolling.	OK, provided that the control buttons and commit buttons are visible	
Vertical scroll	No vertical scroll for typical scenarios (FastTabs can be expanded for special cases). Otherwise, use a Medium dialog.	Yes	Yes	Yes	Avoid putting so much content in the dialog that you cause vertical scrolling of the contents. If your dialog is vertically scrolling at a typical screen resolution, you should make the dialog larger.
FastTabs	Strongly discouraged	OK but discouraged	Yes	Yes	
Tabs	Yes	Yes	Yes	Yes	Switching tabs or expanding FastTabs should never cause jumps in the dialog size. The largest tab content must define the choice of the dialog size.

TYPE OF CONTENT	SMALL DIALOG	MEDIUM DIALOG	LARGE DIALOG	FULL DIALOG	NOTES
List/hierarchy	Yes	Yes	Yes	Yes	
Field groups	Yes	Yes	Yes	Yes	
Nested field groups	No nested field groups that have a mixed layout direction (matrix or tabular layout)	Multiple-column layout	Multiple-column layout	Yes	
Custom controls	Yes	Yes	Yes	Yes	
Grid	Yes, but no horizontal scrolling	Yes, but no horizontal scrolling	Yes, but no horizontal scrolling	Yes	The maximum number of columns depends on the width of the fields in the column. Therefore, the width is defined by *x* × 100% field size.

AX 2012 content



Drop Dialog form pattern

2/18/2021 • 4 minutes to read • Edit Online

This topic provides information about the Drop Dialog form pattern. This pattern is used to initiate actions when the number of fields is seven or fewer.

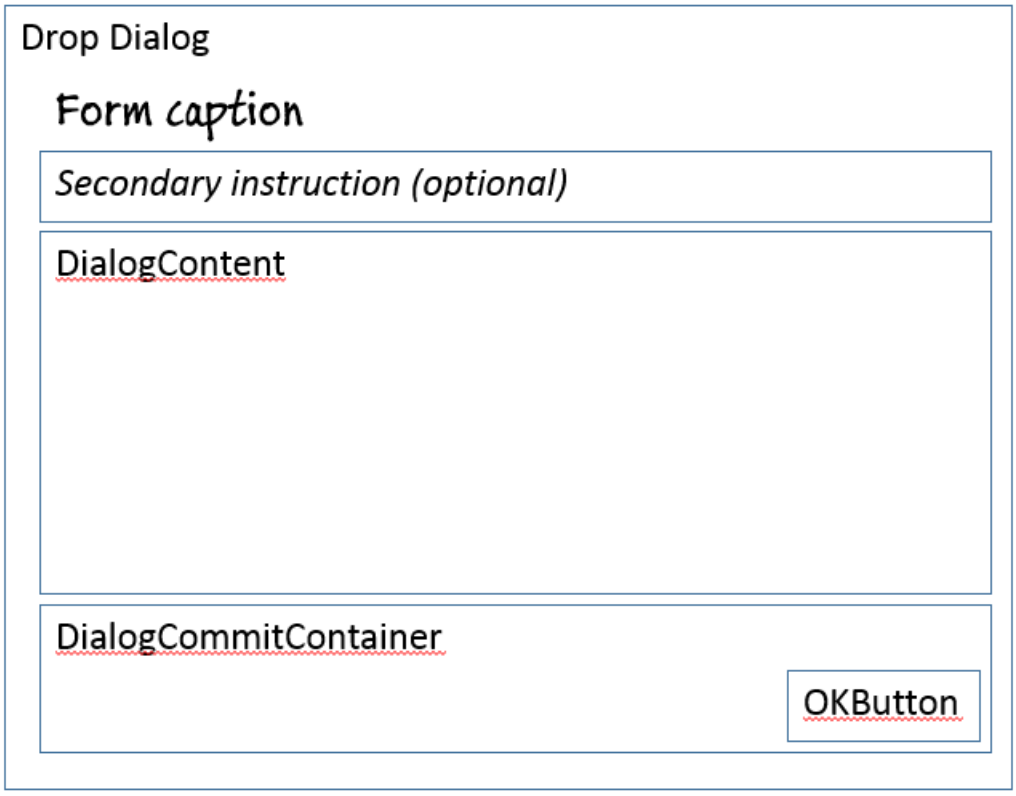
Usage

The Drop Dialog pattern is used to initiate actions when the number of fields is seven or fewer. Drop dialogs are quick and easy for users to use, and are more lightweight than a full dialog that is presented as a slider. Drop dialogs should feel as lightweight to use as a menu. Two patterns are described in this document:

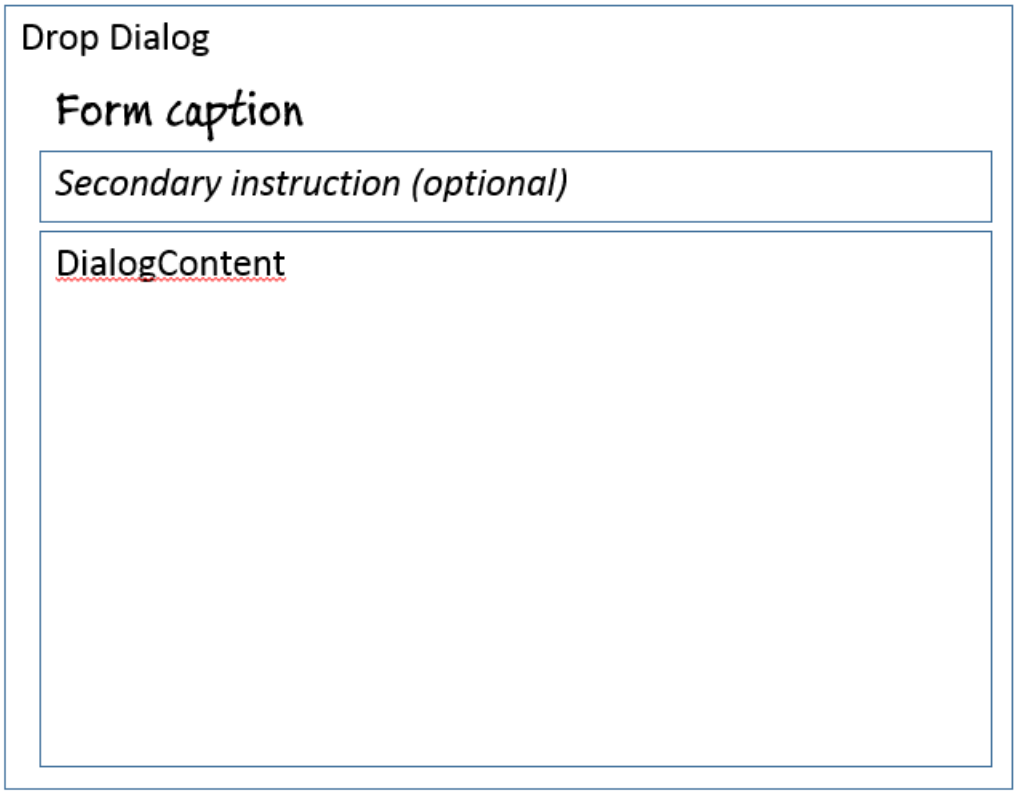
- **Drop dialog** – This is the basic Drop dialog pattern. If your Drop dialog is editable, this is the correct pattern to use.
- **Drop dialog (read only)** – This Drop dialog pattern is for informational forms that aren't editable. This variation doesn't have an **OK** button.

Wireframe

Drop dialog (basic)



Drop dialog (read only)



Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- Manual handling of error messages is no longer required.

Model

Drop dialog (basic) – High-level structure

- Design
  - *SecondaryInstruction (StaticText) [optional]*
  - DialogContent (Group)
  - DialogCommitContainer (ButtonGroup)
    - OKButton ($Button)

Drop dialog (read only) – High-level structure

- Design
  - *SecondaryInstruction (StaticText) [optional]*
  - DialogContent (Group)

Core components

- Apply the Drop Dialog pattern on **Form.Design**.
- Address BP Warnings:
  - **Design.Caption** isn't empty.
  - The form must be referenced by at least one menu item.
  - **StaticText.Text** isn't empty.

Related patterns

- Dialog

Commonly used subpatterns

- Fields and Field Groups
- Toolbar and List

UX guidelines

The verification checklist shows you the steps for manually verifying that the form complies with UX guidelines. This checklist doesn't include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps. **Standard form guidelines:**

- Standard form guidelines have been consolidated into the General Form Guidelines document.

Drop dialog guidelines:

- A Drop dialog should be used if the following conditions exist:
  - There are seven or fewer fields.
  - The user can enter the information quickly.
  - Minimal field validation is required.
  - There are no buttons that open additional child forms.
    - **Exceptions:** Lookups, Enhanced preview, and View details navigation
  - There is no editable grid (select-only grids are allowed).
- Focus should be in the first editable field on the Drop dialog when it is first opened.
- A Drop dialog should have a **main instruction** (form caption) at the top.
  - The main instruction should be used to explain concisely what the user should do in the Drop dialog. The instruction should be a specific statement, an imperative direction, or a question. Good instructions communicate the user's objective with the Drop dialog rather than focusing purely on the mechanics of manipulating it.
  - A final period should not be included if the main instruction is a statement. If the instruction is a question, a question mark should be included.
  - Besides the main instruction, a secondary instruction to the user should be displayed, and it should present additional information that will help the user understand or use the Drop dialog. The secondary instruction should consist of a complete sentence in sentence case and should have end punctuation.
    - **Exception:** If the additional instruction merely repeats the main instruction with slightly different wording, don't include it.
- A Drop dialog should have a **content** area.
  - Constrained input controls should be used to avoid validation errors. Examples include selection lists, check boxes, radio buttons, and command links.
  - Reasonable defaults for each input should be provided whenever possible.
- A Drop dialog should have a **commit** button area that:
  - Does **not** have a **Cancel** button.
  - Has a button that is marked as the default button of the Drop dialog (if a button exists).
  - The label of the default button should be a verb that implements the action that is described in the main instruction. For example, if the main instruction is "Create new product," the button label should be **Create**. If there is no appropriate verb for the button, use **OK**.
  - The commit button area should have specific commit button labels that make sense on their own and are a response to the main instruction.

A Drop dialog should **not** have the following:

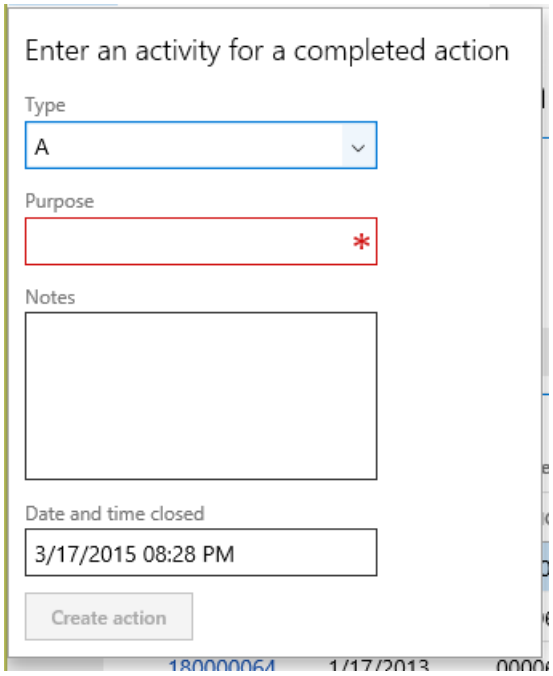
- A toolbar or ActionPane anywhere in the Drop dialog.

- Buttons that navigate to another page or open other dialogs. (Enhanced previews are allowed.)
- Field groups. There are exceptions, such as a radio button or check box group.
- A tab control.
- FactBoxes.
- FastTabs.

Examples

Drop dialog (basic)

Form: **CustCollectionsNewActivityAction** (Click **Accounts receivable** > **Common** > **Collections** > **Collections**, select a row to move to details, and then click **Action**.)



Drop dialog (read only)

This pattern isn't currently used in the product.

Appendix

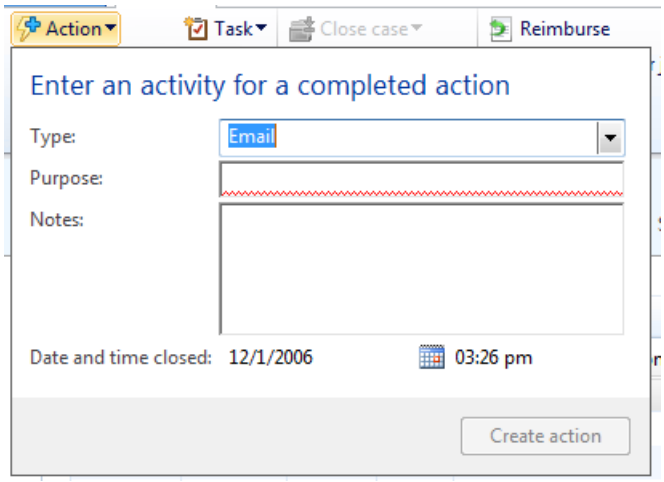
Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues

- **Should a vertical Fields and Field Groups subpattern be added for Drop dialogs?**
  - No, you should use the normal Fields and Field Groups pattern.
- **Should buttons be left-aligned or right-aligned?**
  - Right-aligned. The pattern is currently enforcing this.

AX 2012 content



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Lookup form pattern

2/18/2021 • 3 minutes to read • Edit Online

This topic provides information about the Lookup form pattern. Custom lookup forms should be used when a standard framework-provided lookup would not provide the correct data, or when advanced visualization of the data is required.

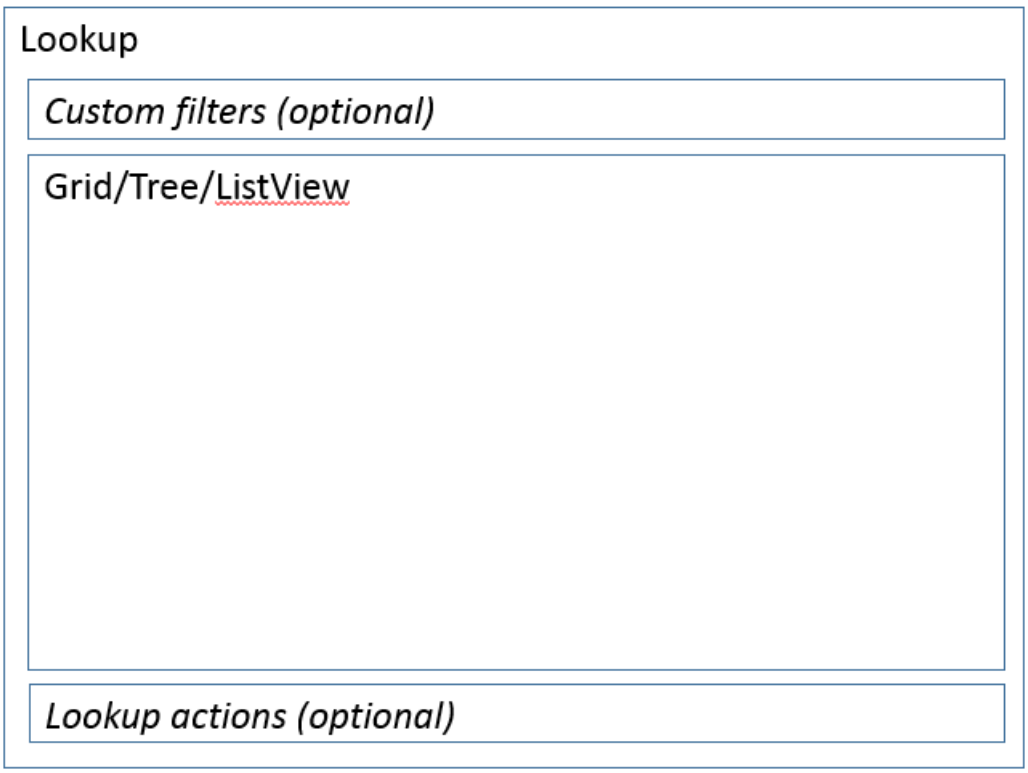
Usage

Custom lookup forms should be used when a standard framework-provided lookup (which is typically generated by using the AutoLookup field group that is defined on the table definition), would not provide the correct data, or when advanced visualization of the data is required. Three patterns are described in this document:

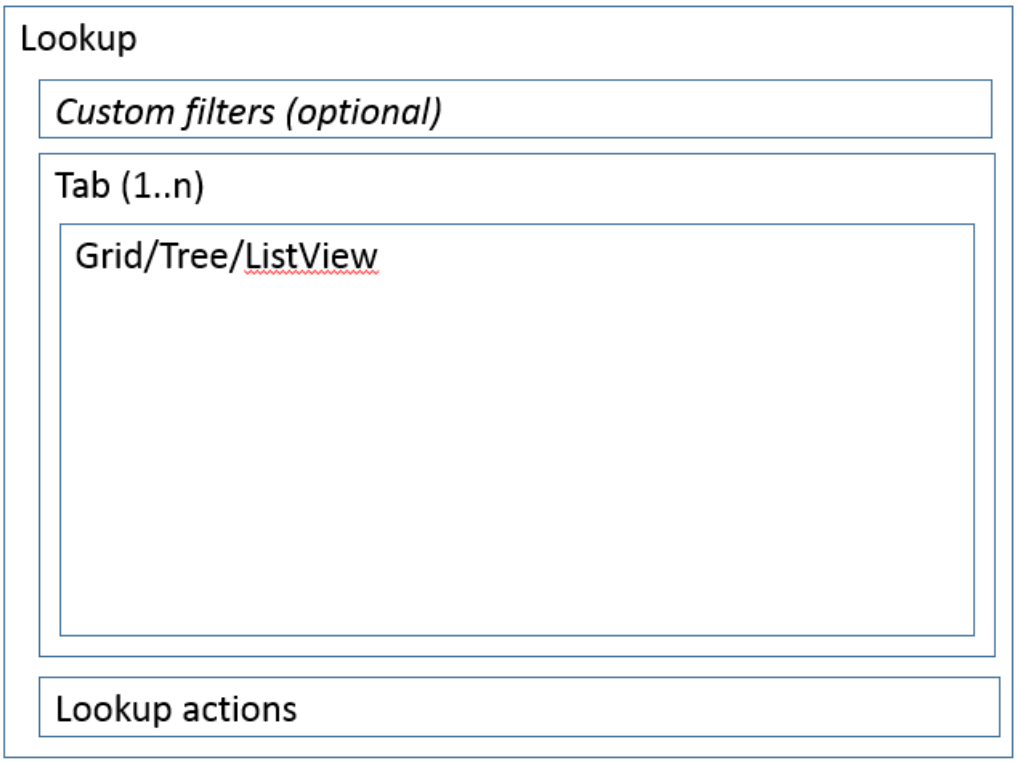
- **Lookup basic** – This is the basic Lookup pattern that has just one list or tree, and also optional custom filters and actions.
- **Lookup w/tabs** – This Lookup pattern is used when more than one view of the lookup can be made available to the user. Tab captions aren't shown. Instead, the tab is selected through a combo box.
- **Lookup w/preview** – This more advanced Lookup pattern enables a preview of the current record in the lookup grid.

Wireframe

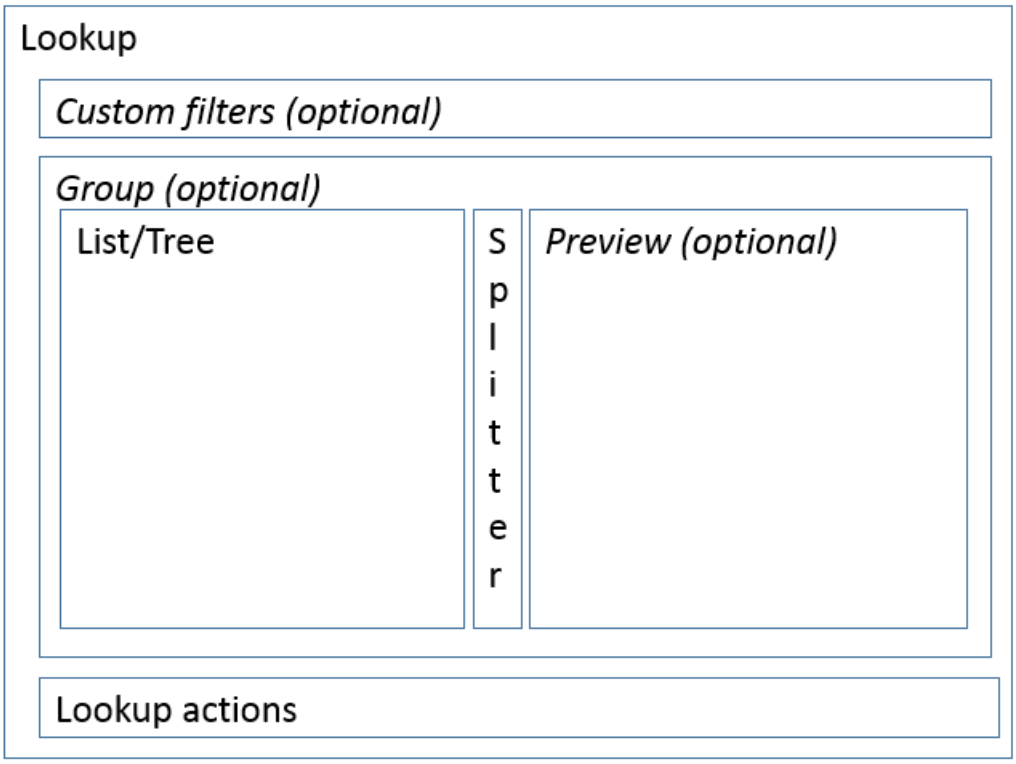
Lookup basic



Lookup with tabs



Lookup with preview



Pattern changes

Here are the changes to this pattern since Microsoft Dynamics AX 2012:

- Tabs should be hidden and controlled by a combo box.
- Optionally add a splitter/preview.

Model

Lookup basic – High-level structure

- Design
  - *CustomFilter (Group) [Optional]*
  - Grid | Tree | ListView
  - *LookupActions (Group) [Optional]*

Lookup w/tabs – High-level structure

- Design
  - *CustomFilter (Group) [Optional]*
  - LookupTab (Tab)
    - LookupTabPage (TabPage, repeats 1..N)
      - Grid | Tree | ListView
  - *LookupActions (Group) [Optional]*

Lookup w/preview – High-level structure

- Design
  - *CustomFilter (Group) [Optional]*
  - LookupContent (Group)
    - Grid | Tree | ListView
  - VerticalSplitter (Group)
    - Preview (Group)
  - LookupActions (ActionPane)

Core components

- Apply the Lookup pattern on Form.Design.
- Address BP Warnings:
  - EDT.FormHelp must reference a form where Style=Lookup.

Commonly used subpatterns

- Custom Filter Group

UX guidelines

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn't include any guidelines that will be enforced automatically through the development environment. Open the form in a browser, and walk through these steps. **Standard form guidelines**

- Standard form guidelines have been consolidated into the General Form Guidelines document.

Lookup guidelines

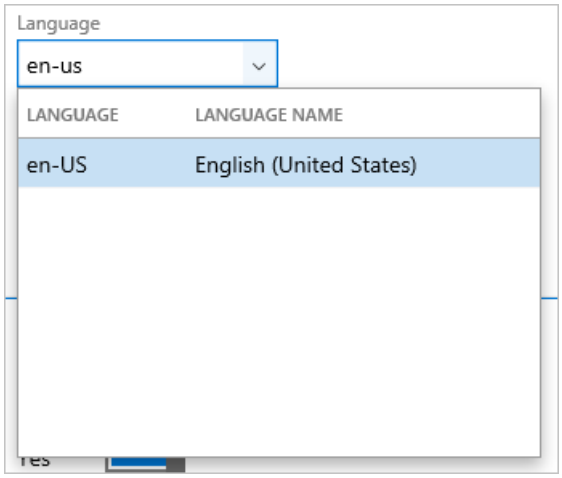
- **Grid** guidelines have been consolidated into the General Form Guidelines document, in the Grid guidelines section.
- If you must show different "views" (tabs) within the lookup, use a combo box to let the user to switch between tabs.
- You can optionally use a tree view in the lookup. Also consider providing a standard grid because of the complexity that is involved in showing additional fields of data in a tree.
- Don't have more than five columns in the grid. The lookup resizes to show all columns, so five columns is very wide.

- The **optional preview area**:
  - The area should help the user choose between two or more records that are similar. For example, if you have two employees who are named John Smith, the preview should provide enough information to help the user differentiate these two people.
  - Don't show editable fields in the preview.
- **Custom filter** guidelines have been consolidated into the Custom Filter Group subpattern document.

Examples

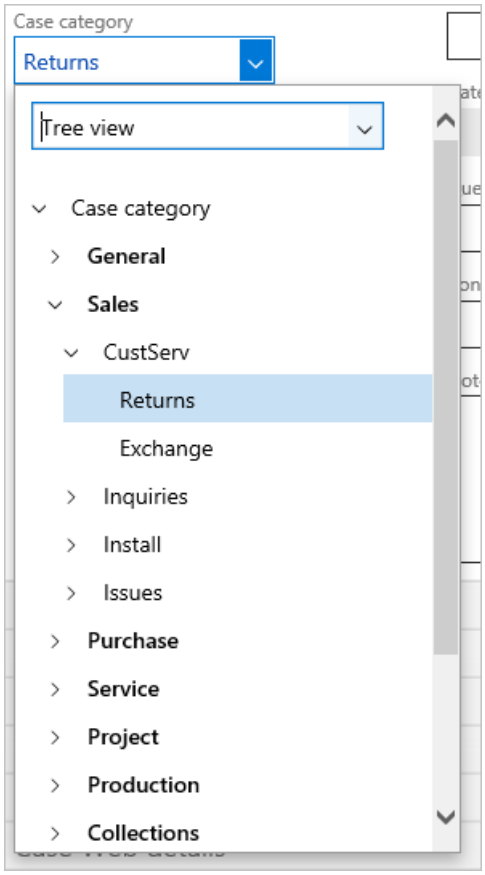
Lookup basic

Form: **SysLanguageLookup** (Click **Settings** > **User settings** on the navigation bar.)



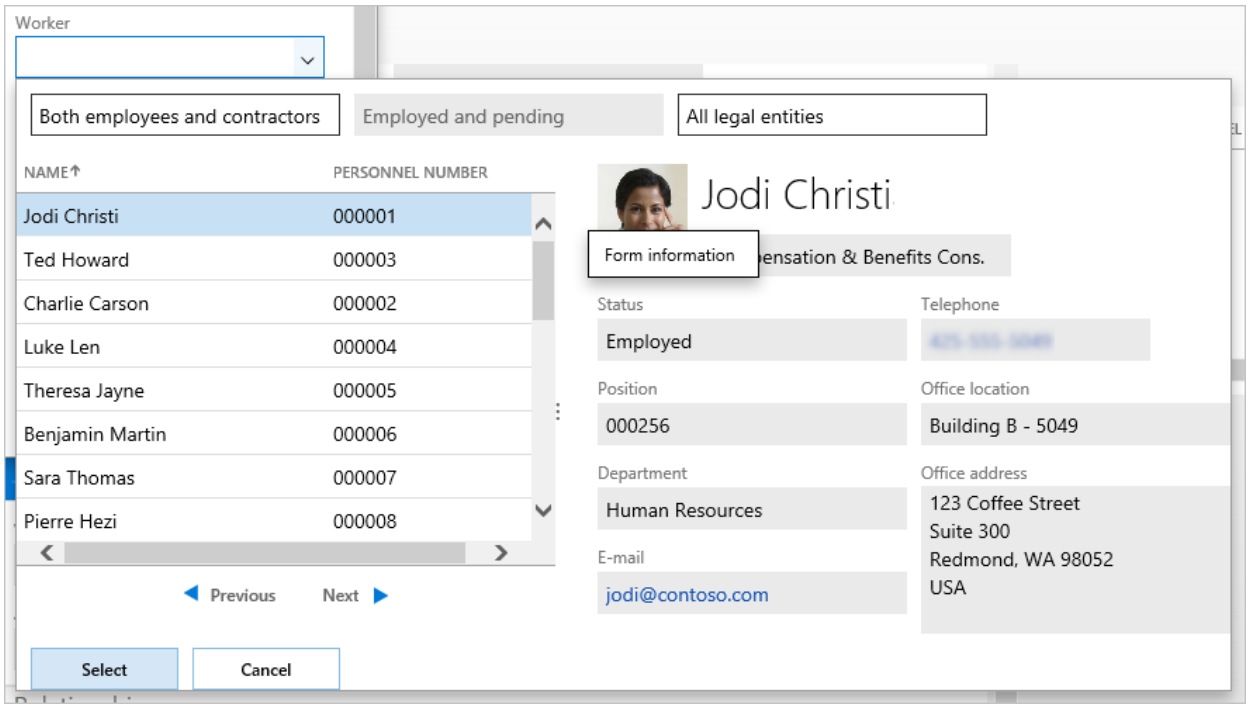
Lookup with tabs

Form: **CaseCategoryLookup** (Click **Common** > **Common** > **Cases** > **All cases**, and then select a case to go to the details.)



Lookup with preview

Form: **HcmWorkerLookup** (Click **Human resources** > **Common** > **Organization** > **Positions** > **Positions**, and then click a record to go to the details. Expand the **Worker assignment** FastTab, click **New**, and then click the drop-down arrow in the **Worker** field.)



Appendix

Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

- **How do I switch between tabs in the Lookup w/tabs pattern?**
  - The Lookup w/tabs pattern intentionally sets **ShowTabs** to **No** on the Tab control. These forms are meant to model an unbound combo box in the custom filter group. This combo box is used instead of tab headers to switch tabs.
  - To do this, follow these steps:
    1. Call the **SysLookup::tab2ComboBox** method post **super** in form **run()** to populate the combo box with captions from visible tabs in the lookup.

       ```
       // Generate view combobox based on tabs
       tab2ComboBoxItemMap = SysLookup::tab2ComboBox(Tab, switchView);
       ```

    2. Override **modified()** on the combo box to update the visible tab, based on the selected value in the combo box.

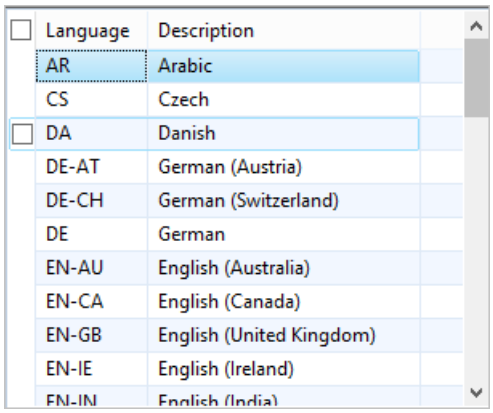
       ```
       Tab.tabChanged(Tab.tabValue(), tab2ComboBoxItemMap.lookup(this.selection()));
       ```

Open issues

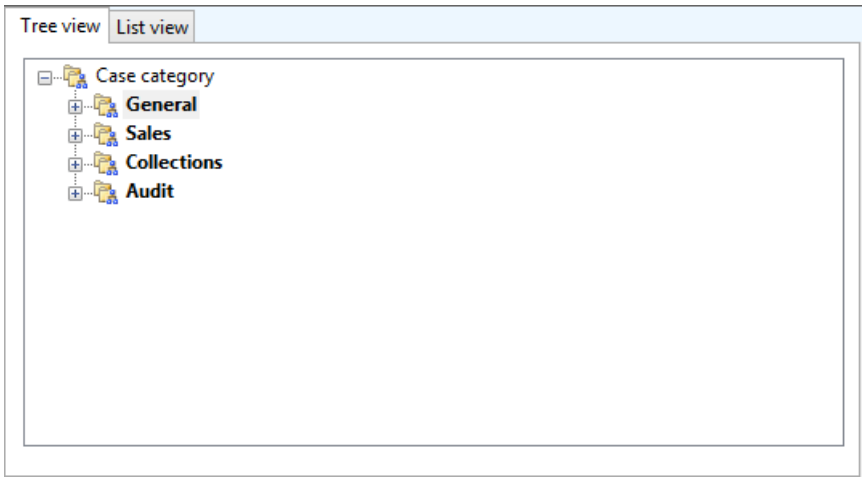
- **Can we incorporate the most recently used values into lookups?**
  - App modeling can make this work right now (for example, the Currency lookup). We are considering general framework support for this feature in the future.

AX 2012 content

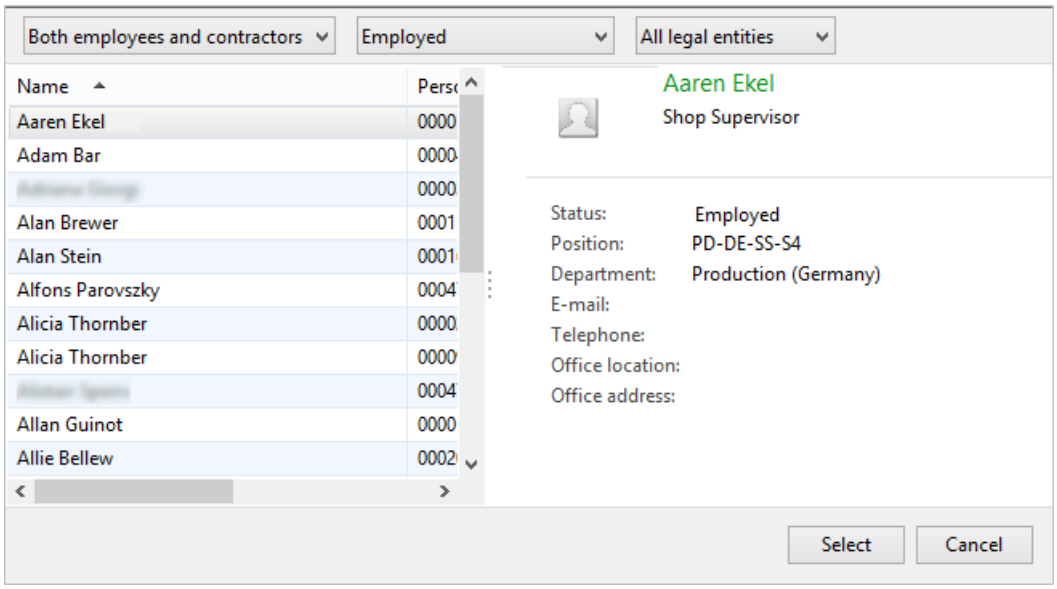
SysLanguageLookup (Lookup basic)



CaseCategoryLookup (Lookup with tabs)



HcmWorkerLookup (Lookup with preview)



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

FactBox form patterns

2/18/2021 • 3 minutes to read • Edit Online

This topic provides information about the FactBox form patterns. FactBoxes are used to provide related information for a record.

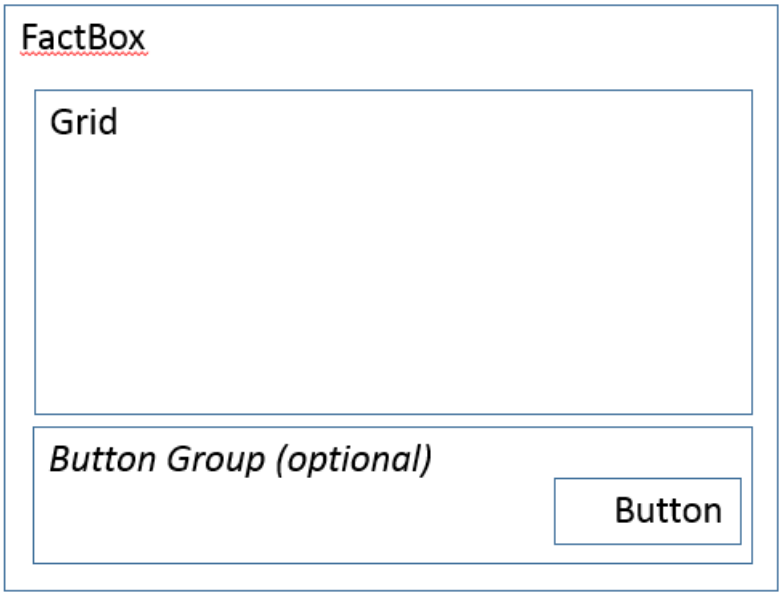
Usage

In general, FactBoxes are used to provide "related information" for a record. They help guarantee that the user doesn't have to open additional forms to get important information, such as totals, balances, overdue orders, and email addresses. The Factbox Grid pattern should be used when there is a child collection (potential for multiple rows) of related information. Two patterns are described in this document:

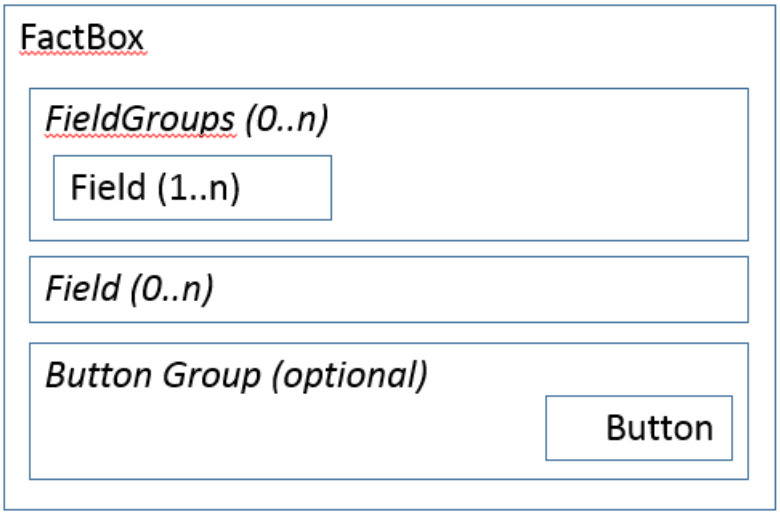
- **Form Part FactBox Grid** – This FactBox pattern is used when there is a child collection (potential for multiple rows) of related information.
- **Form Part FactBox Card** – This FactBox pattern is used when there is just a set of related fields that must be shown.

Wireframe

Form Part FactBox Grid



Form Part FactBox Card



Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- A group has been added around the optional button to make it easier to position the button.

Model

Form Part FactBox Grid – High-level structure

- Design
  - Grid
  - *GridDefaultAction (Button) [Optional]*
  - *ButtonGroup (ButtonGroup) [Optional]*
    - Button

Form Part FactBox Card – High-level structure

- Design
  - *FieldGroups (Group) [0..N]*
    - Fields ($Fields, 1..N)
  - *Fields ($Field) [0..N]*
  - *ButtonGroup (ButtonGroup) [Optional]*
    - Button

Core components

- Apply the FactBox pattern on **Form.Design**.
- Address BP Warnings:
  - **Design.Caption** isn't empty.
  - **Grid.DataSource** isn't empty.

UX guidelines

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn't include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps. **Standard form guidelines:**

- Standard form guidelines have been consolidated into the General Form Guidelines document.

FactBox general guidelines:

- If a backing form exists, the FactBox should have a **(More...)** link defined that goes to the appropriate backing form. The names of the FactBox and backing form should be similar.
- The title should not be a verb or a verb phrase.
- The title should not contain a label to a specific record.
- FactBoxes should not display fields that let a user enter data by typing with the keyboard.
- The title should accurately describe the content and should not be truncated when the FactBox area is at its default size.

FactBox grid guidelines:

- One to four columns should be displayed.

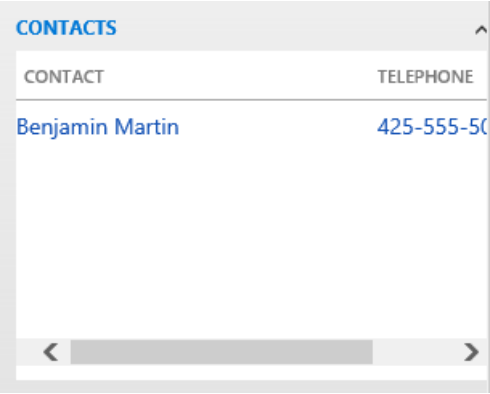
FactBox card guidelines:

- Each field should have a label.
- The ID and name of the header or the line that content is displayed for in the FactBox should not be displayed.
- Two to ten fields should be displayed.
- Currency indicator fields should be displayed as the last field in the FactBox.

Examples

Form Part FactBox Grid

Form: **CustTable** > **ContactsInfoPart**



Form Part FactBox Card

Form: **CustTable** > **CustStatisticsStatistics**

Appendix

Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

- **How do I make the More button work?**
  - The **More** button at the bottom of the FactBox takes the user to a backing form that contains the full list of related records. This button should be implemented by using a regular Button control that overrides the **clicked** method as shown in the following example. Be sure to fill in the **TableRef** and **ListPageRef** properties on the table that provides data for the grid.

```
[Control("Button")]
class More
{
public void clicked()
    {
        super();
        FormPartUtil::openShowMoreForm(element, <TableName>);
    }
}
```

Open issues

- **Should field labels be on the left side in FactBoxes to support a more compact visual?**
  - We plan to allow **LabelPosition**=**Left** inside FactBoxes.

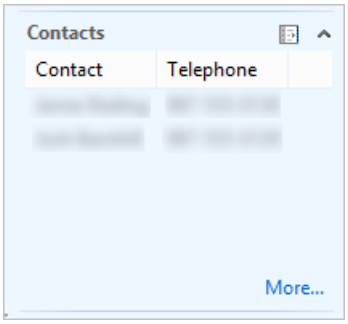
AX 2012 content

AX 2012 links

- AX 2012 MSDN List Page Guidelines (including FactBoxes)

AX 2012 example

CustTable > ContactsInfoPart



Custom Filter Group subpattern

2/18/2021 • 3 minutes to read • Edit Online

Usage

This subpattern is used to show a small collection of input controls (no more than five) that apply a custom filter to a grid or form section. Fields in the Custom Filter Group should be limited to the following field types, which have constrained inputs and can be applied to the query:

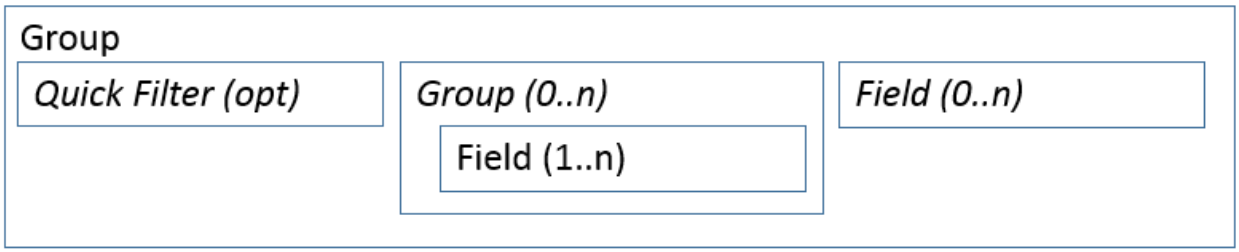
- StringEdits with Lookups
- Date fields
- ReferenceGroup
- Comboboxes
- Checkboxes
- Quick Filter

Two patterns are described in this document. The only difference between these patterns is whether the Quick Filter control is mandatory or optional:

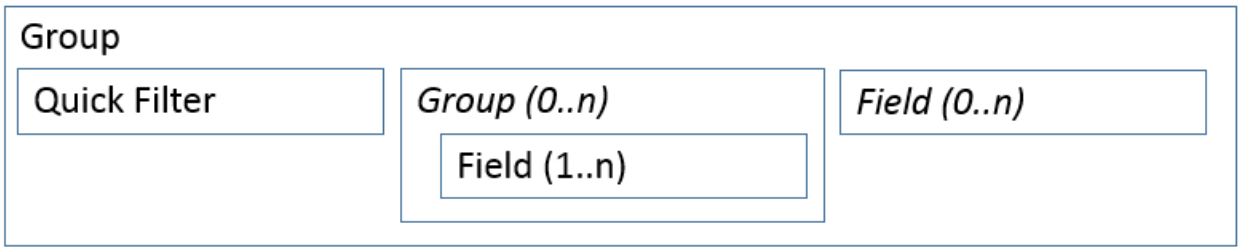
- **Custom Filters** – In this subpattern, the QuickFilter control is optional.
- **Custom and Quick Filters** – In this subpattern, the QuickFilter control is mandatory.

Wireframes

Custom Filters



Custom and Quick Filters



Model

Custom Filters – High-level structure

- CustomFilter (Group)
  - *QuickFilter (QuickFilter) [Optional]*
  - *FieldGroups (Group) [0..N]*

    - Fields ($Field) [1..N]
  - *Fields ($Fields) [0..N]*

Custom and Quick Filters – High-level structure

- CustomFilter (Group)
  - QuickFilter (QuickFilter)
  - *FieldGroups (Group) [0..N]*
    - Fields ($Field) [1..N]
  - *Fields ($Fields) [0..N]*

Core components

- Apply a custom filter-container pattern to a Group control.
- Address BP Warnings:
  - Input controls within the CustomFilterGroup should not have a DataSource or DataField assigned.

Related container patterns

None

Related modeling

Use QueryFilter, not QueryBuildRange, for all custom filters. QueryBuildRange doesn't work correctly with outer-joined fields.

UX guidelines

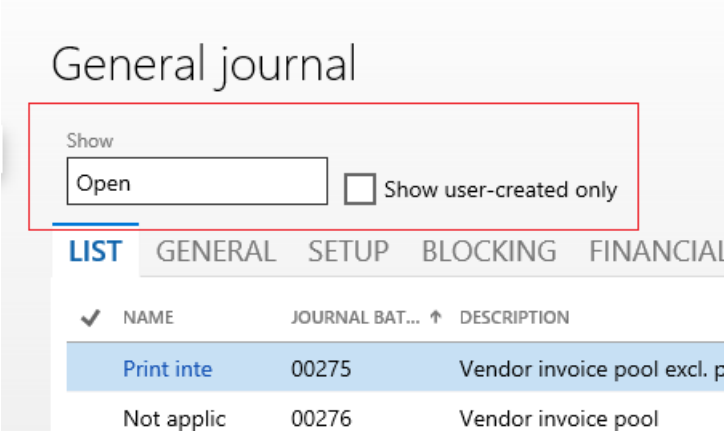
The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn't include any guidelines that will be enforced automatically through the development environment. Open the form in a browser, and walk through these steps.

- **Standard form guidelines:**
  - General form guidelines
- **Custom Filter Group guidelines:**
  - All controls (except QuickFilter) are constrained input controls. Open-ended controls, such as strings, integers, and reals, should not be used.
  - Field labels are turned off to save space. For example, no label is needed for a combobox that has the values **Open**, **Closed**, **Posted**, and All.
    - Show labels when the filter values do not provide sufficient context for the user to understand what the filter does. For example, a date field provides no context, and the user requires a label to understand what type of date is specified (for example, **Created date**).
    - Either all labels are turned off, or all labels are turned on. Don't mix unlabeled filters and labeled filters.
    - **Exception:** When you use a check box–style Boolean, the label can be left on, even though other fields don't show a label.
  - There should not be more than five controls in the custom filter group.

Examples

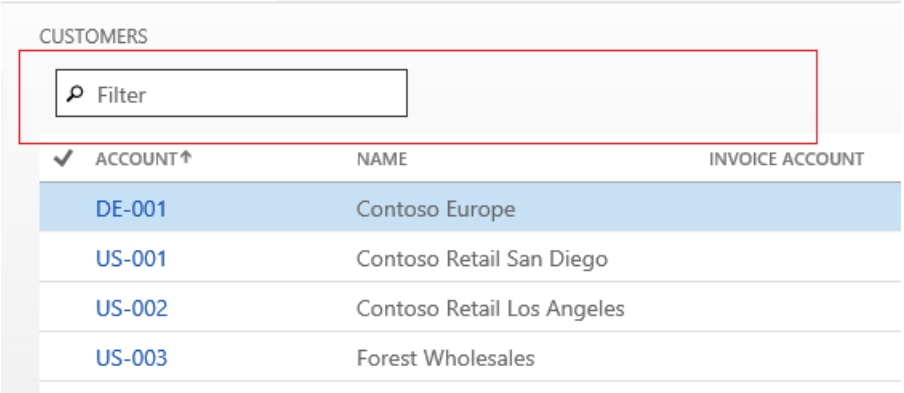
Custom Filters

Form: **LedgerJournalTable (TopFields)**



Custom and Quick Filters

Form: **CustTable (CustomFilterGroup)**



Resources

Typically used by form patterns

- Simple List
- Details Master
- Details Transaction
- List Page

Appendix

Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

- **What do I do with the legal entity?**
  - "Legal entity" is a typical custom filter that belongs in the Custom Filter Group.
- **Should the custom filter be above the toolbar of a list?**
  - We think that the custom filter belongs as close to the grid as possible, because it more directly affects the list, just as the commands in the toolbar do. Additionally, this position makes the logical order of the elements consistent across different page patterns.

Open issues

- **Do we allow Show More/Less in the Custom Filter Group? An example is BudgetAnalysisInquiry_PSN.**
  - No, that will currently be a custom container. If we have enough examples, we might add a new container subpattern.
- **Does the pattern limit the possible input types to those that allow constrained input values?**

- The pattern currently allows any input, but it's against guidelines to have inputs that have unconstrained values.

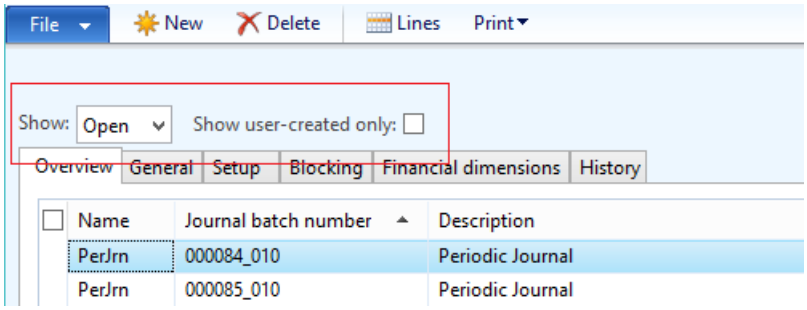
- **Do we allow groups to be used as custom filter groups?**
  - We do allow them now, to make migration easier and to identify Custom Filter group locations. However, we encourage you to use only a small set of fields in these situations (and we might eventually enforce this).

AX 2012 content

AX 2012 links

- MSDN AX 2012 How to Add Controls to the Filter Pane
- MSDN AX 2012 List Page Overview – section Filter Pane

AX 2012 example



Dimension Entry Control subpattern

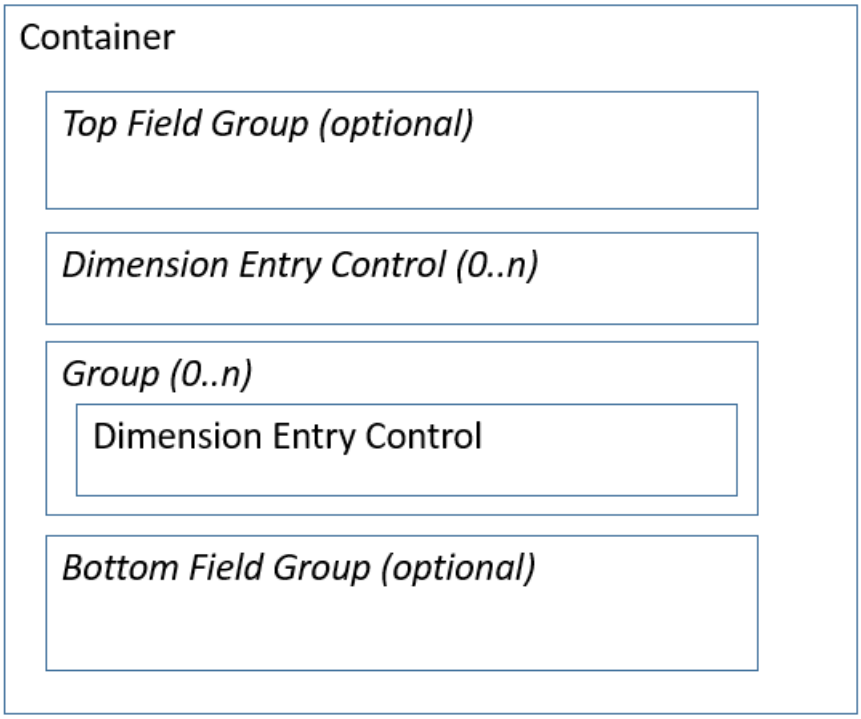
2/18/2021 • 2 minutes to read • Edit Online

This article provides information about the Dimension Entry Control subpattern. This subpattern is used when you have a group or tab page that uses the Dimension Entry control (DEC).

Usage

The Dimension Entry Control pattern is used when you have a group or tab page that uses the Dimension Entry control (DEC).

Wireframe



Model

High-level structure

TabPage | Group *TopFieldGroup (Group) [Optional]* – **Note:** A field subpattern is used.

DECGroup (Group) [0..N] Dimension Entry Control *Dimension Entry Control [0..N] BottomFieldGroup (Group) [Optional]* – **Note:** A field subpattern is used.

Core components

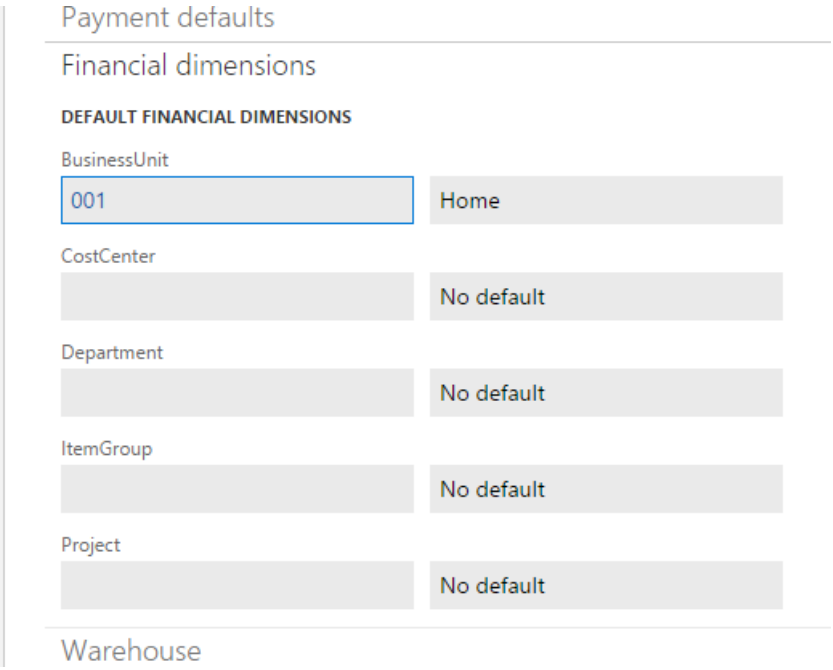
- Apply the Dimension Entry Control subpattern to the TabPage control.

UX guidelines

None.

Examples

Form: **CustTable (TabFinancialDimensions)**



Appendix

Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues

None.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Dimension Expression Builder subpattern

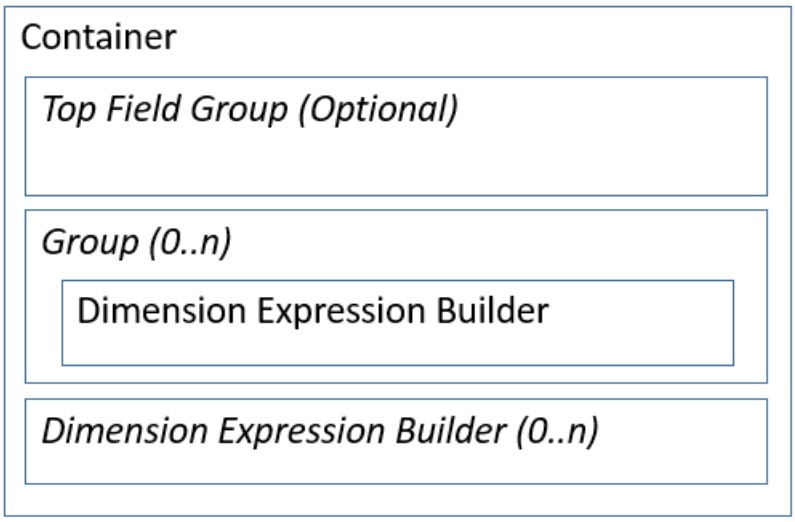
2/18/2021 • 2 minutes to read • Edit Online

This article describes the Dimension Expression Builder subpattern, which is applied to container controls that use the Dimension Expression Builder control.

Usage

The Dimension Expression Builder pattern is used when you have a group or tab page that uses the Dimension Expression Builder control.

Wireframe



Model

High-level structure

TabPage | Group

TopFieldGroup (Group) [Optional] – **Note:** A field subpattern is used.

DEBGroup (Group) [0..N]

Dimension Expression Builder

Dimension Expression Builder [0..N]

Core components

- Apply the Dimension Expression Builder subpattern to the TabPage or Group control.

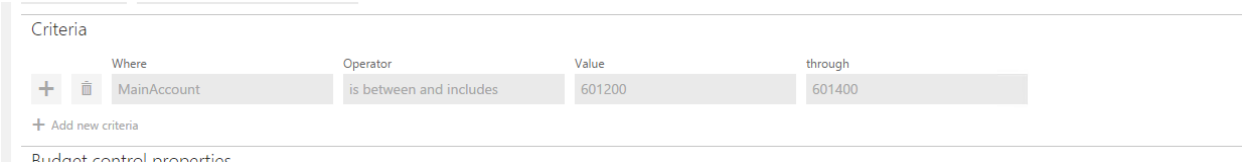
UX guidelines

None

Examples

Form: **BudgetControlConfiguration (RulesDetailsCriteriaFastTabPage)** (**Budgeting** > **Setup** > **Budget**

control > Budget control configuration)



Appendix

Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues

None

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Fields and Field Groups subpattern

2/18/2021 • 2 minutes to read • Edit Online

This topic provides information about the Field and Field Groups form subpattern. This is the most common data entry subpattern. It uses a dynamic number of columns to present multiple fields or groups of fields.

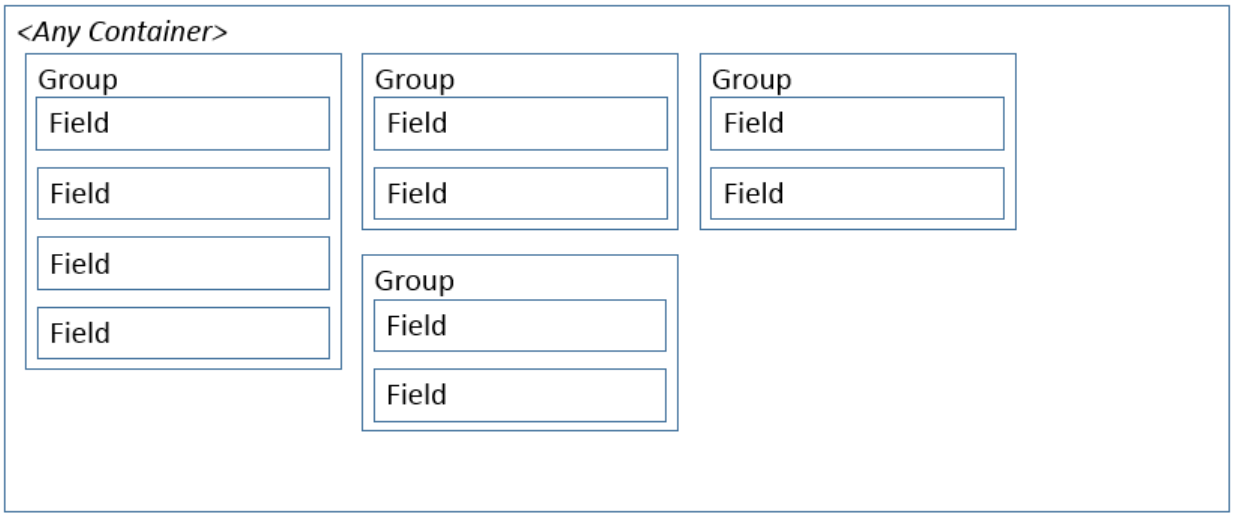
Usage

Field and Field Groups is the most common data entry subpattern and uses a dynamic number of columns to present multiple fields or groups of fields. This subpattern is not used with controls that have dynamic height or width (for example Grid, Tree, RadioButton, ListBox, or ListView), or controls that have larger height or width (for example, Chart). The group controls within this pattern can be used either to group fields under a label or to bind to a table field group.

Typical contents

- Groups or Fields as immediate children of the FastTab
- Groups containing Fields
- Can contain other subpatterns:
  - Horizontal fields and button group

Wireframe



Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- Removed explicit columns and the use of groups to force fields into two or three (or more) columns.
- Changed from fixed columns to dynamic columns.

Model

High-level structure

- [Container] (Columns=Fill)
  - *FieldGroups (Group) [0..N]*

    - Fields ($Field) [1..N]
    - *ActionableFields (Group) [0..N]* mimics the Horizontal Fields and Button Group subpattern
  - *Fields ($Field) [0..N]*
  - *ActionableFields (Group) [0..N]*

Core components

- Apply the FieldsAndFieldGroups subpattern to the container control.
- Address BP Warnings:
  - No additional BP checks are required beyond the AX6.3 BP that were checks carried forward.

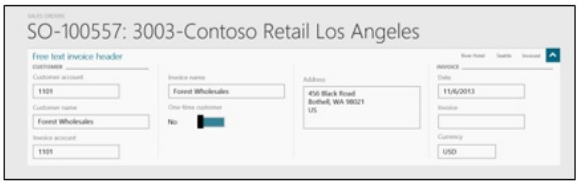
Related patterns

- Horizontal Fields and Buttons Group

UX guidelines

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn't include any guidelines that will be enforced automatically through the development environment. Open the form in a browser, and walk through these steps.

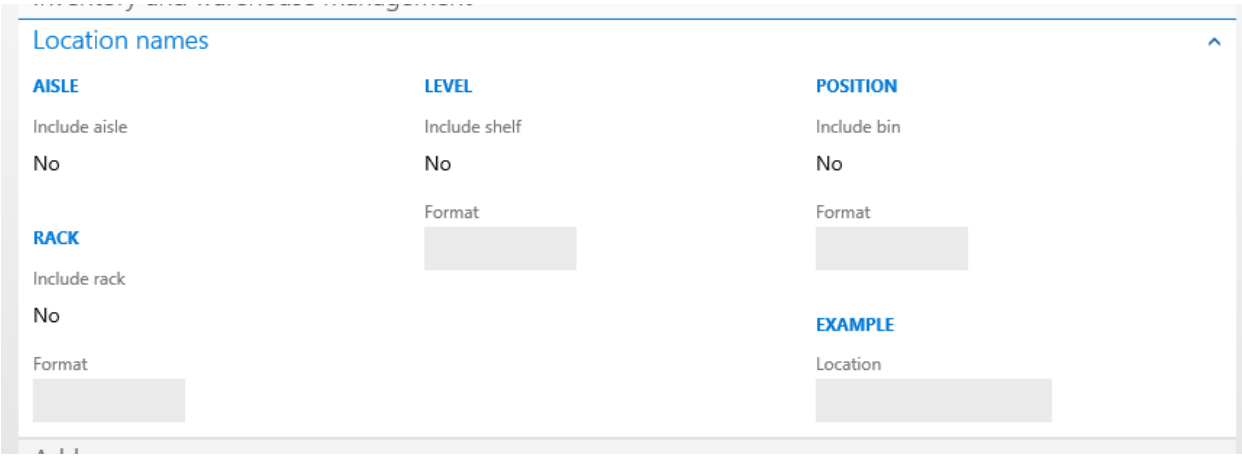
- **Standard form guidelines:**
  - Standard form guidelines have been consolidated into the General Form Guidelines document.
- **Fields and Field Groups guidelines:**
  - The fields in groups should flow across the entire page.



  - When possible, remove unnecessary field group labels.
  - Verify that you have an understandable grouping for your fields.
  - Either all fields should be in Groups that have labels, or no Group labels should be shown.

Examples

Form: **InventLocation (LocationNames)**



Resources

Typically used by patterns

- Simple List and Details
- Table of Contents
- Details Master
- Details Transaction

Appendix

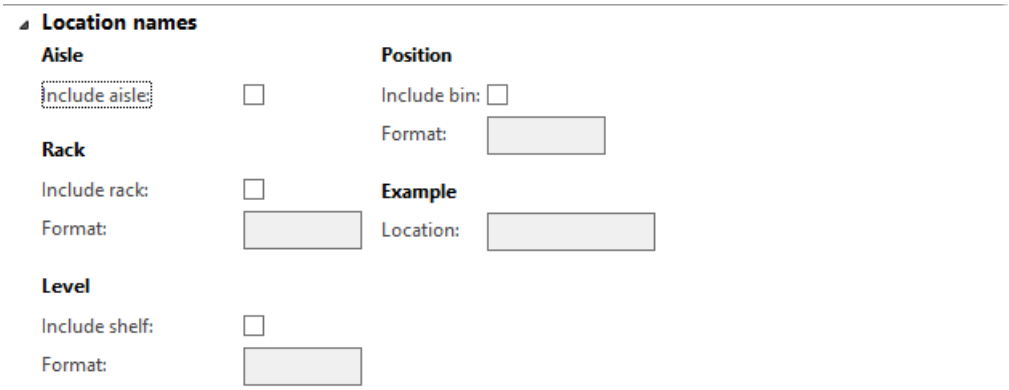
Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues

- Tooling must allow explicit use of the HorizontalFieldsButtonsGroup subpattern instead of mimicking content in the pattern definition.

AX 2012 content



InventLocation

Filters and Toolbar subpatterns

2/18/2021 • 3 minutes to read • Edit Online

This topic provides information about the Filters and Toolbar subpatterns. These workspace-specific subpatterns have been developed to show filters and/or actions inside panorama sections that host lists and charts.

Usage

The Filters and Toolbar subpatterns are workspace-specific subpatterns that have been developed to show filters and/or actions inside panorama sections that host lists and charts. Fields in the filtering parts of these subpatterns should be limited to the following field types. All these field types have constrained inputs and can be applied to the query.

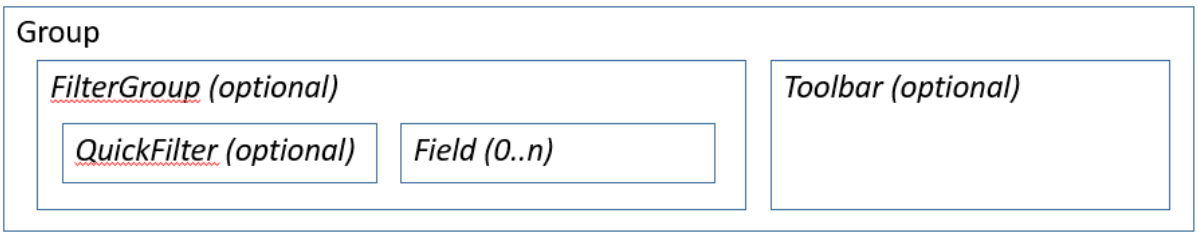
- StringEdits with Lookups
- Date fields
- ReferenceGroup
- Comboboxes
- Checkboxes
- Quick Filter

Two subpatterns are described in this article:

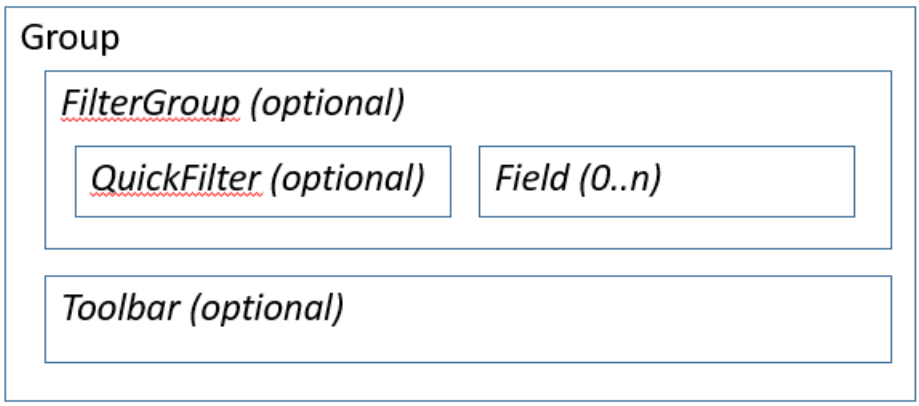
- **Filters and Toolbar - Inline** – In this subpattern, any defined actions appear on the same line as the filter fields.
- **Filters and Toolbar - Stacked** – In this subpattern, any defined actions appear on a separate line below the filter fields.

Wireframe

Filters and Toolbar - Inline



Filters and Toolbar - Stacked



Model

Filters and Toolbar - Inline: High-level structure

- Group (ArrangeMethod=HorizontalLeft)
  - *FilterGroup (Group) [Optional]*
    - *QuickFilter (QuickFilter) [Optional]*
    - *FilterFields ($Field) [0..N]*
  - *Toolbar (ActionPane) [Optional]*

Filters and Toolbar - Stacked: High-level structure

- Group (ArrangeMethod=Vertical)
  - *FilterGroup (Group) [Optional]*
    - *QuickFilter (QuickFilter) [Optional]*
    - *FilterField1 ($Field) [Optional]*
    - *FilterField2 ($Field) [Optional]*
  - *Toolbar (ActionPane) [Optional]*

Core components

Apply the correct Filters and Toolbar subpattern to the container control.

Related container patterns

- Form Part Section List
- Section Chart

UX guidelines

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn't include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps.

- Filters and Toolbar guidelines
  - The **Stacked** variant should be used over narrow lists and charts.
  - The **Inline** variant should be used over wider lists and charts.
- Filters
  - No more than two filter fields should be used in a Filter group. If you require more than two filter fields, an action on the Toolbar can be used to open a Drop Dialog that has more filter fields.
  - The filter fields should not have labels. The context should be obvious from the field value.

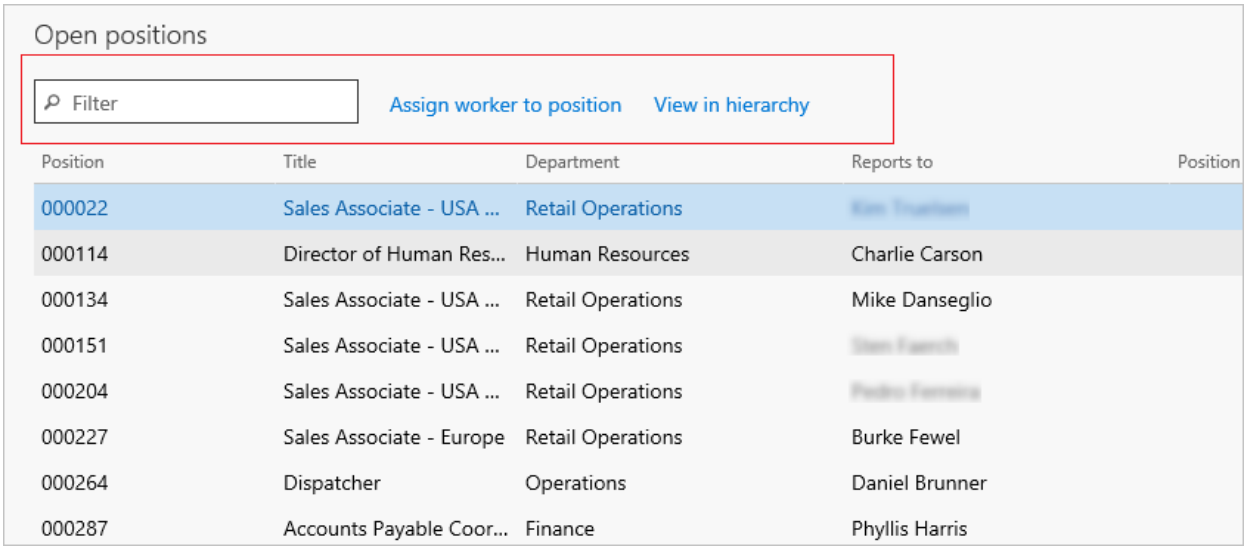
- The combined width of filter fields should not cause the section to become larger than the grid or chart in the section, and should not cause an extra scrollbar on the filters.

- Actions
  - Include only frequently used commands that help users complete tasks in the workspace.
  - No more than three actions should appear on the Toolbar. One action on the Toolbar can be used as a drop-down list of up to three additional actions.

Examples

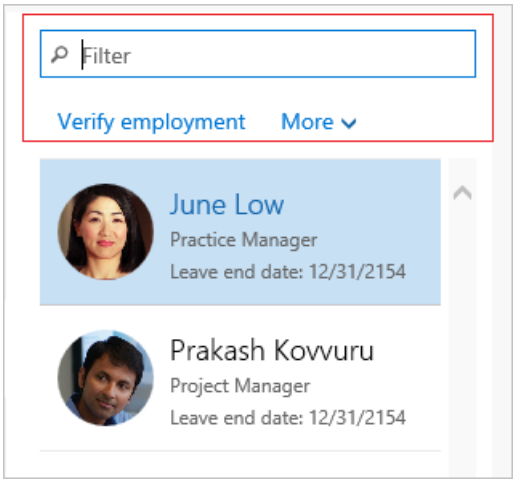
Filters and Toolbar - Inline

Form: **HcmWorkforceManagement** > **HcmOpenPositionsPart** (**All workspaces** > **Workforce management**)



Filters and Toolbar - Stacked

Form: **HcmWorkforceManagement** > **HcmWorkerOnLeaveListPart** (**All workspaces** > **Workforce management**)



Appendix

Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues

Why does the Inline variant allow for an arbitrary number of filter fields, but the Stacked variant allows a maximum of three (a QuickFilter and two custom filters)?

Two factors contribute to this discrepancy:

- A UX guideline specifies a maximum of two filters in these sections (and one of those filters could be a QuickFilter). Therefore, the Stacked variant more closely complies with the guideline.
- The number of fields in the Stacked variant is limited for aesthetic reasons. The filter fields in this variant are intended to take up the full width of the list/chart that appears below them, and their width is therefore **SizeToAvailable**. When this variant is used above narrow lists, as it's intended to be used, that setting can cause very narrow filter fields when more than two filter fields are used. The Inline variant is intended to be used above wider charts/lists. Therefore, the original pattern definition allowed for an arbitrary number of fields. Nevertheless, we do plan to address this discrepancy in the number of allowed filter fields between the two variations in the future.

Fill Text subpattern

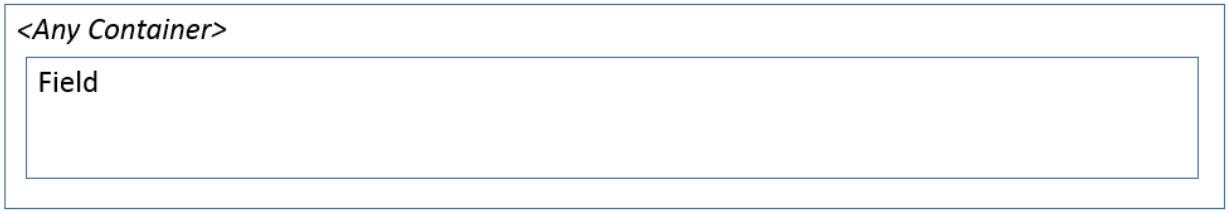
2/18/2021 • 2 minutes to read • Edit Online

This article provides information about the Fill Text subpattern. This subpattern is used when a single String or StaticText control must stretch to the full width of the container, so that users have more space to enter information.

Usage

Fill Text is used when you need a single String or StaticText control to stretch to the full width of the container. This subpattern is typically used for multi-line string controls that require more space for users to enter information.

Wireframe



Model

High-level structure

[Container]

String | StaticText

Core components

- Apply the Fill Text subpattern to the container control.

Related container patterns

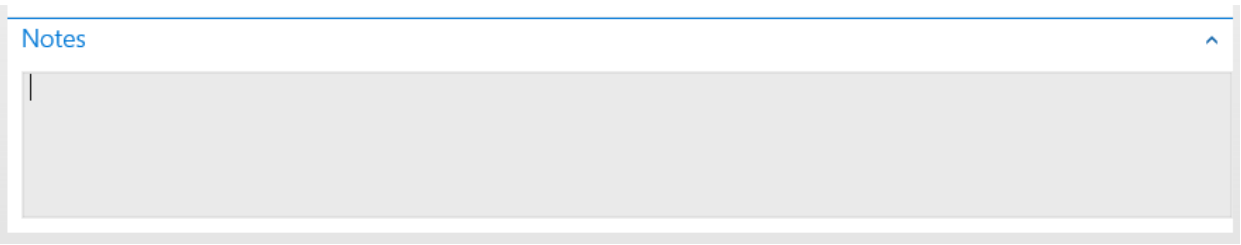
- Fields and Field Groups

UX guidelines

None

Examples

Form: **FmRental (Notes)**



Resources

Typically used by patterns

- Details Master
- Details Transaction
- Simple Details
- Simple List and Details
- Table Of Contents
- Wizard

Appendix

Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues

- The pattern currently sets the **HeightMode** property of the control to **SizeToAvailable**. This can produce very tall string controls if the pattern is used in a **SizeToAvailable** container. We're investigating whether this control should use **SizeToContent** height, or whether it should not set the property at all and should instead let the developer decide the appropriate control height.

Horizontal Fields and Buttons Group subpattern

2/18/2021 • 2 minutes to read • Edit Online

This article provides information about the Horizontal Fields and Buttons Group form subpattern. This subpattern is used when actions must be defined for an individual field on a form.

Usage

This subpattern is used when actions must be defined for an individual field on a form. The buttons are laid out just to the right of the field to visually associate the actions with the field. The buttons should display only an icon (no text). Actions that are associated with a section or an entire form should be placed in a Toolbar or ActionPane above that section or form.

Typical contents

- 1–2 fields
- 1–3 buttons

Wireframe

Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- The layout of fields and buttons will use a single column, where **ArrangeMethod=HorizontalLeft**.

Model

High-level structure

- Group (ArrangeMethod=HorizontalLeft)
  - Field
  - *Field (optional)*
  - Buttons (1–3 buttons)

Core components

- Apply the HorizontalFieldsButtonsGroup subpattern to the container control.
- Address BP Warnings:
  - There should be no more than three buttons.
  - No additional BP checks are required beyond the AX6.3 BP checks that were carried forward.

Related patterns

- Toolbar and Fields
- Horizontal Fields

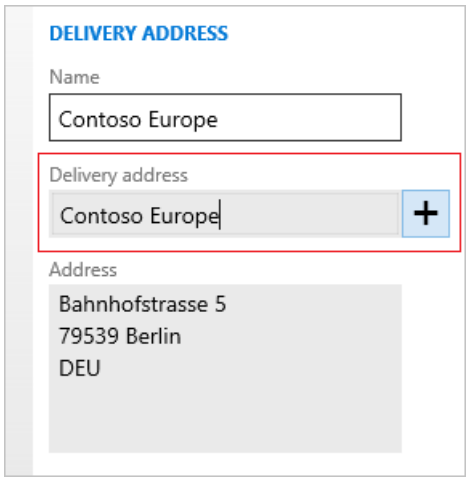
UX guidelines

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn't include any guidelines that will be enforced automatically through the development environment. Open the form in a browser, and walk through these steps.

- **Standard form guidelines:**
  - Standard form guidelines have been consolidated into the General Form Guidelines document.
- **Horizontal Fields and Buttons Group guidelines:**
  - The width of the fields + buttons should not exceed the standard size of a column.
  - Buttons should have a symbol image assigned.
  - Buttons should have tooltips.
  - There should be a maximum of three buttons. The last button can be a menu button.

Examples

Form: **SalesTable (GroupHeaderAddressHeaderOverview)**



Resources

Typically used by patterns

- Simple List and Details
- Table of Contents
- Details Master
- Details Transaction

Appendix

Frequently asked questions

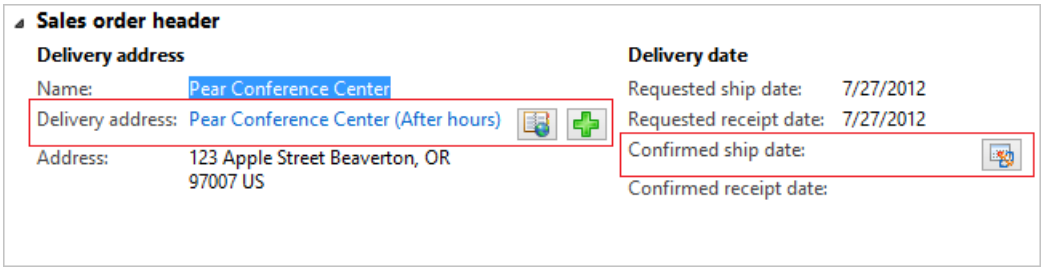
This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues

- None

Dynamics AX 2012 content

SalesTable



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Image Preview subpattern

2/18/2021 • 2 minutes to read • Edit Online

This article provides information about the Image Preview form subpattern. This subpattern can be used for most images that appear within a form container, especially within a FastTab or Group.

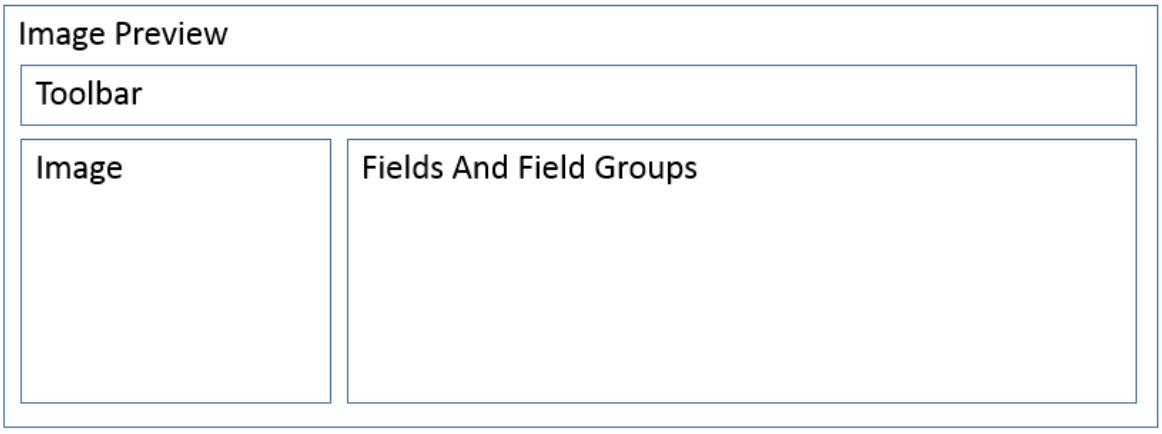
Usage

Image Preview can be used for most images that appear within a form container, especially within a FastTab or Group. This subpattern can be used in conjunction with the FieldsAndFieldGroup and FillText subpatterns to combine images and any associated fields. This subpattern isn't used for tiles or buttons, or for field status images.

Typical contents

- Toolbar (ActionPane where **Style**=**Strip**)
- Image
- Can contain subpatterns:
  - Fields and Field Groups
  - Fill text

Wireframe



Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- Fields are to the right of the image, if there are any fields.
- An ActionPane above the image can be used for associated actions (for example, Upload and Select).

Model

Image only – High-level structure

- [Container] (Columns = Fixed – 1)
  - *Toolbar (ActionPane) [Optional]*
  - Image

Image and fields – High-level structure

- [Container] (Columns = Fixed – 1)
  - *Toolbar (ActionPane) [Optional]*
  - Image
  - Group
    - Image
    - Group - Note: uses a fields subpattern

Core components

- Apply the Image Preview subpattern to the container control.
- Address BP Warnings:
  - No additional BP checks are required beyond the AX6.3 BP checks that were carried forward.

Related container patterns

- Fields and Field Groups
- Fill Text

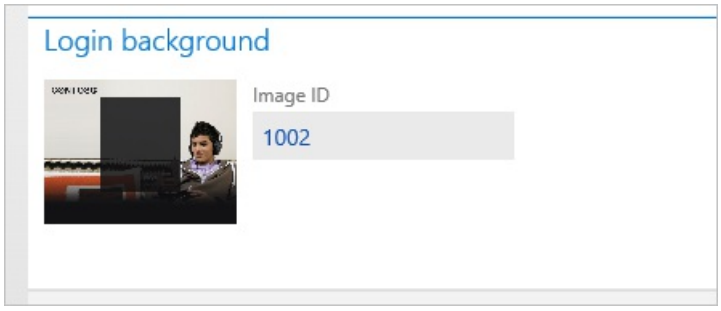
UX guidelines

The verification checklist shows the steps for manually verifying that the form complies with the UX guidelines. This checklist doesn't include any guidelines that will be enforced automatically through the development environment. Open the form in a browser, and walk through these steps.

- **Image Preview guidelines:**
  - Any fields should be placed to the right of the image.

Examples

Form: **RetailVisualProfile (Login)**



Resources

Typically used by patterns

- Details Master
- Details Transaction
- Simple Details
- Simple List and Details
- Table of Contents

Appendix

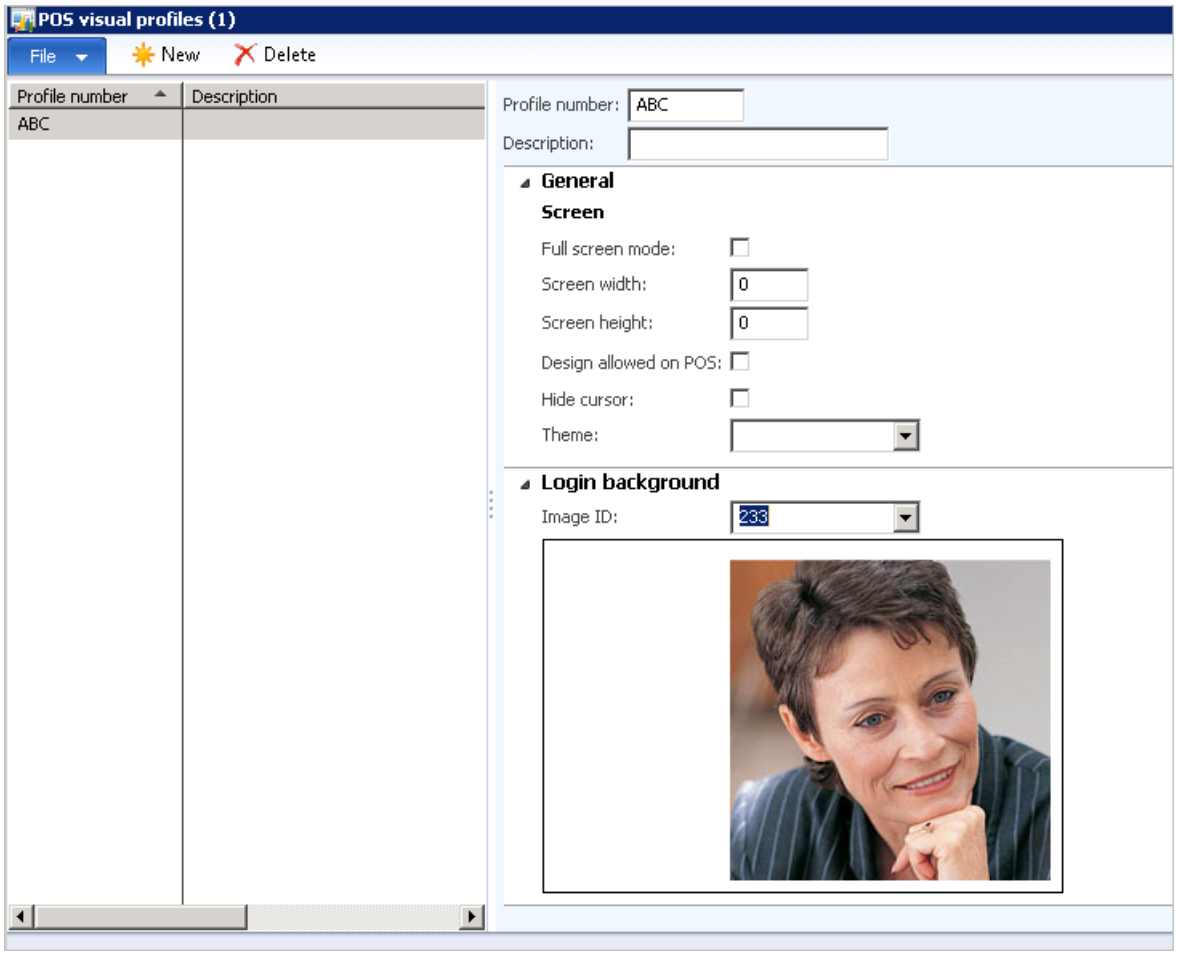
Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues

None.

AX 2012 content



> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

List Panel subpattern

2/18/2021 • 2 minutes to read • Edit Online

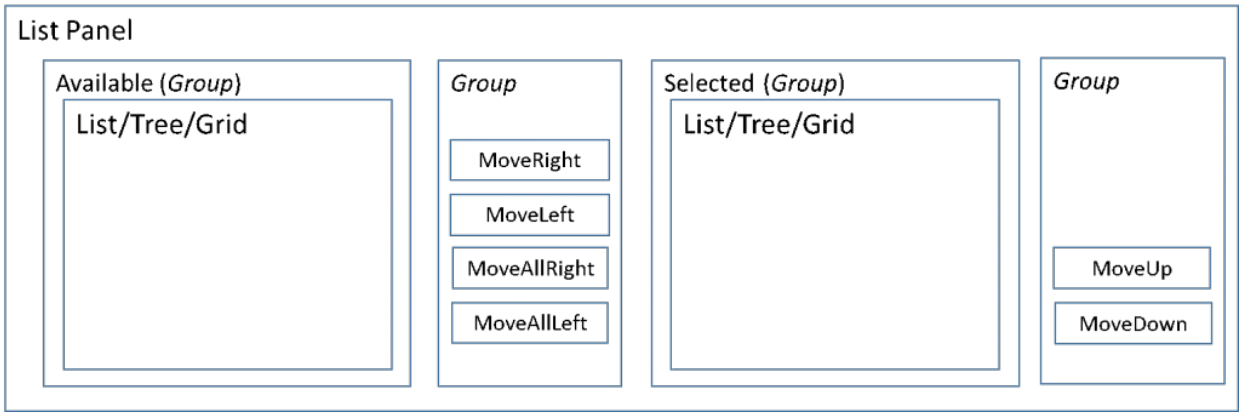
This article provides information about the List Panel form subpattern. Application teams use this subpattern to manage two lists that move data between each other.

Usage

List Panel is the subpattern that application teams use to manage two lists that move data between each other. This pattern is meant to represent a modeled version of the **SysListPanel** class (programmatic) approach of managing two lists that move data between each other. The List Panel subpattern can be applied on the following controls:

- TabPage control
- Group control

Wireframe



Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- List (Grid/ListView) and Tree controls are supported.
- The right panel is the selected section.
- The left panel is the available section.
- Six buttons are available as actions:
  - Add
  - Remove
  - Add All (Optional)
  - Remove All (Optional)
  - Move Up (Optional)
  - Move Down (Optional)

Model

High-level structure

- [Container]

- *CustomFilterGroup (Group) [Optional]*
- ListPanelGroup (Group)
  - AvailablePanel (Group)
    - Grid | Tree | ListView | ListBox
  - ActionPanel (Group)
    - AddButton (Button)
    - RemoveButton (Button)
    - *AllAllButton (Button) [Optional]*
    - *RemoveAllButton (Button) [Optional]*
  - SelectedPanel (Group)
    - Grid | Tree | ListView | ListBox*
  - *MoveUpDownPanel [Optional]*
    - MoveUpButton (Button)
    - MoveDownButton (Button)

Core components

- Apply the ListPanel subpattern to the container (TabPage or Group) control.
- Address BP Warnings:
  - No additional BP checks are required beyond the AX6.3 BP checks that were carried forward.

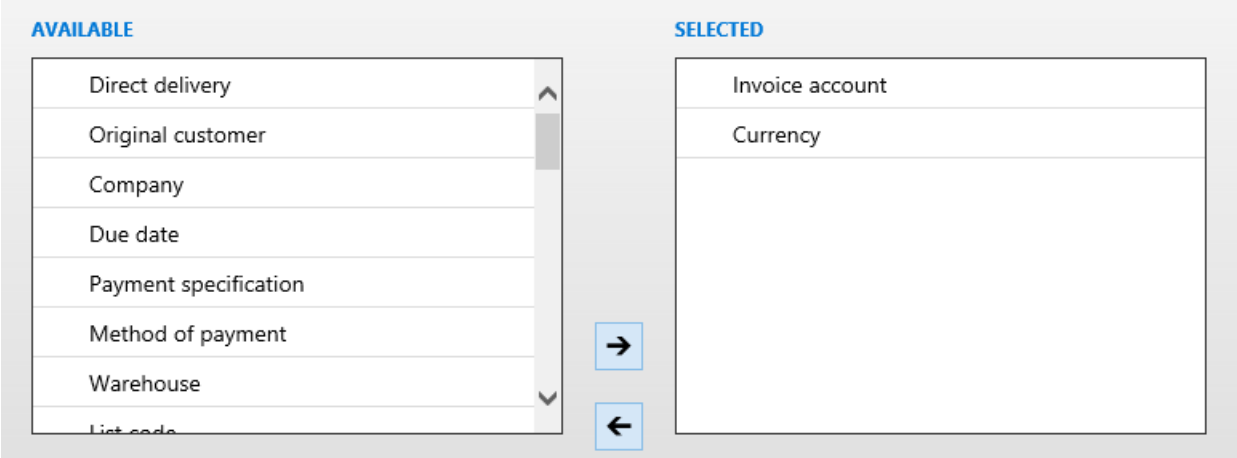
UX guidelines

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn't include any guidelines that will be enforced automatically through the development environment. Open the form in a browser, and walk through these steps.

- **Standard form guidelines:**
  - Standard form guidelines have been consolidated into the General Form Guidelines document.

Examples

Form: **SalesSummaryParameters (GroupQuotation)**



Resources

Typically used by patterns

- Simple List and Details
- Table of Contents
- Details Master
- Dialog
- Wizard

Appendix

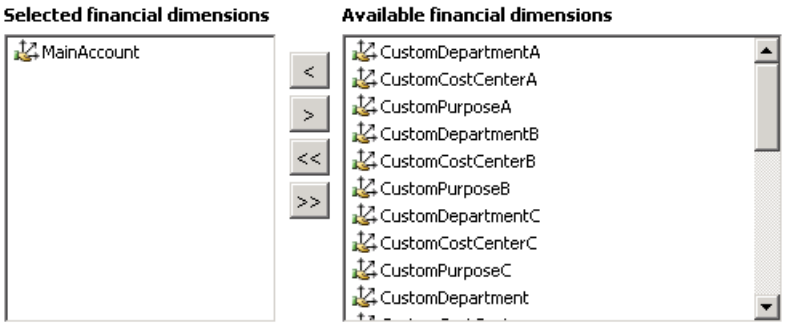
Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues

- None

AX 2012 content



Nested Simple List and Details subpattern

2/18/2021 • 3 minutes to read • Edit Online

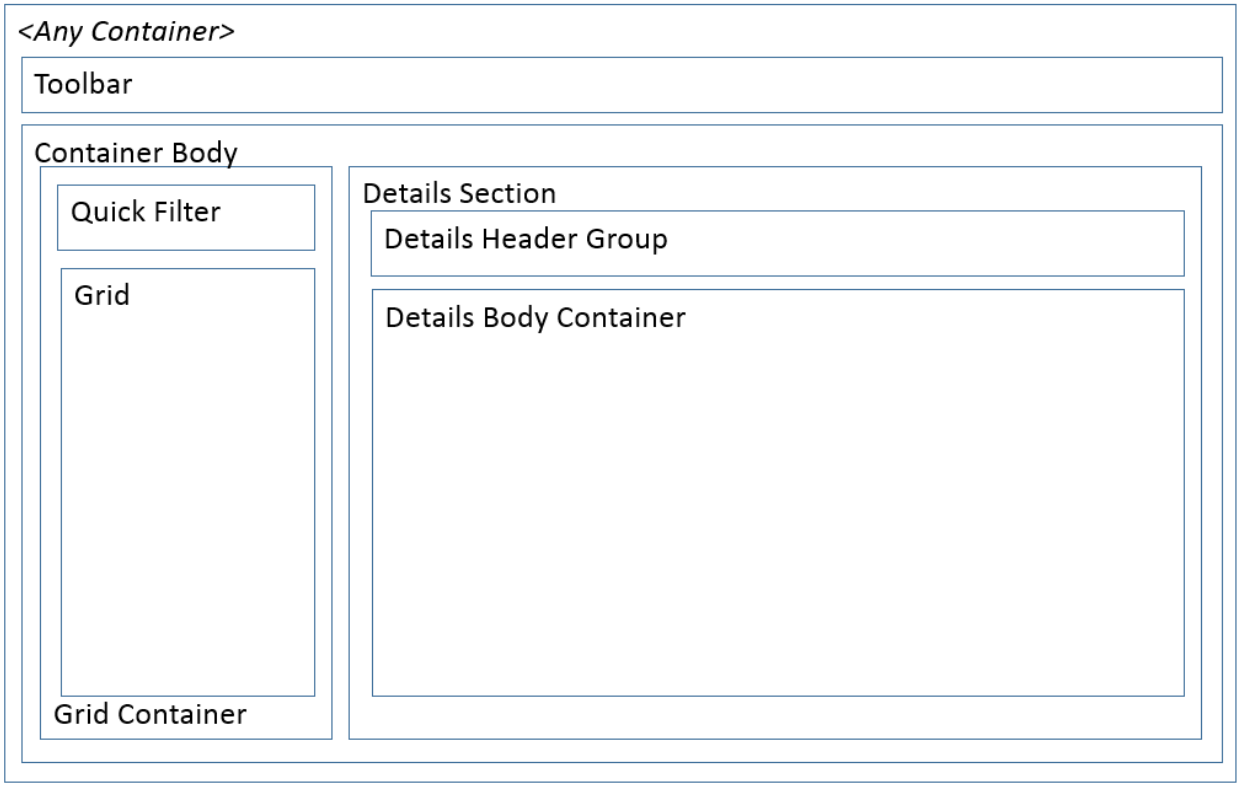
This topic provides information about the Nested Simple List and Details (NSL+D) subpattern. This subpattern is used to display information about a secondary or child entity when that child entity is presented within another form type.

Usage

This article describes a variant of the Simple List and Details (SL+D) pattern that is named the Nested Simple List and Details (NSL+D) subpattern. Whereas the SL&D form pattern is used to display information about the primary entity on the form, the NSL+D subpattern is used to display information about a secondary or child entity when that child entity is presented within another form type. The amount of information that is related to the child entity should be too much for a grid (10 or more fields) but not enough for the child entity to deserve its own form. The NSL+D subpattern has a few differences from the SL+D form pattern:

- You may not nest an NSL+D subpattern within another NSL+D subpattern.
- The NSL+D subpattern uses a Toolbar for contextual actions.
- The details portion of the NSL+D subpattern is simpler than the SL+D pattern. The NSL+D subpattern uses only groups, whereas the SL+D pattern organizes content into FastTabs.

Wireframe



Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- This pattern is new. Any pattern changes to the SL+D pattern can be found in the Simple List and Details pattern document.

Model

High-level structure

- <Container>
  - ActionPane (ActionPane Style=Strip)
  - ContainerBody (Group Columns=2)
    - ListContainer (Group)
      - Grid | Tree | ListView
    - DetailsContainer (Group)
      - DetailsHeader (Group)
      - *DetailsGroup (Group) [Optional]*

Core components

1. Apply the NestedSimpleListDetails subpattern to the container control.
2. Resolve the required BP checks:
   a. Set **Grid.Datasource=<secondary data source>**.
   b. Set grid data source **InsertIfEmpty=No**.
   c. Set **ActionPane.DataSource=<same data source as grid>**.
   d. Set Toolbar Command Add properties.
   e. Set Toolbar Command Remove properties.
   f. If the grid data source is read-only, make sure that there are no **Add/Remove** buttons on the Toolbar.

UX guidelines

The verification checklist shows the steps for manually verifying that the form complies with the UX guidelines. This checklist doesn't include any guidelines that will be enforced automatically through the development environment. Open the form in a browser, and walk through these steps.

Standard form guidelines

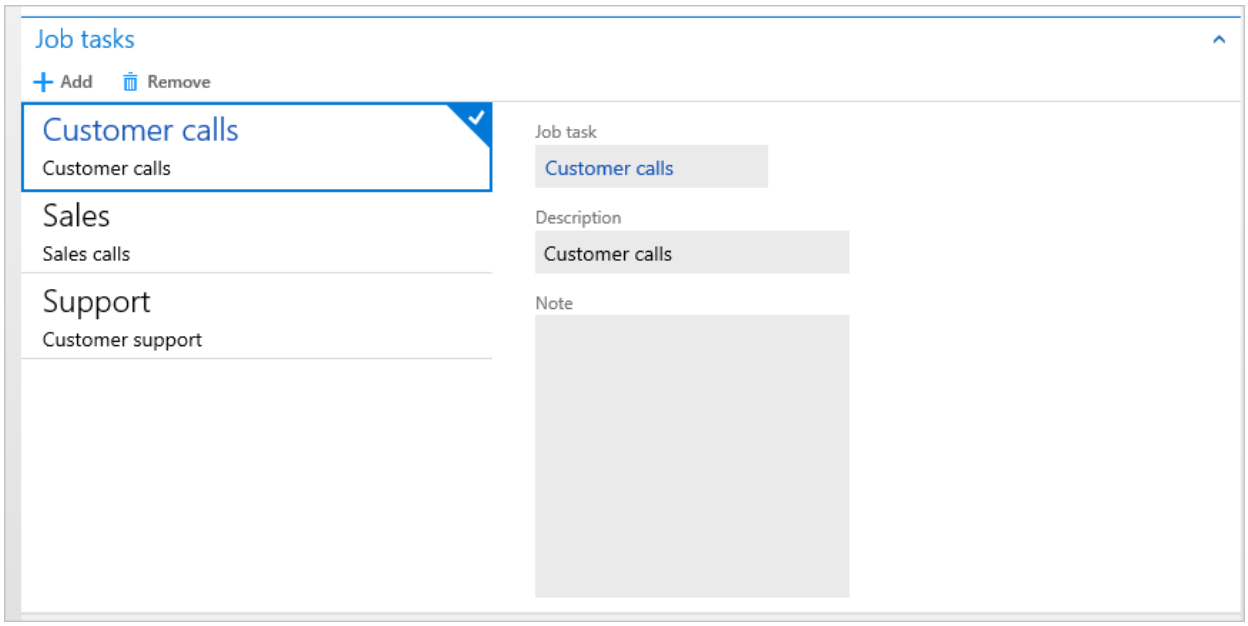
- Standard form guidelines have been consolidated into the General Form Guidelines document.

Nested simple list & detail guidelines

- There should not be duplicate **New** or **Delete** buttons.
- If a **grid** is used for the list portion of the pattern:
  - The grid should be a **list-style** grid.
  - List-style grids should display no more than three rows (lines) for each record in the List style grid. Typically, just the ID and Description are sufficient.
  - When there is no data, the grid control should not automatically add a new record.
- A **details** section that is displayed on the right of the Container Body:
  - Display the grid columns as the first fields in the Details Header Group, in the same order that they are displayed in the grid.
  - When a record is added, focus should go to the first field in the details section.

Examples

Form: **HcmJob** (**TaskTabPage**)



Resources

Typically used by patterns

- Simple List and Details
- Table of Contents
- Details Master
- Details Transaction

Appendix

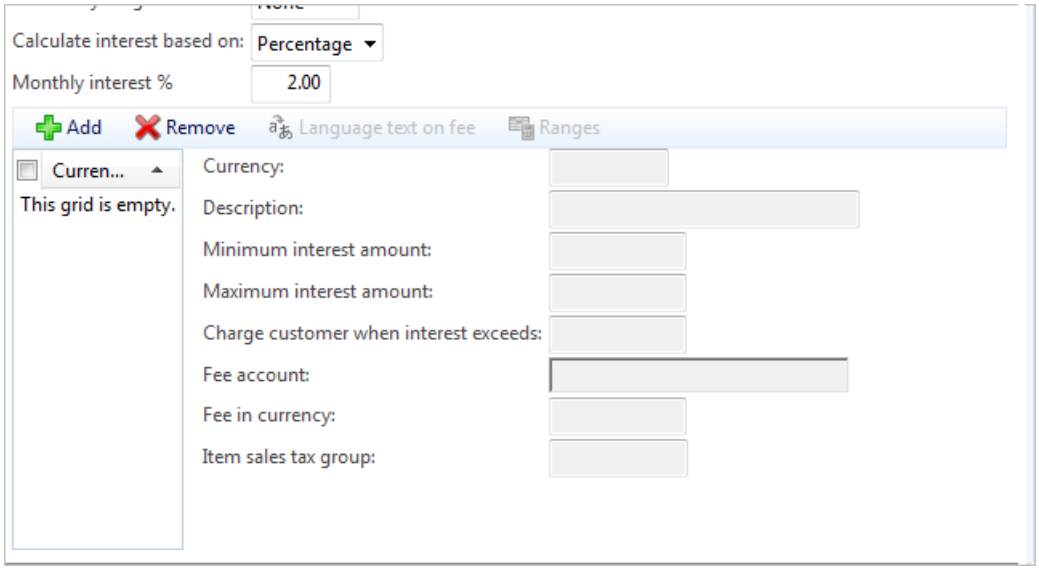
Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues

- **The details area of the nested pattern should not have FastTabs. The framework should verify/enforce this.**
  - Currently we aren't allowing tabs of any kind inside this pattern.

AX 2012 content



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Section Chart form pattern

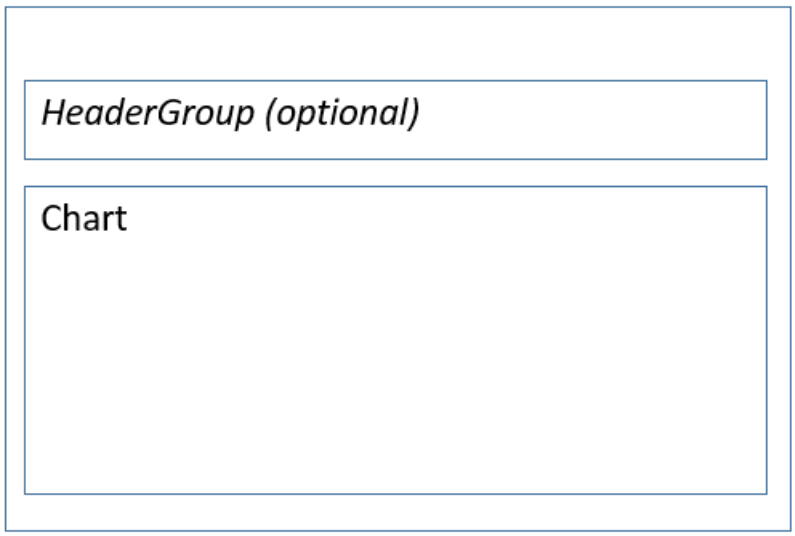
2/18/2021 • 2 minutes to read • Edit Online

This topic provides information about the Section Chart form pattern. This pattern is primarily used in conjunction with the Operational Workspace pattern, and specifically on forms that contain a chart control.

Usage

The Section Chart form pattern is intended to be used primarily in conjunction with the Operational Workspace pattern. Specifically, the chart section or summary section contains Form Part Controls that point to forms that contain charts. These referenced forms are intended to use the Section Chart pattern.

Wireframe



Pattern changes for Finance and Operations

This pattern didn't exist for Microsoft Dynamics AX 2012.

Model

High-level structure

- Form Design
  - *HeaderGroup (Group) [Optional]* – This uses one of the Filters and Toolbar subpatterns.
  - Chart

Core components

Apply the Section Chart pattern to the appropriate form/container.

Related container patterns

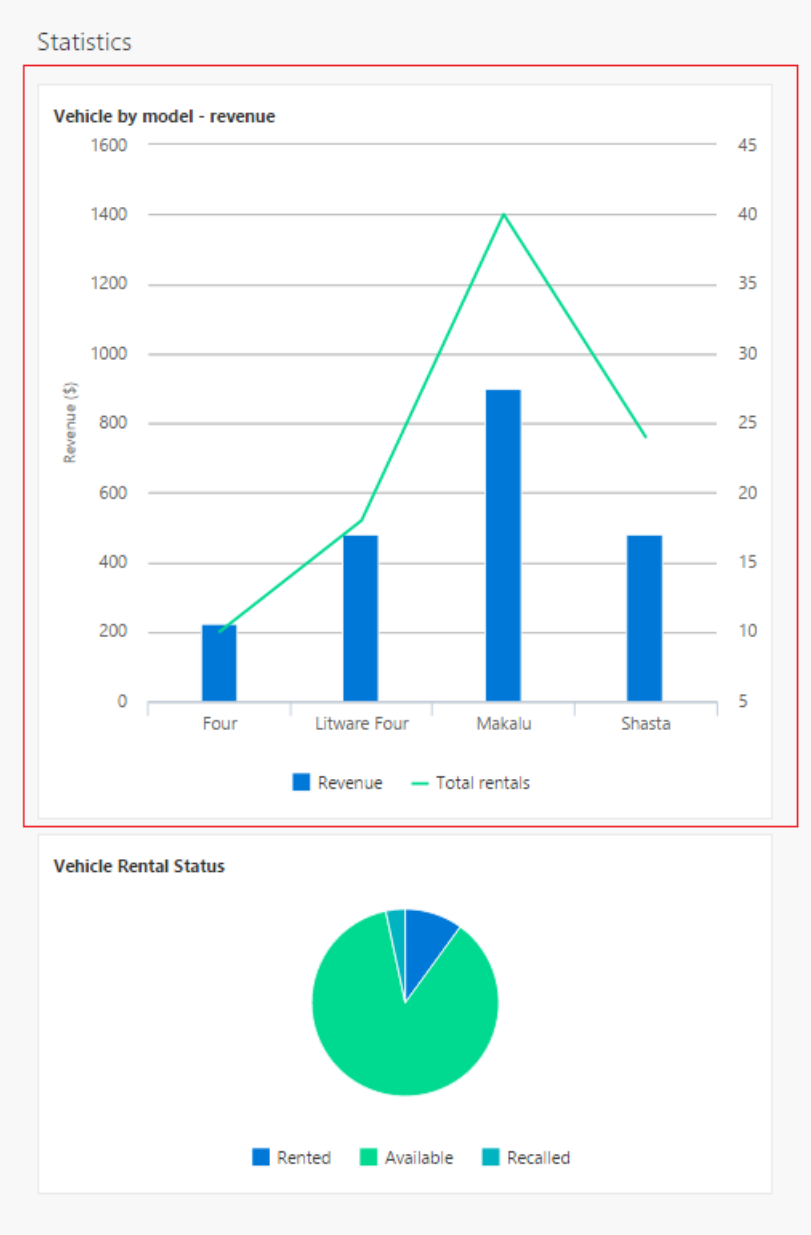
- Workspace
- Section stacked chart

UX guidelines

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn't include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps. None

Examples

Form: **FmBiChartPart_VehicleByModel** (**All workspaces** > **Reservation Management** (see the **Statistics** section)



Appendix

Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues

None

Section Power BI subpattern

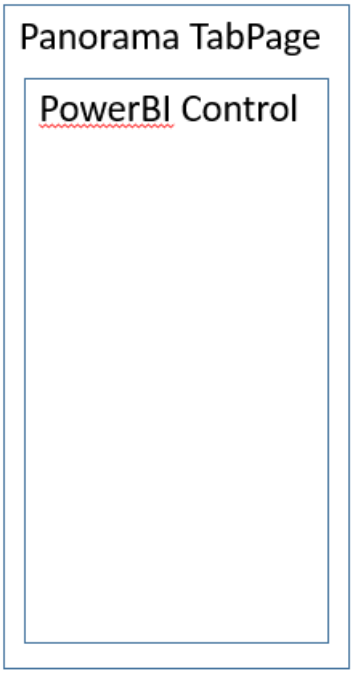
2/18/2021 • 2 minutes to read • Edit Online

This article provides information about the Section PowerBI subpattern. This subpattern is used as part of the Operational Workspace pattern, specifically for the panorama section that contains a PowerBI control.

Usage

The Section PowerBI subpattern is used as part of the Operational Workspace pattern, specifically for the panorama section that contains the PowerBI control.

Wireframe



Pattern changes for Microsoft Dynamics AX

This pattern didn't exist for Microsoft Dynamics AX 2012.

Model

High-level structure

TabPage PowerBI (PowerBI)

Core components

Apply Section PowerBI to the appropriate tab page in the workspace.

Related container patterns

- Operational workspace

UX guidelines

None

Examples

Form: **FmClerkWorkspace** (**All workspaces** > **Reservation Management**) PowerBI must be configured before the form can appear. (For information about how to configure PowerBI, see the Appendix.)

Appendix

Related articles

- Configure Power BI integration for workspaces
- Features and services available through Power BI integration

Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

- **How do I configure PowerBI for integration with my workspace?**
  - See the Configure Power BI integration for workspaces article.

Open issues

None

Section Related Links subpattern

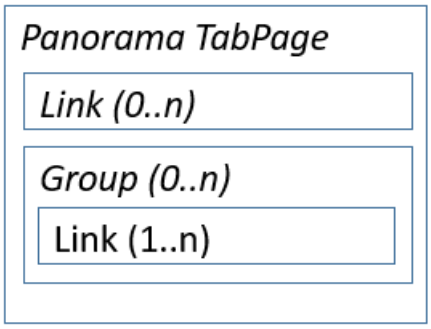
2/18/2021 • 2 minutes to read • Edit Online

This article provides information about the Section Related Links subpattern. This subpattern is used as part of the Operational Workspace pattern, specifically for the last panorama section that contains a set of links to other forms.

Usage

The Section Related Links subpattern is used as part of the Operational Workspace pattern, specifically for the last panorama section that contains a set of links to other forms.

Wireframe



Pattern changes for Microsoft Dynamics AX

This pattern didn't exist for Microsoft Dynamics AX 2012.

Model

High-level structure

- TabPage
  - *LinkButton ($Button) [0..N]*
  - *ButtonGroup (Group) [0..N]*
- LinkButton ($Button) [1..N]

Core components

Apply Section Related Links to the appropriate tab page in the Operational Workspace.

Related container patterns

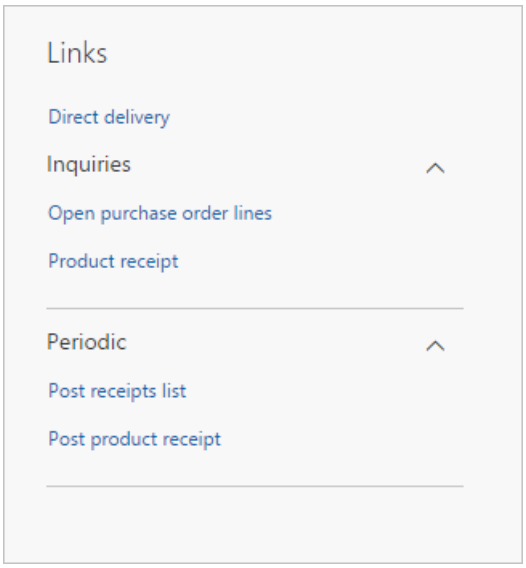
- Operational Workspace

UX guidelines

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn't include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps. None

Examples

Form: **PurchOrderProcessReceiptsWorkspace** (**All workspaces** > **Purchase order receipt and follow-up** (see the **Links** section)



Appendix

Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues

None

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Section Stacked Chart subpattern

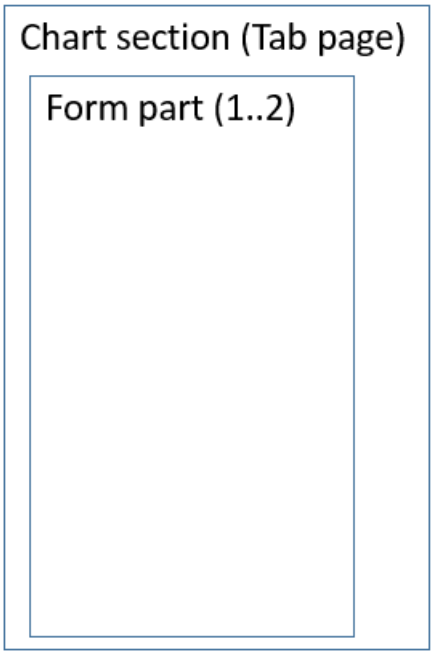
2/18/2021 • 2 minutes to read • Edit Online

This article provides information about the Section Stacked Chart subpattern. This subpattern is used as part of the Operational Workspace pattern when a panorama section contains one or two charts.

Usage

The Section Stacked Chart subpattern is used as part of the Operational Workspace pattern, specifically for a panorama section that contains one or two charts.

Wireframe



Pattern changes for Microsoft Dynamics AX

This pattern didn't exist for Microsoft Dynamics AX 2012.

Model

High-level structure

- TabPage
  - *ChartPart (FormPart) [0..N]*

Each Form Part points to a form that contains a single chart. Each of these forms should use the Section Chart form pattern.

Core components

Apply Section Stacked Chart to the appropriate tab page in the workspace.

Related container patterns

- Operational workspace

- Section Chart

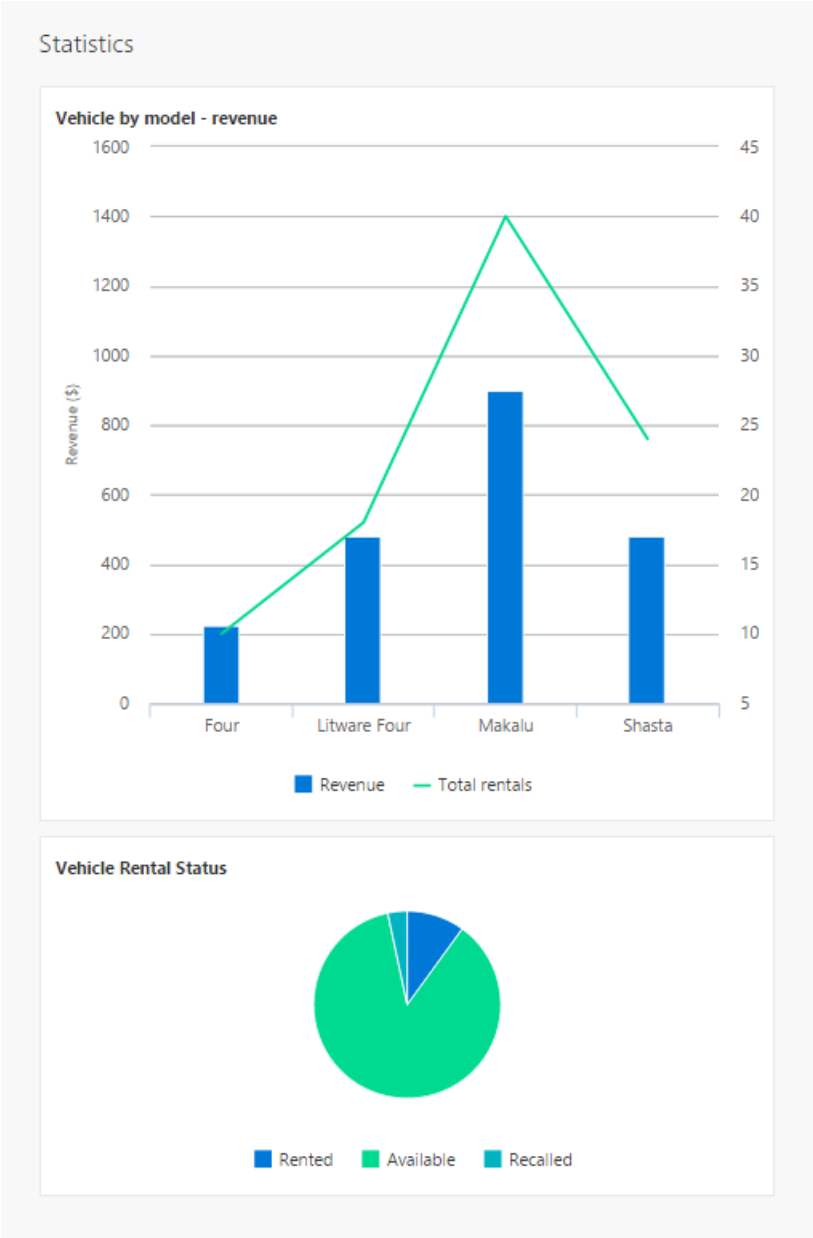
UX guidelines

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn't include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps.

- There should be no more than two charts in this section.
- Each Form Part Control should point to a form that uses the Section Chart pattern.

Examples

Form: **FmClerkWorkspace** (**All workspaces** > **Reservation Management**)



Appendix

Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues

None

Section Tabbed List subpattern

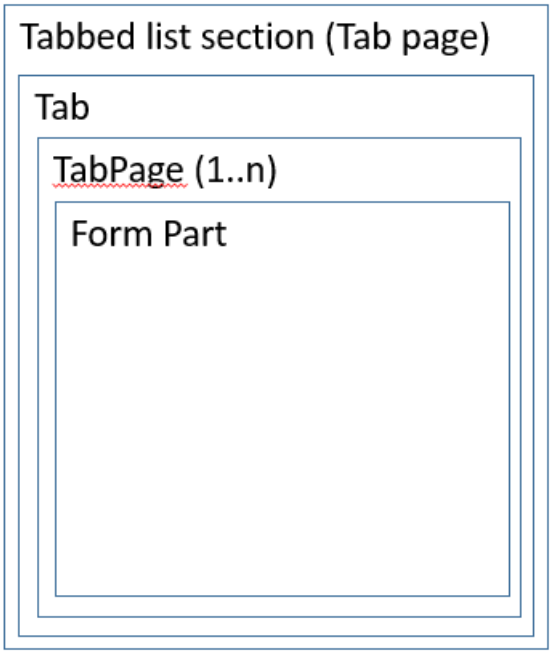
2/18/2021 • 2 minutes to read • Edit Online

This article provides information about the Section Tabbed List subpattern. This subpattern is used as part of the Operational Workspace pattern, specifically for a panorama section that contains a set of vertical tabs, each of which contains a filtered list of data.

Usage

The Section Tabbed List subpattern is used as part of the Operational Workspace pattern, specifically for a panorama section that contains a set of vertical tabs, each of which contains a filtered list of data.

Wireframe



Pattern changes for Microsoft Dynamics AX

This pattern didn't exist for Microsoft Dynamics AX 2012.

Model

High-level structure

- TabPage
- TabbedList (Tab) (Style=VerticalTabs)
    - *TabbedListPage (TabPage) [0..N]*
        - TargetForm (FormPart)

Each Form Part points to a form that contains the content for the section. Each of these forms should use one of the Form Part Section List form patterns.

Core components

Apply Section Tabbed List to the appropriate tab page in the workspace.

Related container patterns

- Operational Workspace
- Form Part Section List

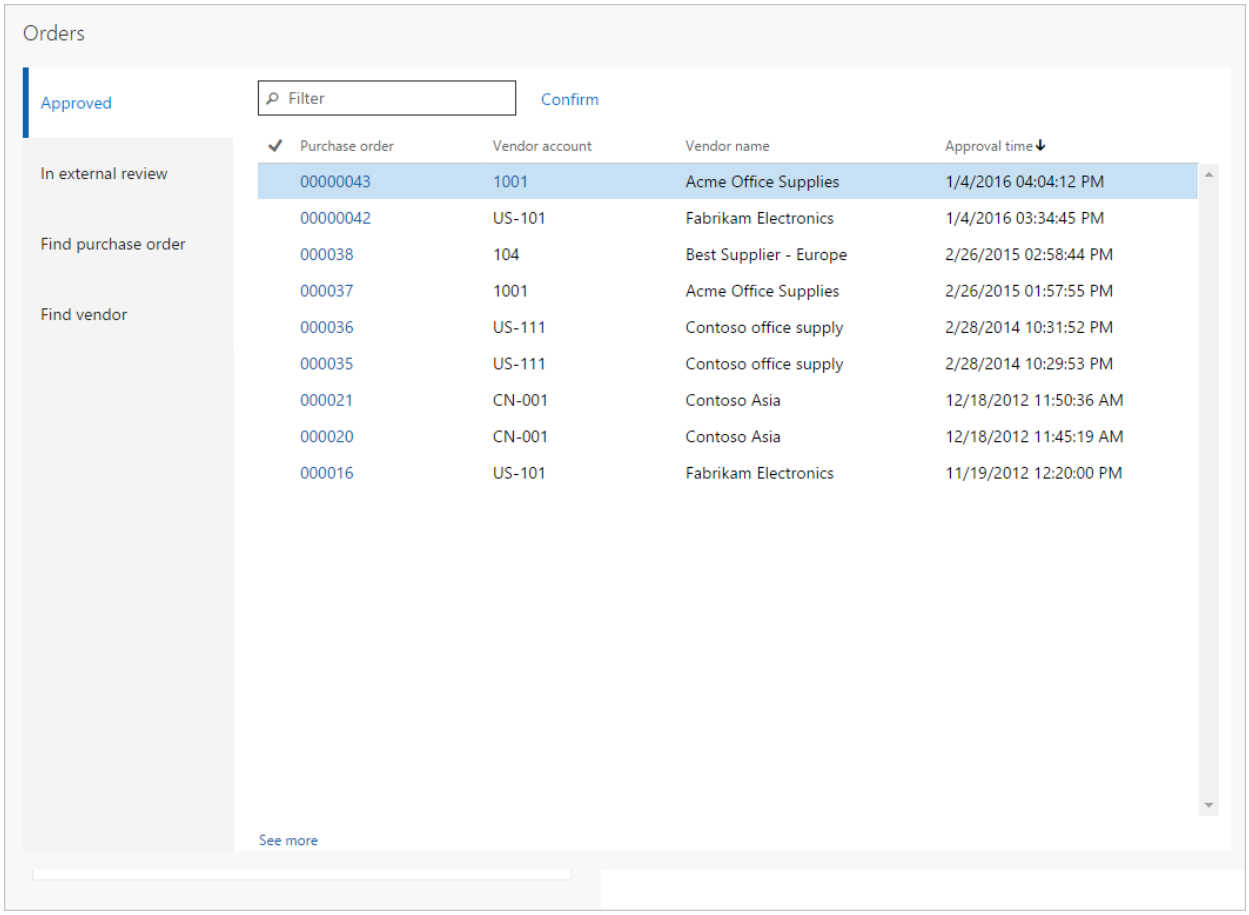
UX guidelines

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn't include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps.

- At least one list should be present in the tabbed list section.
- Each Form Part Control should point to a form that uses one of the Form Part Section List patterns.

Examples

Form: **PurchOrderMaintainWorkspace** (**All workspaces** > **Purchase order preparation**)



Appendix

Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues

None

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Section Tiles subpattern

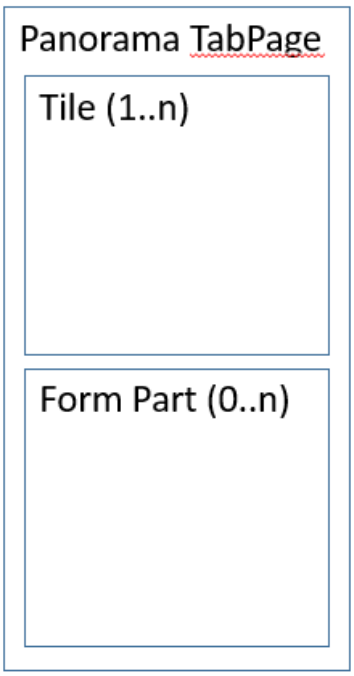
2/18/2021 • 2 minutes to read • Edit Online

This article provides information about the Section Tiles subpattern. This subpattern is used as part of the Operational Workspace pattern, specifically for the first panorama section (the Summary section) that contains a set of tiles, charts, and singleton cards.

Usage

The Section Tiles subpattern is used as part of the Operational Workspace pattern, specifically for the first panorama section (the **Summary** section) that contains a set of tiles, charts, and singleton cards.

Wireframe



Pattern changes for Microsoft Dynamics AX

This pattern didn't exist for Microsoft Dynamics AX 2012.

Model

High-level structure

- TabPage
  - *TileButton (TileButton) [0..N]*
  - *TargetForm (FormPart) [0..N]*

The Form Parts are used to embed Charts or singleton Cards into the **Summary** section of the workspace. Each form that represents a Chart should use the Section Chart form pattern.

Core components

Apply Section Tiles to the first tab page in the Operational Workspace.

Related container patterns

- Operational Workspace
- Section Chart

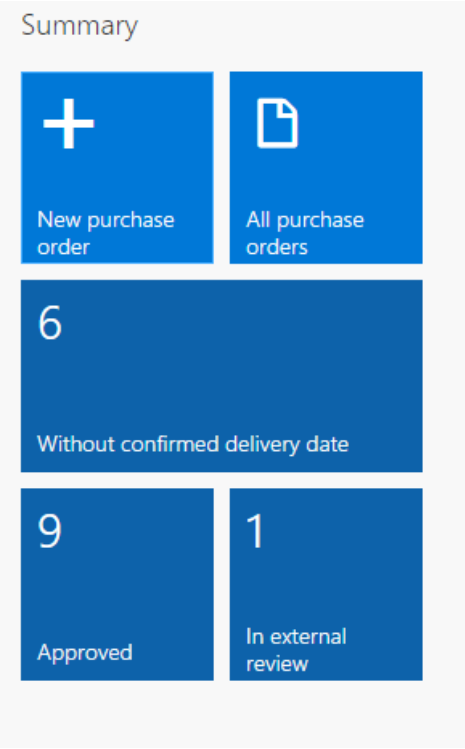
UX guidelines

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn't include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps.

- The **Summary** section should be named "Summary" or a variant that qualifies the word “Summary.”
- No two tiles in the workspace should have the same symbol.
- There should be a maximum of one "New" tile.
- Chart sizes should correspond to multiples of tile sizes.
  - Available sizes include 1 tile tall × 2 tiles wide, 2 × 2, 2 × 3, 2 × 4, 2 × 6, 4 × 4, 4 × 6, and 4 × 8.

Examples

Form: **PurchOrderMaintainWorkspace** (**All workspaces** > **Purchase order preparation** (see the **Summary** section)



Appendix

Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues

None

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Tabular Fields subpattern

2/18/2021 • 2 minutes to read • Edit Online

This article provides information about the Tabular Fields subpattern. This subpattern is used to show information efficiently in a tabular format.

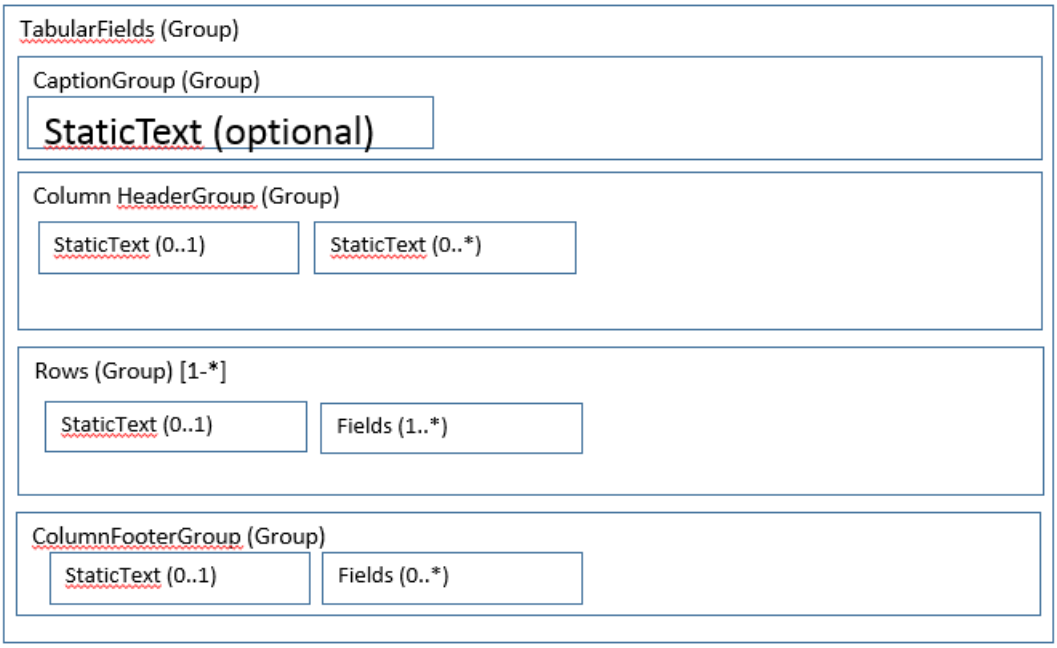
Usage

This subpattern is used to show information efficiently in a tabular format. The fields are arranged in a table that contains rows and columns, and that optionally contains column headers, row labels, a caption, and a footer. The Tabular Fields subpattern can be applied on the following controls:

- TabPage control
- Group control

Wireframes

Structural wireframe



Pattern changes

In previous releases of Microsoft Dynamics AX, there was no formally accepted way to model this pattern. Therefore, this pattern was modeled in many inconsistent ways that must be modified to match the current pattern. The most common way to model this pattern was to use groups for columns. However, groups are now used for the rows. The primary reason for this change was to better match the HTML/CSS constructs, and it also helps keep the tab sequence and semantics of a table.

Model

High-level structure

- TabularFields (Group*)
  - CaptionGroup (Group)
    - *TableCaption (StaticText) [Optional]*
  - TableHeaderRow (Group)
    - *Column0Label (StaticText) [Optional]* – **Note:** This static text fills col0, row0 with a blank.
    - ColumnLabels (StaticText) [1..N] – **Note:** These are the normal column headers.
  - TableRows (Group) [1..N]
    - *RowLabel (StaticText) [Optional]*
    - RowValues ($Field) [1..N] OR SecondaryColumnLabel (StaticText) [1..N]
  - TableFooterGroup (Group)
    - *Column0Label (StaticText) [Optional]* – **Note:** This static text fills col0, footer with a blank.
    - *RowValues ($Field) [0..N]* – **Note:** All the footer fields are in view mode.

Note that the four groups in the top-level tabular fields are mandatory structural elements. However, the contents of all those groups exception the Rows (Group) are optional. Additionally, note that Tabular Fields can also be used on a TabPage control. The structure is the same as the structure that is shown here.

Core components

Apply the Tabular Fields pattern on the top-level group or tab page. Address the pattern errors and problems.

UX guidelines

No manual verification is required.

Examples

Form: **LedgerJournalTransVendPaym (Balances)** (**Accounts payable** > **Journals** > **Payment journal** > **Lines**)

	DEBIT	CREDIT	DIFFERENCE
VOUCHER	0.00	0.00	0.00
JOURNAL	0.00	0.00	0.00

Resources

Typically used by patterns

- Simple List and Details
- Table of Contents
- Details Master

- Details Transaction

Appendix

Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

- **Why are we changing how the tabular field layout is created?**
  - To accomplish the layout in HTML, we must align with the way that HTML layout works. HTML layout groups by rows, not by columns.

Open issues

- None

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Toolbar and Fields subpattern

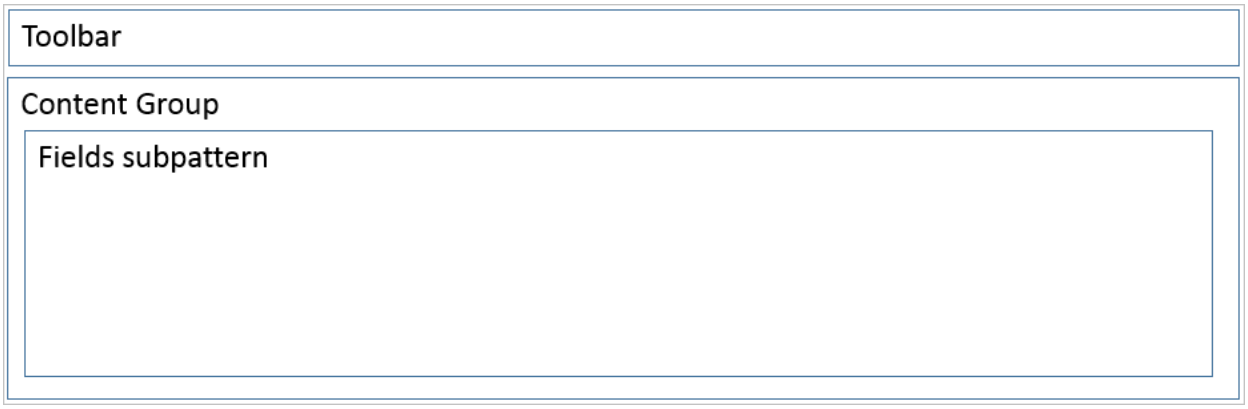
2/18/2021 • 2 minutes to read • Edit Online

This article provides information about the Toolbar and Fields subpattern. This container pattern is used to show actions above a subpattern of data fields. The toolbar should contain fewer than 10 actions.

Usage

This container pattern is used to show actions above a subpattern of data fields. The toolbar should contain fewer than 10 actions.

Wireframe



Model

High-level structure

- [Container]
  - Toolbar (ActionPane, Style=Strip)
  - ContentGroup (Group) – **Note:** A fields subpattern is used.

Core components

- Apply the ToolbarFields subpattern to the container control.
- Address BP Warnings:
  - No additional BP checks are required beyond the AX6.3 BP checks that were carried forward.

Related patterns

- Toolbar and List

Commonly used subpatterns

- Fields and Field Groups
- Tabular Fields
- Dimension Expression Builder

UX guidelines

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn't include any guidelines that will be enforced automatically through the development

environment. Open the form in the browser, and walk through these steps.

Standard form guidelines:

- Standard form guidelines have been consolidated into the General Form Guidelines document.

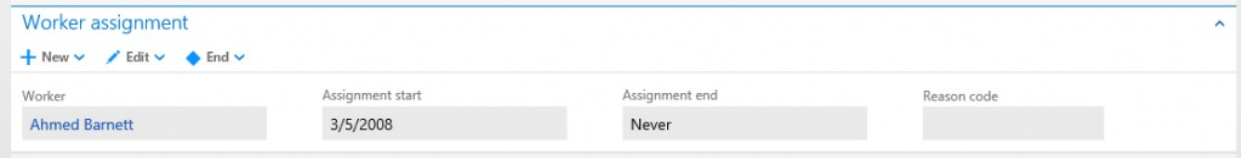
Toolbar guidelines:

- Toolbar guidelines have been consolidated into the Dynamics AX General Form Guidelines document.

Examples

Toolbar and Fields

Form: **HcmPosition (WorkerAssignmentTabPage)**



Resources

Typically used by patterns

- Simple List and Details
- Table of Contents
- Details Master
- Details Transaction

Appendix

Frequently asked questions

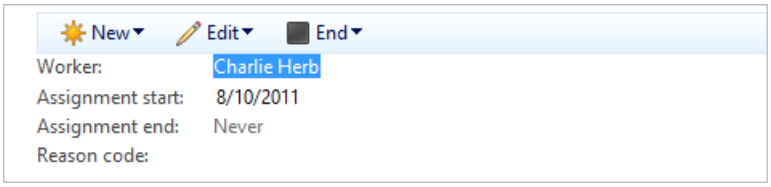
This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues

- **Should the ShowMoreLess group be part of the pattern, or should it be its own subpattern?**
  - We will treat the **ShowMoreLess** group as a custom container pattern until there is enough demand to justify the addition of a new pattern.

Microsoft Dynamics AX 2012 content

HcmPosition



Toolbar and List subpattern

2/18/2021 • 2 minutes to read • Edit Online

This article provides information about the Toolbar and List form subpattern. This subpattern is used to show child collections for the parent entity as either a tabular grid or a tree.

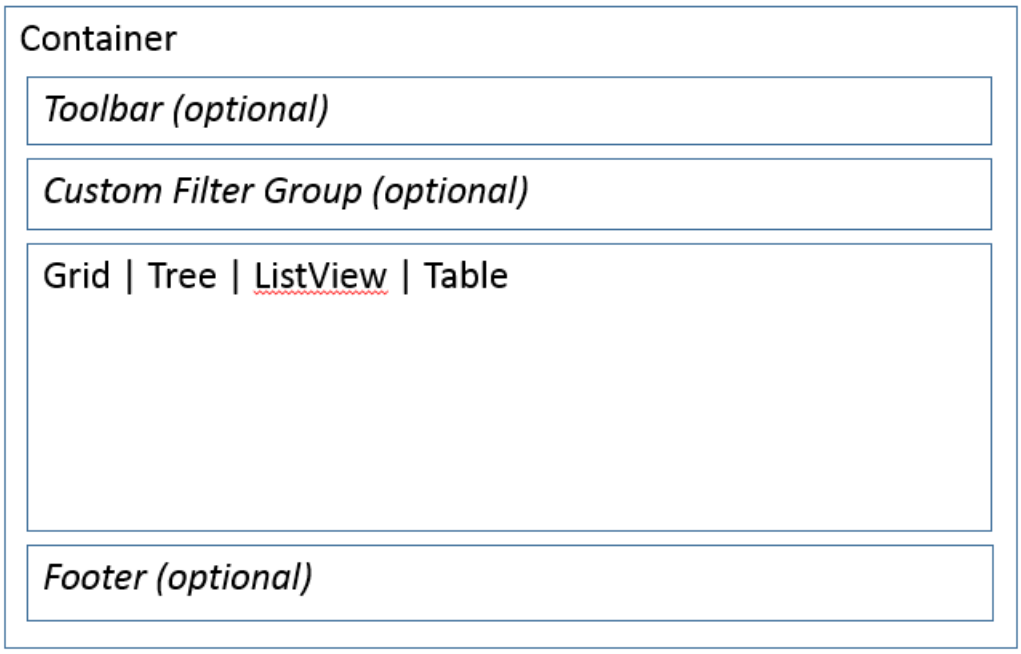
Usage

This subpattern is used to show child collections for the parent entity as either a tabular grid or a tree. The toolbar contains fewer than 10 actions. If a grid is used, it contains fewer than 10 fields. This article describes two patterns:

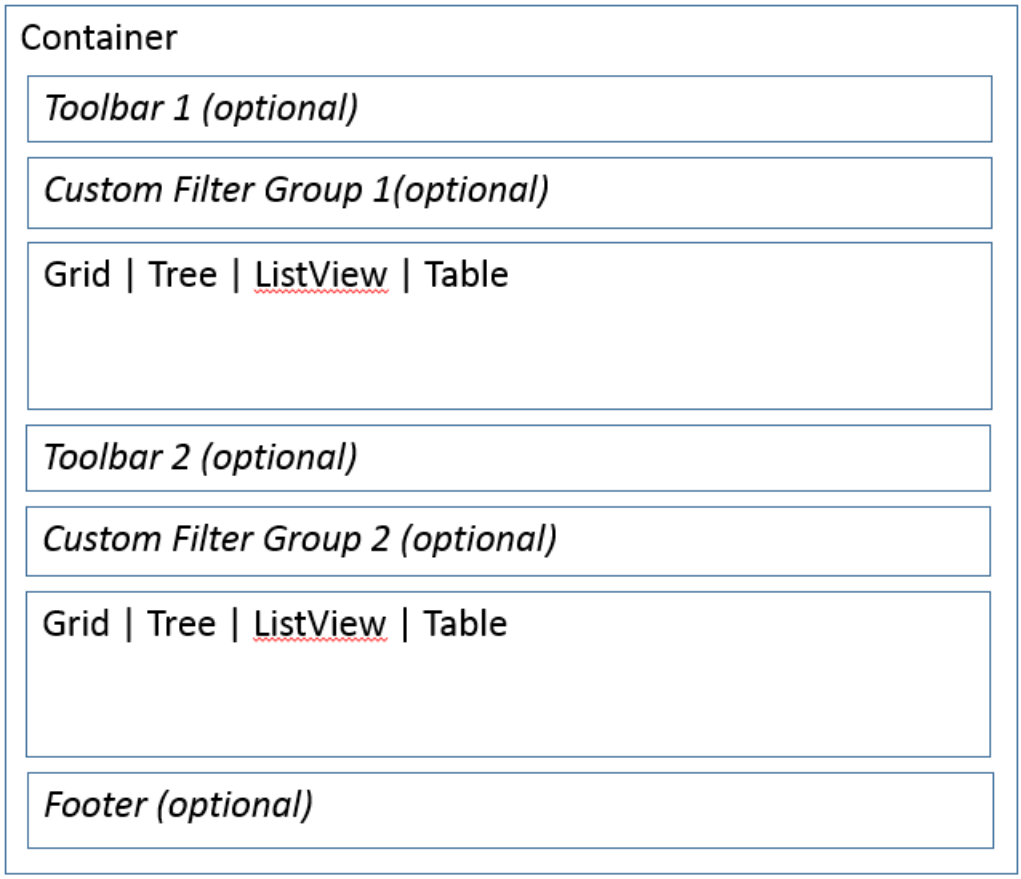
- **Toolbar and list** – This is the basic version of the pattern and should be used by default.
- **Toolbar and list (double)** – This variant includes two lists and an optional toolbar above each list.

Wireframes

Toolbar and list



Toolbar and list (double)



Model

Toolbar and list – High-level structure

- [Container]
  - *Toolbar (ActionPane, Style=Strip) [Optional]*
  - *CustomFilterGroup (Group) [Optional]*
  - Grid | Tree | ListView | Table
  - *Footer (Group) [Optional]*

Toolbar and list (double) – High-level structure

- [Container]
  - *Toolbar1 (ActionPane, Style=Strip) [Optional]*
  - *CustomFilterGroup1 (Group) [Optional]*
  - Grid | Tree | ListView | Table
  - *Toolbar2 (ActionPane, Style=Strip) [Optional]*
  - *CustomFilterGroup2 (Group) [Optional]*
  - Grid | Tree | ListView | Table
  - *Footer (Group) [Optional]*

Core components

- Apply the ToolbarList subpattern to a container control.
- Address BP Warnings:
  - **Grid.DataSource** must not be empty.
  - No additional BP checks are required beyond the AX6.3 BP checks that were carried forward.

Related patterns

- Toolbar and Fields

UX guidelines

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn't include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps.

Standard form guidelines:

- Standard form guidelines have been consolidated into the Microsoft Dynamics AX General Form Guidelines document.

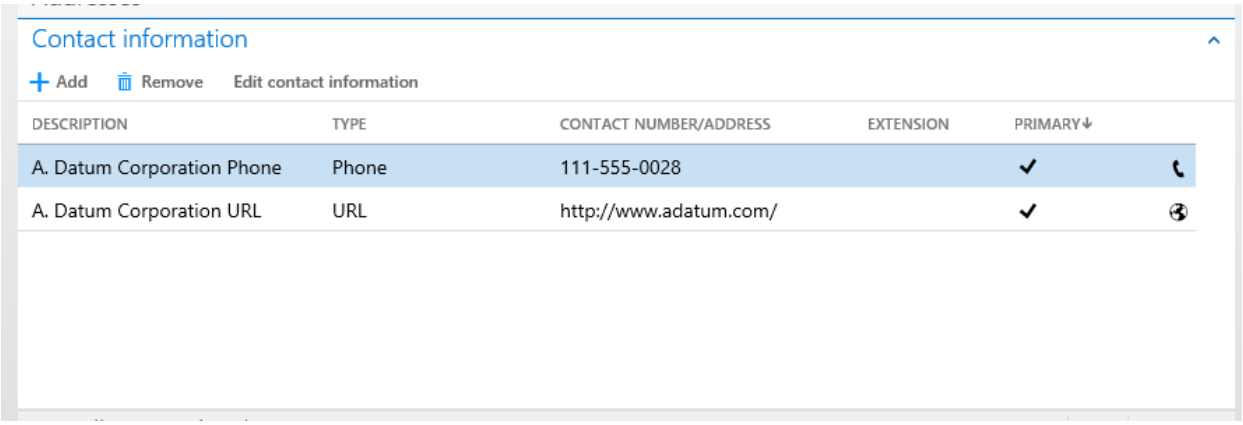
Toolbar guidelines:

- Toolbar guidelines have been consolidated into the Dynamics AX General Form Guidelines document.

Examples

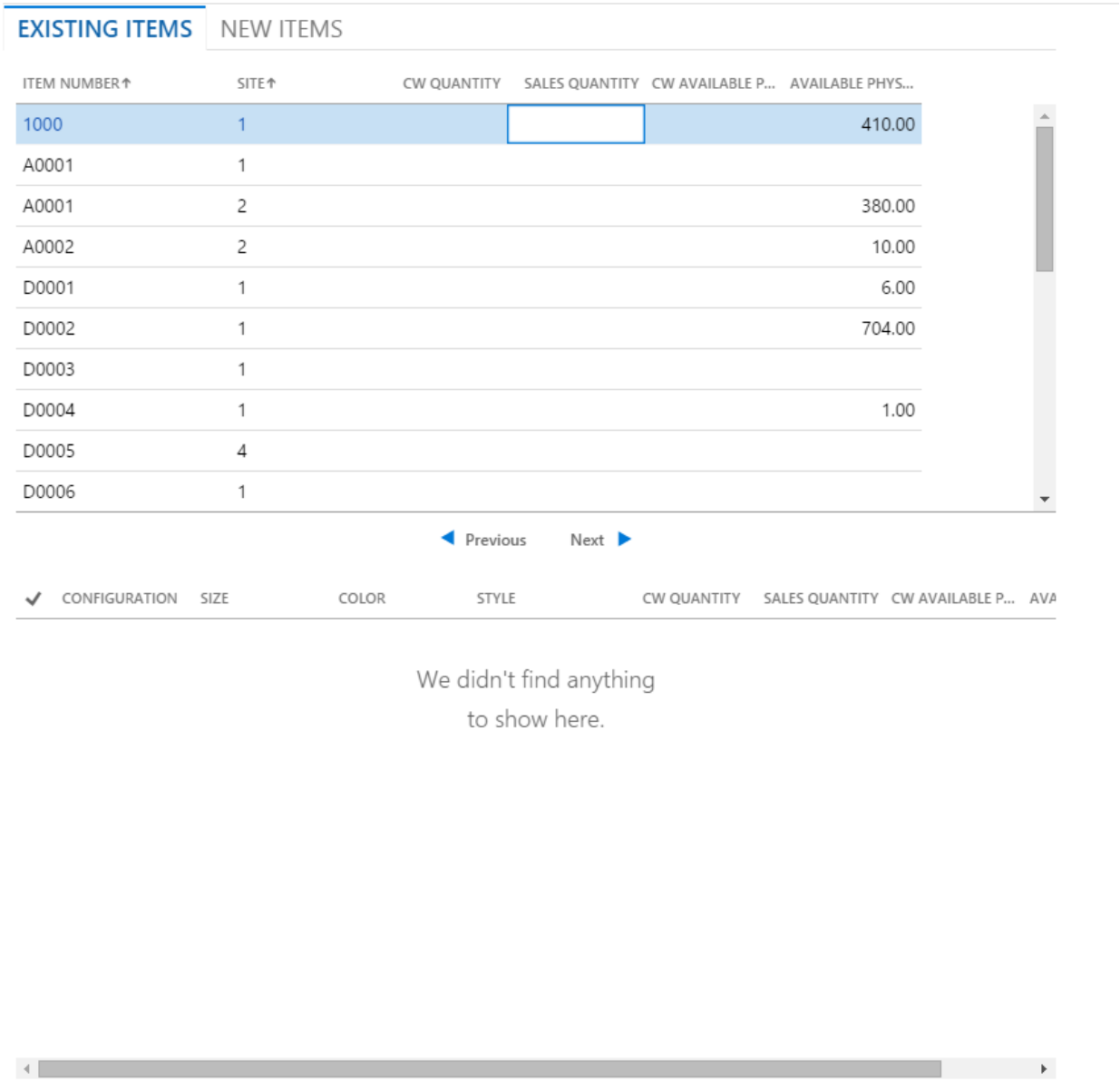
Toolbar and list

Form: **VendTable (TabCommunication)**



Toolbar and list (double)

Form: **SalesQuickQuote (TabPageExistingItems)**



Resources

Typically used by patterns

- Simple List and Details
- Table of Contents
- Details Master
- Details Transaction

Appendix

Frequently asked questions

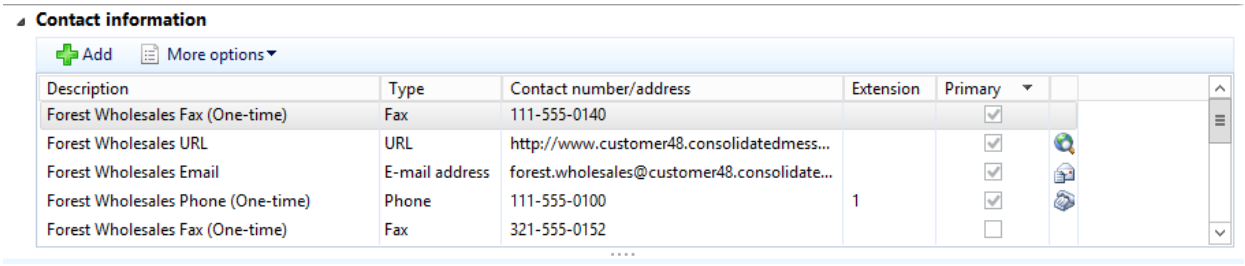
This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues

- **Does CommandButton.Command contain Add and Remove actions that provide the default icon, label, and tooltip?**
  - Not yet. Deliverable 1052359 will look at setting these defaults for Add and Remove commands.
- **Do we care about separators between groups of icons in a migration context?**
  - Currently, we don't show separators between button groups on toolbars.

Microsoft Dynamics AX 2012 content

VendTable



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Workspace Page Filter Group subpattern

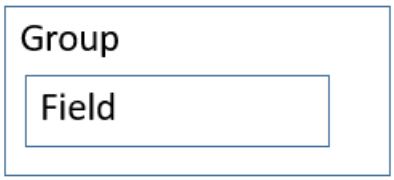
2/18/2021 • 2 minutes to read • Edit Online

This article provides information about the Workspace Page Filter Group subpattern. This subpattern is used as part of the Operational Workspace pattern when a workspace must expose a single workspace-wide filter on the form.

Usage

The Workspace Page Filter Group subpattern is used as part of the Operational Workspace pattern, specifically when a workspace must expose a single workspace-wide filter on the form.

Wireframe



Pattern changes for Microsoft Dynamics AX

This pattern didn't exist in Microsoft Dynamics AX 2012.

Model

High-level structure

- Group
  - Field ($Field)

Core components

Apply Workspace Page Filter Group to the appropriate group in an (operational) workspace.

Related container patterns

- Custom Filter Group
- Operational workspace

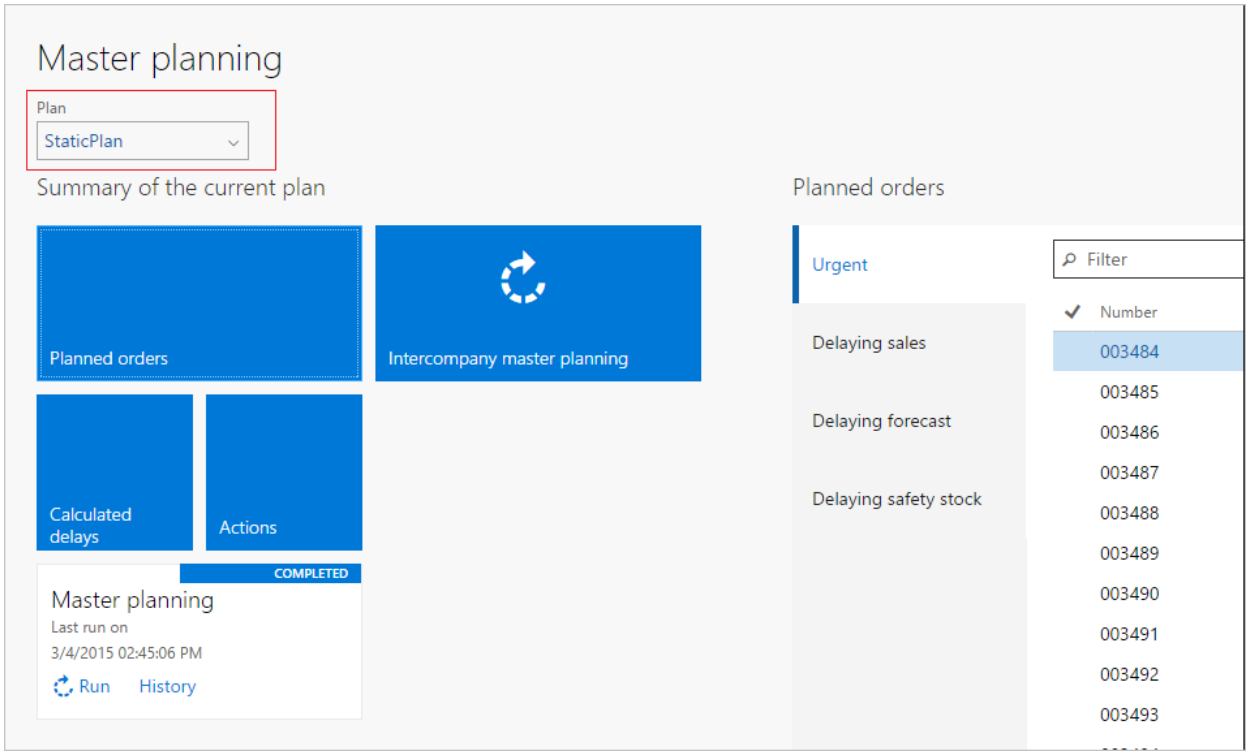
UX guidelines

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn't include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps.

- The filter field should have a drop-down that contains the list of available values.
- The expectation is that the filter field will be modified multiple times per day by the target user. If the filter field is modified less often or is primarily static, it should be put in a configuration dialog.
- If more filter fields are needed, they should be put in a configuration dialog (up to five filter fields).

Examples

Form: **ReqCreatePlanWorkspace** (**All workspaces** > **Master Planning**)



Appendix

Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues

None

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Build extensible controls

2/18/2021 • 22 minutes to read • Edit Online

This topic describes how to create new application controls that have a property sheet in Visual Studio and have server-side business logic.

Prerequisites

For this tutorial, you must access the environment by using Remote Desktop, and you must be provisioned as an administrator on the instance. For more information, see Deploy and access development environments.

Overview

The Control Extensibility Framework lets you create new application controls. You can use the same tools that Microsoft uses to build controls that are already present in the program, such as the chart control. Three important artifacts are involved in the process of developing an extensible control:

- **The X++ build class** – The build class lets a developer define the properties that appear in the Microsoft Visual Studio property sheet for the control. The developer can also define the modeling behavior for the control when it's used in the form designer. The build class is consumed by the run-time class to initialize the state of the control based on the value of properties in the property sheet.
- **The X++ run-time class** – The run-time class lets a developer define server-side business logic and data access patterns for an extensible control. Two concepts that are specific to building extensible controls are the *properties* and *commands* that the X++ class defines. Each property and command that is defined is serialized into a JavaScript view model at run time, and can be consumed by the client parts of the extensible control (the HTML and JavaScript). These properties and commands are the main channels for moving information between the server-side and client-side parts of the control.
- **The control HTML and JavaScript** – Each control uses HTML, JavaScript, and CSS files to define control visualization and client-side interaction patterns. By using the Microsoft Dynamics HTML binding syntax together with jQuery, a developer can consume the properties and commands that are defined in X++ to design powerful data-driven UI.

All three artifacts of extensible control development are explained in more detail in the following sections.

Key concepts

- Defining an extensible control's design-time behavior
- Defining an extensible control's run-time behavior
- Defining an extensible control's view by using HTML and CSS
- Defining an extensible control's view model by using JavaScript

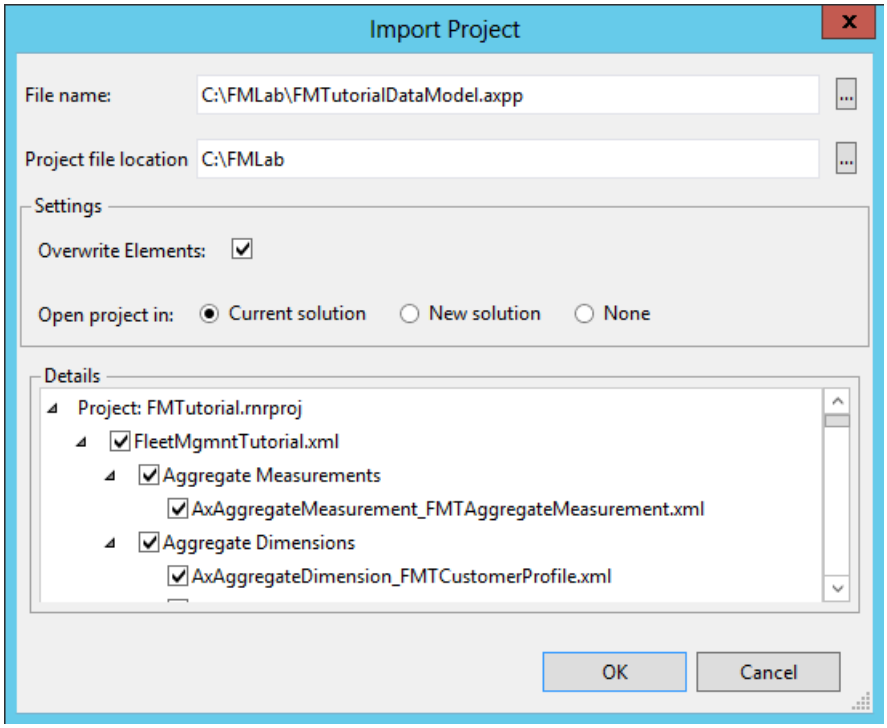
Setup

Import the tutorial project and transactional data

Use Visual Studio to import the tutorial project. The tutorial project includes the artifacts that you will use to complete this tutorial. Use Visual Studio to open the FMTutorial project and load the data for the tutorial. You will use the **FMTDataHelper** class to load data for the Fleet Management tutorial.

1. Download the Fleet Management sample from https://github.com/Microsoft/FMLab, save it to **C:**, and unzip it.

2. On the desktop, double-click the Visual Studio shortcut to open the development environment.
3. On the **Dynamics 365 **menu, click **Import Project**.
4. In the **Import Project** dialog box, next to the **File name** text box, click the ellipsis button (...).
5. In the **Select the file to import** dialog box, browse to **C:\FMLab**, click **FMTutorialDataModel.axpp**, and then click **Open**.
6. In the **Project file location** field, enter **C:\FMLab**.
7. Select the **Overwrite Elements** check box and the **Current solution** option. The following screen shot shows the completed **Import Project** dialog box.



8. Click **OK**.
9. In Solution Explorer, under the **FMTutorial** project, expand **Classes**.
10. Right-click **FMTDataHelper**, and then click **Set as Startup Object**.
11. On the **BUILD** menu, click **Rebuild Solution**. You use the rebuild to make sure that all the files in the project are built, regardless of timestamps. You can view the build progress in the **Output** window.
12. After the build is completed, press Ctrl+F5 to run the project. The **Login** form closes when authentication succeeds, and then the data is loaded.

Set up aggregate data

Use FMTAggregateMeasurements to populate the Microsoft SQL Server Analysis Services database with aggregate data.

> **NOTE**
> These steps must be completed immediately after you use the **FMTDataHelper** class to import data. You may **NOT** need to do these steps if the aggregate measure is "InMemoryRealTime", depending on what tutorial files you have.

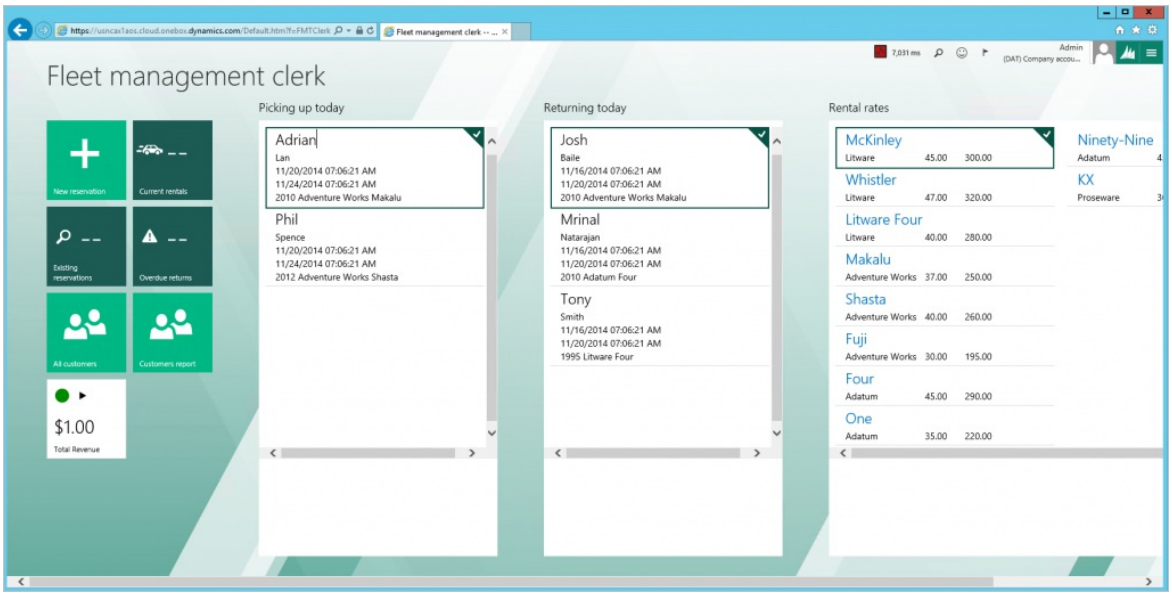
1. In Solution Explorer, under **Analytics**, double-click **FMTAggregateMeasurement**.
2. In the designer, right-click **FMTAggregateMeasurement**, and then click **Deploy and Process**.

Preview the clerk workspace

Before you begin to build the contact control, look at the appearance of the current implementation. In the following sections, you will use the Control Extensibility Framework to enrich the visualization of the controls and the form.

1. In Solution Explorer, expand **Forms**, right-click **FMTClerkWorkspace**, and then click **Set as Startup Object**.

2. Press Ctrl+F5 to open the **Fleet management clerk** page in Internet Explorer. As the following screen shot shows, the data on this page appears as a simple grid in a list style that contains several string and date controls.



3. Exit Internet Explorer.

Modify the build class for the contact control

To save time, you will work on a partially completed extensible control that is named the contact control. You will extend the contact control to complete its design, run-time, and visualization behaviors. The partially completed contact control already supports multiple title fields, subfields, and action buttons. However, it doesn't currently support an image. To add image support, you must extend the design experience for the contact control. You will add a data field that can specify image data.

Technical overview

To see an example of a build class, in Solution Explorer, expand **Classes**, right-click **FMTBuildContactControl**, and then click **View Code**. The class code appears in the code editor. **FMTBuildContactControl** is the build class for the contact control. For each extensible control, the build class defines the properties that the control shows in the property sheet. The build class also defines the modeling experience for the control in the Visual Studio form designer. There are three primary design-time behaviors that you can define for an extensible control. Each behavior is declaratively defined by using a **FormDesign** attribute. Here are the design-time behaviors that you can define:

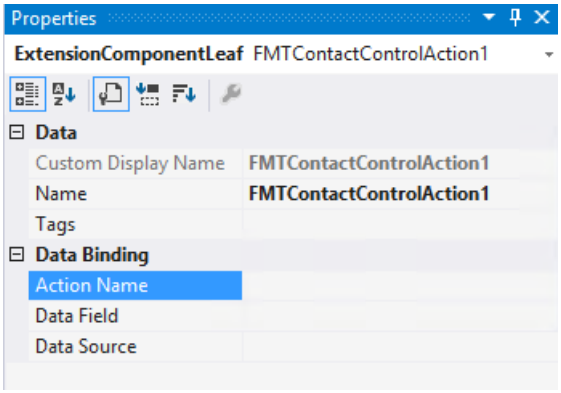
- **Name** – You can specify the control name that appears in the form designer when you add the control to a form. To specify the name, add a **FormDesignControlAttribute** attribute to the build class declaration of the extensible control. For example, the following declaration of the **FMTBuildContactControl** class shows the attribute.

```
[FormDesignControlAttribute("FMT Contact Control")]
Class FMTBuildContactControl extends FormBuildControl
```

- **Designer properties** – These are the properties that you see in the property sheet when you add the control to a form. There are several attributes that let you add various types of designer properties. For example, the **FormDesignPropertyAttribute** attribute adds a property to the property sheet, and the property name and the section are supplied as arguments to the attribute. For example, the following code adds the **Action Name** property to the **FMTContactControlAction** class.

```
[FormDesignPropertyAttribute("Action Name", "Data Binding")]
public str parmActionName (str _actionName = actionName){...}
```

The following screen shot shows how this property appears in the **Properties** pane in Visual Studio.

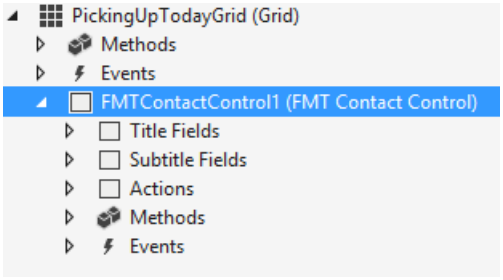


- **Child design components** – These are child nodes that you see after you add the control to a form. There are two types of child design components: leaf and leaf collection.
  - A leaf is defined by using a **FormDesignComponentAttribute** attribute on an X++ method that accepts or returns another build class. The build class determines the properties that the leaf has in the property sheet.
  - A leaf collection is defined by using a **FormDesignComponentCollectionAttribute** attribute. The allowable leaf types for the collection are defined by using **FormDesignComponentValidChildAttribute** attributes.

For example, the following code adds a leaf collection that is named **Actions** for the **FMTBuildContactControl** class.

```
[FormDesignComponentCollectionAttribute("Actions"),
FormDesignComponentValidChildAttribute(classstr(FMTContactControlAction), "Action")]
public List parmActions(List _actions = actions){...}
```

The following screen shot shows how the specified child design component appears when you add the control to a form.



Tutorial steps

1. Check that the code for the **FMTBuildContactControl** class appears in the code editor. If it doesn't, in Solution Explorer, expand **Classes**, right-click **FMTBuildContactControl**, and then click **View Code**.

2. Add a child design component to the FMTBuildContactControl class. A child design component lets a developer who places the control in a form to specify the image that appears on the control. In this step, you will add the **FormDesignComponentAttribute** attribute to create a new entry in the property sheet. You will then add the **FormDesignPropertyDataFieldAttribute** attribute, which indicates that the new designer property enables the selection of a data field.

   a. Add the highlighted code that follows to the declarations for the class. This code adds the **FormBindingDataField** field to the X++ that the **FMTBuildContactControl** class is using.

```
[FormDesignControlAttribute("FMT Contact Control")]
class FMTBuildContactControl extends FormBuildControl
{
    str dataSource;
    FormBindingDataField imageField;
    List titleFields;
    List subFields;
    List actions;
```

   b. Add the following code to the **FMTBuildContactControl** class. Add this method after the designer property for the data source.

```
[FormDesignComponentAttribute ("Image")]
public FormBindingDataField parmImageField(FormBindingDataField _imageField =
imageField)
{
    if(!prmisdefault(_imageField))
    {
        imageField = _imageField;
    }
    return imageField;
}
```

> **NOTE**
>
> The child design component will show the properties that are available on the **FormBindingDataField** build class. This is appropriate, because you want to enable image data binding to a data field and data source. This is all that is required to add a designer property to the build class of the contact control.

3. Press Ctrl+S to save your changes, and then close the code editor.

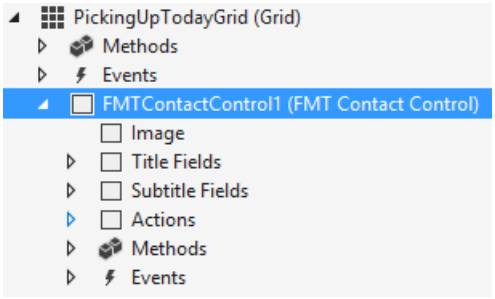
4. In Solution Explorer, right-click **FMTutorial**, and then click **Build**.

5. If the **FMTPickingUpTodayPart** form isn't already open, expand **Forms**, and then double-click

FMTPickingUpTodayPart. The form opens in the form designer.

6. In the form designer, expand **Design** > **PickingUpTodayGrid**, and then select and delete the controls that currently appear in the grid.

7. Right-click **PickingUpTodayGrid**, point to **New**, and then click **FMTContactControl**. Expand the **FMTContactControl** node, and notice that **Image** appears as a new child design component. The following illustration shows the contact control in the form designer.



You must also update the run-time class for the contact control to consume the design-time changes. You will revisit adding the control to the form and specifying data bindings and property values later.

8. Press Ctrl+S to save your changes, and then close the form designer.

Modify the runtime class for the contact control

Next, you must modify the run-time class to read the data source and data field for the image from the build class. You must also create a run-time property, so that the image data is available to the control's client HTML and JavaScript.

Technical overview

To see an example of the run-time class, in Solution Explorer, expand **Classes**, right-click **FMTContactControl**, and then click **View Code**. The class opens in the code editor. **FMTContactControl** is the run-time class for the contact control. The class defines the run-time behavior of the contact control. The run-time class typically contains X++ for data access or business logic. In addition, there are two primary run-time behaviors that are related to extensible controls that you define in the run-time class. Each behavior is declaratively defined by using an attribute.

- **Run-time properties of the control** – These properties can be of two types:
  - *Static properties*, which are set via code or initialized with values from designer properties.
  - *Bindable properties*, for which the run-time value is determined by a binding to a data source and data field combination.

  Run-time properties are declared by using **FormPropertyAttribute** attributes. The following example shows a property declaration in **FMTContactControl**.

```
[FormPropertyAttribute(FormPropertyKind::Value, "TitleFields")]
public List parmTitleFields(List _value = titleFieldsProperty.parmValue()){...}
```

  The **FormPropertyAttribute** attribute accepts two arguments:

  - The first argument indicates to the framework the kind of JavaScript view model property to create.
    - If you supply **BindableValue**, a **ReferenceProperty** is generated in the JavaScript view model. A **ReferenceProperty** updates itself when data changes in the data source.
    - If you supply **Value**, a **ValueProperty** is generated in the JavaScript view model. A developer

must write code to update the value of a **ValueProperty**.

- The second argument of the attribute sets the name for the property as it will be defined in the JavaScript view model.

> **NOTE**
>
> Don't be concerned if **TitleFields** don't seem to be bound to data because the example uses a **Value** property. The **TitleFields** property returns a List that contains **FormBindingDataFields**, each of which is data-bound. The X++ method that has the **FormPropertyAttribute** attribute is a simple getter/setter that uses a **FormProperty** as the backing field. The **FormProperty** contains the logic for updating the property, based on value or data source changes. It also serves as the backing field for the property.

- **Run-time commands for the control** – Commands enable the client parts of the control to trigger X++ logic, based on client-side user interactions. Commands are declared by using a **FormCommandAttribute** attribute. The single argument specifies the name of the command as it will appear in the JavaScript view model. The following example shows a command declaration in **FMTContactControl**.

```
[FormCommandAttribute("ExecuteAction")]
public void executeAction(str _actionName){...}
```

Tutorial steps

1. Verify that the **FMTContactControl** class is open in the code editor. If it isn't, in Solution Explorer, expand **Classes**, right-click **FMTContactControl**, and then click **View Code**.

2. Add a run-time property for the image data to **FMTContactControl**. In the **FMTContactControl** class, declare a **FormProperty** that is named **imageFieldProperty**, as shown by the highlighted line in the following example.

```
[FormControlAttribute(...)]
class FMTContactControl extends FormTemplateControl
{
    FormProperty dataSourceProperty;
    FormProperty imageFieldProperty;
    FormProperty titleFieldsProperty;
    FormProperty subtitleFieldsProperty;
    FormProperty actionsProperty;
```

3. Add the following X++ method after the **parmDataSource** X++ method. The new method will serve as the getter/setter for **imageFieldProperty**.

> **NOTE**
>
> You don't return the value of the image data here, because the framework will let you bind to the data in the client, as you will see later.

```
        [FormPropertyAttribute(FormPropertyKind::BindableValue, "ImageField")]
        public str parmImageField(anytype _value = imageFieldProperty.parmValue())
        {
            if(!prmisdefault(_value))
            {
                imageFieldProperty.setValueOrBinding(_value);
            }
            return "";
        }
```

4. Initialize **imageFieldProperty** by adding the highlighted line in the following example to the new method of **FMTContactControl**.

```
        public void new(FormBuildControl _build, FormRun _formRun)
        {
            super(_build, _formRun);

            this.setTemplateId("FMTContactControl");
            this.setResourceBundleName("/Resources/Html/FMTContactControl");

            dataSourceProperty = this.addProperty(methodstr(FMTContactControl,
parmDataSource), Types::String);
            imageFieldProperty = this.addProperty(methodstr(FMTContactControl,
parmImageField), Types::String);
            titleFieldsProperty = this.addProperty(methodstr(FMTContactControl,
parmTitleFields), Types::Class);
            subtitleFieldsProperty = this.addProperty(methodstr(FMTContactControl,
parmSubtitleFields),
               Types::Class);
            actionsProperty = this.addProperty(methodstr(FMTContactControl, parmActions),
Types::Class);
        }
```

5. Now supply the binding to **imageFieldProperty** by adding the highlighted line in the following example to the **applyBuild** method of **FMTContactControl**.

```
        ...
        if(build)
        {
            this.parmDataSource(build.parmDataSource());
            this.parmImageField(FormBindingUtil::initBinding(build.parmImageField().parmDataSou
            rce(), build.parmImageField().parmDataField(), this.formRun()));


        ...
```

6. Press Ctrl+S to save the changes. You've now finished modifying the run-time class. Next, you will update the HTML view to display the image.

Modify the HTML for the contact control

The HTML of the contact control is where you add UI elements, such as text boxes, images, and buttons, that interact with the properties and commands that are defined in the run-time class. Extensible controls use a declarative HTML-based binding syntax to bind HTML element behaviors to properties, commands, JavaScript expressions, and JavaScript functions. These bindings are parsed at run time, and the resulting HTML is injected

into the DOM. The following section explains a few of the bindings that are used in FMTContactControl.htm to add an image to the control.

Technical overview

The **bind** attribute, together with the **text** binding handler enables binding to the **text** property of an HTML element. For example, the following HTML uses the **bind** attribute and the **text** binding handler.

```
<span data-dyn-bind="text:'Hello World!'"></span>
```

The preceding HTML is equivalent to the following HTML.

```
<span>Hello World!</span>
```

You will see the benefits of the binding when you bind to properties or commands. For example, if you have a view model property that is named **FirstName**, you can bind to it as shown in the following example. Here, **$data** is the object that contains the view model properties and commands.

```
<span data-dyn-bind="text: $data.FirstName"></span>
```

The HTML output changes, based on the current value of **FirstName**. The following example shows the output if **FirstName** has a value of **John**.

```
<span>John</span>
```

If the value of the **FirstName** property changes for some reason (for example, X++ or JavaScript was run to update the property), the binding is automatically reevaluated, and the HTML output immediately reflects the change. All binding handlers follow this pattern of automatic reevaluation when the binding value changes. The **if** and **foreach** binding handlers are unique in that they perform DOM manipulation based on the binding values.

- To conditionally add an element to the DOM, use the **if** binding handler and supply the condition under which the element should be added. If the condition is false, the element isn't added to or removed from the DOM, and no bindings that are associated with the element are evaluated. Of course, if the binding value that is supplied to **if** changes, an element that was removed will be added to the DOM again, and the bindings will be evaluated.
- To iterate over an array of elements, use the **foreach** binding. This is useful when nearly identical HTML elements must be displayed.

The following table shows some of the other binding handlers.

BINDING HANDLER	DESCRIPTION
css	Specify a CSS class, based on a condition.

BINDING HANDLER	DESCRIPTION
style	Apply CSS styles, and bind the values to properties.
attr	Bind an HTML attribute.

In addition to using HTML elements inside the HTML for your control, you can also add framework controls such as CheckBox, Group, Tile, SectionContainer, Label, and List to your control. Instead of binding handlers, each framework control enables binding values to be passed to its view model properties. For example, a **CommandButton** is added by using the **role** attribute.

```
    <div data-dyn-role="CommandButton" ></div>
```

In this case, **ActionCommand** can be supplied with a JavaScript function.

```
    <div data-dyn-role="CommandButton"
         data-dyn-bind="ActionCommand: function(){alert('Hello World!')}"></div>
```

One additional feature of the HTML binding syntax is the context-aware nature of bindings. By default, the context of all HTML elements is set to the JavaScript view model for the control. However, the context changes in certain circumstances. For example, for a **foreach** binding, every child element that is nested inside the hosting element (the element that has the **foreach** binding) obtains the current item in the loop as the context. To access the context of the parent element when you're inside of a **foreach** binding, use the **$parent** object. The following example from FTMContactControl.htm will help make this point clearer.

```
<div data-dyn-bind="foreach: Actions">
    <div data-dyn-role="CommandButton"
        data-dyn-bind="ImageType: 'Symbol', ImageName: $data.ActionName,
        ActionCommand: function(){$parent().ButtonClicked($data)}">
    </div>
</div>
```

Actions is a List property that is available on the control's JavaScript view model. This property was defined in the **FMTContactControl** run-time class. Each action in the **Actions** list has **Data Source**, **Data Field**, and **Action Name** properties. Within the **foreach** loop, **$data** refers to the current action, and **$data.ActionName** cam retrieve the **ActionName** property from the current action in the loop. Within the loop, view model properties on the control aren't accessible via **$data**. Instead, **$parent** can be used to retrieve the view model properties.

Tutorial steps

Add the HTML for the **ImageField** property that you created in the run-time class.

1. In Solution Explorer, expand the **Resources** folder under the **FMTutorial** project, and double-click **FMTContactControlHTM**. The FMTContactControl.htm file opens in the HTML editor.
2. Add the following HTML to the FMTContactControl.htm HTML. The gray text is shown just for placement context.

```
    <div class="contact">

        <span data-dyn-bind="if: $data.ImageField">
            <span data-dyn-role="Image" data-dyn-bind="Height: '70', Width: '70', Value:
$data.ImageField"></span>
        </span>

...
```

3. Press Ctrl+S to save the changes to FMTContactControl.htm.

In the preceding example, you use the framework image control to render the image. **Value** is a property that is defined on the Image control. This property lets you specify the value for the image data. The image control supports several kinds of image types, but for this example, you're concerned with only two possible types: URLs and Base64 strings. Because the image type depends on data that is known only at run time, you will use a property that derives this information, **ImageValue**. You might notice that no such property is defined in the run-time class for **FMTContactControl**. Therefore, this property isn't part of the automatically generated JavaScript view model for that control, and it also isn't defined on **$data**. To make the **ImageValue** property accessible via **$data**, you must extend the automatically generated JavaScript view model to add the property.

Review the JavaScript for the contact control

As was mentioned earlier, for every X++ method that has either a **FormPropertyAttribute** or **FormCommandAttribute** attribute, a JavaScript property or command is generated and made accessible to an extensible control's HTML via the view model. You can extend this view model with additional properties and commands that are defined only on the client. In other words, the properties and commands have no associated X++ methods. After you extend the view model, the additional client-only properties and commands can be used in bindings via the **$data** object.

Technical overview

The Control Extensibility Framework offers many functions that help with data bindings and data access. Some of the functions that are used in FMTContactControl.htm, such as **$field** or **$model**, make it easy to access the data source and its fields from the HTML bindings. These functions are functional aliases that are used in the HTML bindings for JavaScript functions that are defined by the framework. Within the extended JavaScript view model, the equivalent, non-aliased functions are **$dyn.getField** and **$dyn.getModel**. You can also use jQuery within the extended JavaScript view model by using the **$** symbol. The following example shows the standard pattern that is used to define a constructor for the extended JavaScript view model. In this example, you save a reference to **this**, apply the base **Control** class behaviors, and then combine the automatically generated properties and commands with the properties and command from the extended view model.

```
        $dyn.controls.ContactControl = function (props) {
        var self = this;

        $dyn.ui.Control.apply(this, arguments);

        $dyn.ui.applyDefaults(this, props, $dyn.ui.defaults.ContactControl);
...
```

The **self** variable now contains all properties and commands that are generated from the X++ run-time class. The following example shows how to add a client-only property to extend the view model.

```
self.ActionTypes = {
            Email: function (action) {
                var url = 'mailto:' + $dyn.value(action.Value);
                self.Open(url);
            },
            Phone: function (action) {
                var number = $dyn.value(action.Value);
                number = number.replace(/\D/g, '');
                if (number.length == 10) {
                    // assume US country code
                    number = "1" + number;
                }
                var tel = "callto://+" + number;
                self.Open(tel);
            }
        };
```

The **self** variable will then contain all the properties and commands that are generated from the X++ run-time class, and also the **ActionTypes** property that was added as a client-only property. There are many more topics that are related to building view models for controls, but they are outside the scope of this tutorial. For this tutorial, we don't need to make any changes to the view model for **FMTContactControl**. Therefore, you can close the FMTContactControl.js file and proceed to the next section.

Add the extensible control to the Fleet Management workspace

You will now update the **Fleet Management Clerk** workspace so that it uses the contact control that you just completed.

1. In Solution Explorer, expand **Forms**, and then double-click **FMTPickingUpTodayPart**. The form opens in the form designer.
2. In the form designer, expand **Design** > **PickingUpTodayGrid**.
3. If there is an existing contact control, delete it. You must remove and then re-add the control, so that the form designer picks up the X++ changes that you made. Right-click the existing control, and then click **Delete**.
4. Right-click **PickingUpTodayGrid**, point to **New**, and then click **FMT Contact Control**.
5. Click the **FMTContactControl1** node that you just added, and set the **Data Source** property to **FMTCustomer**.
6. Expand the **FMTContactControl1** node, click **Image**, and then, in the **Properties** pane, set the following properties.

PROPERTY	VALUE
Data Source	FMTCustomer
Data Field	Image

7. Create new title fields:
    a. Right-click **Title Fields**, and then click **New Title Field**.
    b. Click the **Title Field** node that you just created, and then, in the **Properties** pane, set the following properties.

PROPERTY	VALUE
Name	FirstName
Data Source	FMTCustomer
Data Field	FirstName

c. Right-click **Title Fields** again, and then click **New Title Field**.

d. Click the **Title Field** node that you just created, and then, in the **Properties** pane, set the following properties.

PROPERTY	VALUE
Name	LastName
Data Source	FMTCustomer
Data Field	LastName

8. Create new subtitle fields:

a. Right-click **Subtitle Fields**, and then click **New Subtitle Field**.

b. Click the **Subtitle Field** node that you just created, and then, in the **Properties** pane, set the following properties.

PROPERTY	VALUE
Name	StartDate
Data Source	FMTRental
Data Field	StartDate
Formatting Expression	Pickup {0}

c. Right-click ***Subtitle Fields*** again, and then click ***New Subtitle Field***.

d. Click the **Subtitle Field** node that you just created, and then, in the **Properties** pane, set the following properties.

PROPERTY	VALUE
Name	EndDate
Data Source	FMTRental
Data Field	EndDate
Formatting Expression	Return {0}

e. Right-click **Subtitle Fields** again, and then click **New Subtitle Field**.

f. Click the **Subtitle Field** node that you just created, and then, in the **Properties** pane, set the following properties.

PROPERTY	VALUE
Name	VehicleDescription
Data Source	FMTVehicle
Data Field	Description

g. Press Ctrl+S to save your changes.

9. Copy **PickingUpTodayGrid** by right-clicking in the grid and clicking **Copy**.
10. In Solution Explorer, click **Forms** > **FMTReturningTodayPart**, and then double-click **FMTReturningTodayPart**. The form opens in the form designer.
11. Expand **Design**, right-click **ReturningTodayGrid**, and then click **Delete**.
12. Right-click **Design**, and then click **Paste**.
13. Select the **PickingUpTodayGrid** grid that you just added to the **FMTReturningTodayPart** form. Set the **Name** property to **ReturningTodayGrid**, and then press Ctrl+S to save the changes to the **EMTReturningTodayPart** form.
14. In Solution Explorer, find the **FMTRentalRatesPart** form. Double-click the form to open it in the form designer, and then click **Design** > **RentalRatesGrid**.
15. Delete each field from **RentalRatesGrid**. To remove the fields, click the first field, hold down the Shift key while you click the last field, and then press Delete.
16. Right-click in the grid, point to **New**, and then click **FMT Contact Control**.
17. Expand **FMTContactControl1**, click **Image**, and then, in the **Properties** pane, set the following properties.

PROPERTY	VALUE
Data Source	FMTVehicleModel
Data Field	Image

18. Right-click **Title Fields**, and then click **New Title Field**.
19. Click the title field node that you just created, and then, in the **Properties** pane, set the following properties.

PROPERTY	VALUE
Name	VehicleModel
Data Source	FMTVehicleModel
Data Field	Model

20. Right-click **Subtitle Fields**, and then click **New Subtitle Field**.

21. Click the **Subtitle Field** node that you just created, and then, in the **Properties** pane, set the following properties.

PROPERTY	VALUE
Name	VehicleMake
Data Source	FMTVehicleMake
Data Field	Make

22. Right-click **Subtitle Fields**, and then click **New Subtitle Field**.
23. Click the **Subtitle Field** node that you just created, and then, in the **Properties** pane, set the following properties.

PROPERTY	VALUE
Name	RatePerDay
Data Source	FMTModelRate
Data Field	RaterPerDay **Note:** The **Data Field** value must match the table field name. If you correct the spelling error, the values won't match, and you will receive a run-time error.
Formatting Expression	${0} per day

24. Right-click **Subtitle Fields**, and then click **New Subtitle Field**.
25. Click the **Subtitle Field** node that you just created, and then, in the **Properties** pane, set the following properties.

PROPERTY	VALUE
Name	RatePerWeek
Data Source	FMTModelRate
Data Field	RatePerWeek
Formatting Expression	${0} per week

26. Press Ctrl+S to save your changes to **FMTRentalRatesPart**.
27. In Solution Explorer, right-click the **FMTClerkWorkspace** form, and then click **Set as Startup Object**.
28. Press Ctrl+F5 to open the updated contact control in Internet Explorer.

> **NOTE**
>
> If you receive a JavaScript error, you might have to clear the Internet Explorer cache, so that the browser loads the new JavaScript file:

a. When you're prompted to open the debugger, click **No**.
b. While Internet Explorer is open, press F12 (or click **Settings** > **F12 Developer Tools**), and then press Ctrl+R.
c. In the **Clear Browser Cache** dialog box, click **Yes**.
d. Reload the page by pressing Ctrl+F5.

In this tutorial, you've seen how you can use X++ when you define the design-time and server-side behaviors for a control, and how you can consume a powerful HTML-based and JavaScript-based framework when you design the UI and user interaction patterns. The Control Extensibility Framework helps provide a separation between the modeled behavior of a control and its physical manifestation. As a best practice, you should try to maintain this loose coupling between data, metadata, and UI when you build extensible controls.

Bidirectional or right-to-left support

To validate right-to-left (RTL) support for your extensible control, you simply need to set the **dir** (direction) attribute on the HTML document. When this attribute is changed, the browser will automatically change the layout direction of your control. You should make sure that your control doesn't implement any styling which interferes with this layout. Instead of setting this attribute manually, you can also validate by placing your control on a form, and then selecting a RTL language. Selecting a RTL language will cause the client to also update the **dir** attribute appropriately. For more information, see dir attribute in the HTML standards.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Keyboard shortcuts for extensible controls

2/18/2021 • 5 minutes to read • Edit Online

Keyboard shortcuts are an important consideration when you create any extensible control. This topic provides information that will help you choose keyboard shortcuts for your extensible controls. It also outlines the recommended method for implementing keyboard shortcuts for extensible controls.

Overview

For accessibility, it's essential that keyboard-only users be able to use controls. Therefore, keyboard shortcuts are an important consideration when you create any extensible control. This topic provides information that will help you choose key combinations to use as keyboard shortcuts. It highlights the shortcuts that are currently used by Finance and Operations apps and supported browsers, shortcuts that are planned for implementation, and shortcuts that one or more browsers don't allow to be overridden. This topic also outlines the recommended way to implement keyboard shortcuts for extensible controls.

Choosing a key combination

When you're trying to choose a key combination to use as a keyboard shortcut, it's important that you be aware of other existing shortcuts. In this way, you help guarantee that your shortcut won't overlap an existing shortcut. If you try to collide with an existing shortcut, one of the following outcomes might occur:

- The new keyboard shortcut might not work, because a browser doesn't allow that key combination to be overridden, or a framework-provided shortcut takes precedence over the new shortcut.
- The new keyboard shortcut might remove expected keyboard functionality, because users expect specific key combinations to perform specific functions in a browser. Alternatively, you might override framework-provided shortcuts or other control shortcuts, so that keyboard-only users can't use them.

Because of these potential issues, we recommend that you adhere to this guidance when you choose a key combination:

- **Don't** choose any key combination that is currently used by Finance and Operations apps, or that is planned for future implementation.
- **Do** pick key combinations that will work in all supported browsers.
- **Do** be careful when you override shortcuts that are used by a supported browser. You should not suppress shortcuts for important or frequently used browser functionality.
- **Do** use longer key combinations (three keys) for control-specific behavior. Shorter combinations should be reserved for user-defined keyboard shortcuts.
- **Don't** choose any key combination that involves Ctrl+Alt, because this combination maps to Alt+Gr for some Eastern European languages and will conflict with other shortcuts.

Keyboard shortcut links

Here are links to the keyboard shortcuts that are documented for Finance and Operations apps and supported browsers:

- Keyboard shortcuts
- Microsoft Edge
- Google Chrome
- Internet Explorer 11
- Apple Safari

Planned keyboard shortcuts

In addition to the keyboard shortcuts that are currently used, there are several shortcuts that are planned for future implementation. To avoid conflicts with framework-provided shortcuts, you should not choose the following key combinations for extensible controls.

SHORTCUT	FUNCTIONALITY
F3	Move to the nearest QuickFilter.
Alt+F3	Add a filter that is based on the value of the current control (Filter by value).
Alt+Shift+F3	Clear all user-defined filters.
F6	Move to the nearest toolbar.
Shift+F7	Move to the toast message.

Browser/operating system keyboard shortcuts to avoid

Keyboard shortcuts that correspond to important functionality

The following table provides a short, non-exhaustive list of keyboard shortcuts that correspond to important functionality in a browser or operating system. You should not choose the key combinations in this table.

SHORTCUT	FUNCTIONALITY
Ctrl+A	Select all the text in the current field, or select all content on the page.
Ctrl+C	Copy.
Ctrl+V	Paste.
Ctrl+X	Cut.
F5	Refresh the page.
Ctrl+F5	Refresh the page, and ignore cached content.
Shift+F10	Simulate a right-click.
Tab / Shift+Tab	Move to the next/previous control.
Ctrl+Tab / Ctrl+Shift+Tab	Move to the next/previous browser tab.
Alt+Tab / Alt+Shift+Tab	Move to the next/previous application.
Alt+Right arrow / Alt + Left arrow	Go to the next/previous page in the browser history.

Keyboard shortcuts that can't be overridden by some browsers

Some browsers don't allow the following keyboard shortcuts to be overridden. Therefore, you should not choose the following key combinations, because the shortcut won't work in all browsers.

Alt+A	Alt+T	Ctrl+F4	Alt+Tab

Alt+C	Alt+V	Alt+F5	Alt+Shift+Tab
Alt+D	Ctrl+W	Alt+F6	Spacebar
Alt+Shift+D	Ctrl+Shift+W	Alt+Shift+F6	Alt+Shift+Backspace
Alt+E	Alt+X	F12	Ctrl+Pause/Break
Alt+F	Ctrl+Shift+0	Ctrl+Esc	Ctrl+Shift+Pause/Break
Alt+H	Ctrl+Shift+7	Ctrl+Shift+Esc	Ctrl+Page up
Ctrl+N	F1	Alt+Esc	Ctrl+Page down
Ctrl+Shift+N	Ctrl+F1	Alt+Shift+Esc	Ctrl+Plus sign (+)
Ctrl+Shift+Q	Ctrl+Shift+F1	Ctrl+Tab	Shift+Plus sign (+)
Ctrl+T	Shift+F1	Ctrl+Shift+Tab	Ctrl+Minus sign (–)

Implementing keyboard shortcuts

We recommend that you use the registration mechanism that is described in this section to implement keyboard shortcuts. There are several benefits to registering a control's shortcuts in this manner:

- You specify only the modifier keys that you want. If any other modifiers are pressed, the shortcut won't run. Therefore, a keyboard shortcut that is defined for Ctrl+Down arrow won't be triggered if Ctrl+Shift+Down arrow is pressed.
- You can reuse code.
- If you return **false** from the handler function, propagation won't stop on the event. If you return **true** (or if you don't specify a return value), propagation will stop.
- Ctrl and Meta modifiers are treated the same. Therefore, you can specify keyboard shortcuts that will work for both iOS and Microsoft Windows. For example, the shortcut Ctrl+G on Windows will be translated to Meta+G on iOS.
- Built-in telemetry for the keyboard shortcuts is used.

Define a keyboard shortcut

1. Define a Shortcuts object on the control's prototype. Then define the keyboard shortcuts inside the Shortcuts object. These shortcuts should have the following structure. Make sure that the key code that you're trying to use is defined in the $dyn.ui.KeyCodes object.

```
Shortcuts: {
    Name: {
        Keys: { modifier1: true, modifier2:true,
            keyCode: $dyn.ui.KeyCodes.* },
        // Only specify the modifiers you need
        // (between alt, ctrl/meta, shift)
        Handler: function (evt) {
            // Code to handle shortcut.
        },
        // Additional code.
}
```

   If more than one key code should apply to your keyboard shortcut, pass in an array of codes, as shown

here.

```
Keys: {modifier1:true,
    keyCode:[$dyn.ui.KeyCodes.*, $dyn.ui.KeyCodes.*, ... ] }
// Note: If any of these keyCodes match then the handler is called. I.E.
// The keyCodes in the array are OR'd not AND'd
```

2. Add the keydown handler by using the following code.

```
keydown: function (event) {
    $dyn.util.handleShortcuts(this, event);
},
```

3. Bind the keyDown handler. The keyDown handler is automatically bound for form objects. Other controls must be manually bound to the keyDown handler, as for any regular binding.

> **IMPORTANT**
> "keyDown" is case sensitive.

```
<div data-dyn-bind="keyDown: $data.keydown"></div>;
```

Examples

Here is a Form example.

```
Shortcuts: {
    Save: {
        Keys: { ctrl: true, keyCode: $dyn.ui.KeyCodes.letterS },
        Handler: function (evt) {
            var control = evt ? $dyn.context(evt.target) : undefined;
            this.executeShortcuts(true, "SaveRecord", control);
        },
    },
    New: {
        Keys: { alt: true, keyCode: $dyn.ui.KeyCodes.letterN },
        Handler: function (evt) {
            var control = evt ? $dyn.context(evt.target) : undefined;
            this.executeShortcuts(true, "NewRecord", control);
        },
    },
    Delete: {
        Keys: { alt: true, keyCode: $dyn.ui.KeyCodes.deleteKey },
        Handler: function (evt) {
            this.executeShortcuts(false, "DeleteRecord");
        },
    },
    // Additional code
}
```

Here is a Dialogs example that extends the Form control.

```
Shortcuts: $dyn.extendPrototype($dyn.controls.Form.prototype.Shortcuts, {
    InvokeDefaultButton: {
        Keys: { keyCode: $dyn.ui.KeyCodes.enter },
        Handler: function (evt) {
            this.executeShortcuts(false, "InvokeDefaultButton");
        }
    },
})
```

Extensible control programming reference

2/18/2021 • 34 minutes to read • Edit Online

This topic provides reference content for extensible control programming.

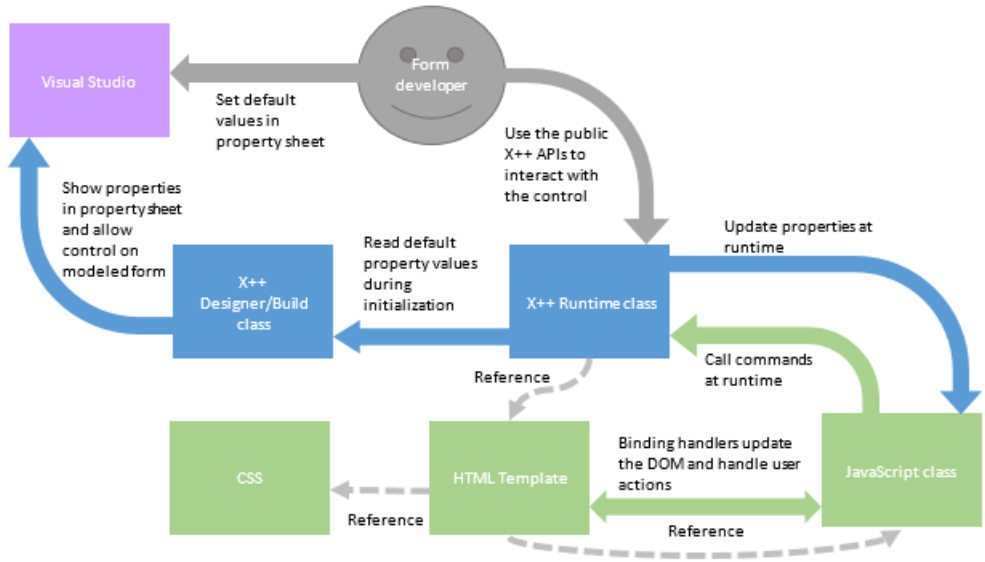
This document describes the API, HTML, and JavaScript support for creating extensible controls.

Examples

This document contains small code snippets that show how to use each API that is documented. More complete examples of finished controls that leverage many of these APIs can be found on Github. Extensible Control Examples on Github

Control block diagram

This high-level diagram illustrates the key components of an extensible control and how they interact with each other. Your extensible control solution will contain two X++ classes that implement your control. The runtime class implements the runtime data, presentation, and behavior of your control. The build class defines how your control is displayed in Form Designer, Property Window, and Application Explorer.



X++

The X++ API of your control is the Form-developer-facing API. Be sure to consider the APIs and behaviors you want to provide to the Form developer when designing the X++ APIs for your control.

Runtime: The X++ runtime class

The runtime class defines the public, developer-facing API for your control. It also contains the runtime logic for your control. The job of the runtime class is to maintain the state of the control via the control's properties.

Runtime: Class declaration

Declare an X++ class that extends **FormTemplateControl** or a type derived from **FormTemplateControl**. ***FormTemplateControl* contains basic properties that are necessary for every control, such as the Template ID and the Resource Bundle. The following example extends the base control class, **FormTemplateControl**.

```
class MyControl extends FormTemplateControl
```

Runtime: FormControlAttribute

You must apply the **FormControlAttribute** attribute to the X++ class declaration.

```
[FormControlAttribute(<Template ID>, <Resource Bundle Path>, <Build class name>))]
```

You must supply the following arguments:

- **Template ID**: A string that specifies the ID of the template. The Template ID is the JavaScript class name & the HTML element ID of the control. By convention, the TemplateID matches the class name of the control's runtime class.
- **Resource Bundle Path**: A string that specifies the path to the resource bundle. The Resource Bundle Path is the web path where the main HTM files are located. At runtime, the Client framework will load the HTM file specified by the Resource Bundle Path. At runtime, the Client framework will search the Resource Bundle for the HTML element with an ID matching the specified Template ID. At runtime, the Client framework will instantiate the JavaScript class specified by the Template ID. This path is relative to the root of the web directory.
- **Build class name**: A string that specifies the name of the build class. The build class name is the X++ class that determines the design-time behavior. At design time, the Visual Studio framework reflects over this attribute and loads the specified X++ class as the designer class. At runtime, the X++ framework will instantiate the specified X++ class as the build class in the super of applyBuild().

The following example shows a typical class and attribute declaration for a control named "MyControl".

```
[FormControlAttribute('MyControl', '/resources/html/MyControl', classStr(MyControlBuild))]
class MyControl extends FormTemplateControl
```

Runtime: FormCommandAttribute

The **FormCommandAttribute** is applied to a method in your control class, which allows the method to be called from a control's JavaScript class. A method with this attribute applied is called a **command.** Use the **FormCommandAttribute** on only the X++ methods that need to be accessed directly from the control's JavaScript class. An X++ method serving as a command can only accept string arguments. The method must perform the necessary operations to serialize or deserialize the string arguments into other types. The **FormCommandAttribute** has no effect on the behavior of the X++ method when the method is used from within X++. The **FormCommandAttribute** exposes the X++ method as an external endpoint that is accessible from JavaScript. As such, every command should be threat modeled and tested for exploits, and should perform validation on all of its arguments. The underlying X++ method should be declared private so that it is not accessible from X++. If X++ code needs to access this method's behavior, then a separate X++ method should be declared as public without the **FormCommandAttribute.** This public method should contain any shared code that is needed by both X++ and JavaScript. The private X++ method with the **FormCommandAttribute** can then call this public method to access the shared code. This practice allows the command to perform logic that is specific to calls coming from JavaScript (such as argument type deserialization, argument validation, security validation, etc.) before executing the core shared X++ logic. You supply the following arguments to the **FormCommandAttribute** constructor:

- **Name**: A required string that specifies the name of the command. A few best practices for naming Properties:
  - Capitalize the first letter, and use PascalCase.

    - Use a verb in the name.
    - Include the Property name if the Command is used to read/write a FormProperty (ex: Set_<PropertyName>)
    - Do not use any of the names of inherited JavaScript properties.
- **Execute immediate**: An optional boolean that specifies whether calls to this command are deferred or require immediate execution. By default, commands have **Execute immediate** set to **true,** so calls are not deferred. Most commands likely require immediate execution because their X++ logic must run before allowing the user to complete their next action. However, commands that have no side-effects on the user's ability to take their next action can likely be safely deferred to gain a performance benefit. For commands that can be deferred, set Execute immediate to **false** to reduce network chattiness.

The following example declares a command with the name of "SetText".

```
[FormCommandAttribute("SetText")]
private void setText(str value)
{
    // Add implementation code here.
}
```

Runtime: FormPropertyAttribute

The **FormPropertyAttribute** is applied to a method in your control class, which allows an X++ method to be called as a **FormProperty** getter/setter from the control's JavaScript class. A method with this attribute applied is called a **property.** Only use the **FormPropertyAttribute** on those X++ methods which need to be accessed directly from the control's JavaScript class. The **FormPropertyAttribute** has no effect on the behavior of the X++ method when the method is used from within X++. Every property exposes an endpoint to the browser. As such, every property should be threat modeled and tested for exploits. The underlying X++ method should be declared private so that it is not accessible from other X++ code. If other X++ code needs to access the property, then declare a separate public X++ method without the **FormPropertyAttibute,** and move the shared property logic to this method. Then call this method from the private X++ method with the **FormPropertyAttribute. This practice allows the property to perform logic that is specific to calls coming from JavaScript (such as argument type deserialization, argument validation, security validation, etc.) before executing the core shared X++ logic.** The underlying X++ method must accept and return the desired type of the property. If the desired type if an EDT, the property must accept and return the base type of the EDT. The supported property types are:

- X++ primitive types
- X++ data contracts (whose members are also supported types)
- X++ List (whose items are also supported types)

You supply the following arguments to the **FormPropertyAttribute** constructor:

- **FormPropertyKind:** A required **FormProperyKind** value that specifies the type of the property. Use **FormPropertyKind::Value** for Properties not bound to a data source field, and use **FormPropertyKind::BindableValue** for properties that may be bound to a data source field.
- **Name:** A required string that specifies the name of the property. A few best practices for naming properties:
    - Capitalize the first letter, and use PascalCase.
    - Do not use any of the names of inherited JavaScript properties
- **Read only**: An optional boolean that specifies whether this property is writable from the control's JavaScript class. By default, properties have **Read only** set to **false,** so they are writeable. This argument does not affect the ability to write to this property from X++. To make the X++ method read only, remove all method arguments from the method declaration. A majority of properties should not be writable from the control's JavaScript class. Because most property values require validation, a command should be used as a setter for

the property so that validation logic is can be run before the backing property is set.

- **Execute immediate**: An optional Boolean that specifies whether writes to this property are deferred or require immediate execution. By default, properties have **Execute immediate** set to **false,** so writes are deferred. Because the majority of properties should not be writeable form the control's JavaScript class, the **Execute immediate** flag defaults to **false** and provides performance benefits. Even in the case of properties that are writable from the control's JavaScript class, the performance side effects of immediate execution should be carefully considered before enabling the behavior.

The following example shows a typical property declaration. Most properties share the same boilerplate code for getting/setting, as shown below. The textProperty variable is the backing FormProperty field for this property.

```
[FormPropertyAttribute(FormPropertyKind::Value, "Text", true)
private str parmText(str _value = textProperty.parmValue())
{
    if(!prmIsDefault(_value))
    {
        textProperty.setValueOrBinding(_value);
    }
    return textProperty.parmValue();
}
```

Runtime: FormProperty

Behavior

FormProperty is an X++ derived type used by the control framework for the synchronization of property values between X++ and JavaScript. **FormProperty** objects are considered the backing fields used internally by properties. Each **FormProperty** is typically used in 4 places throughout a control's X++ runtime class:

1. The **FormProperty** is declared, usually right below the class declaration
2. The **FormProperty** is instantiated in the **new** method of the class
3. The **FormProperty** is initialized in the **applyBuild** method of the class
4. The **FormProperty** is read and written in the X++ method for the property

The following example shows a **FormProperty** being used in a typical controls' X++ runtime class.

```
[FormControlAttribute("MyControl", "/resources/html/MyControl", classStr(BuildMyControl))]
class MyControl extends FormTemplateControl
{
    FormProperty textProperty;

    public void new(FormBuildControl _build, FormRun _formRun)
    {
        super(_build, _formRun);

        this.setTemplateId("MyControl");
        this.setResourceBundleName("/resources/html/MyControl");
        textProperty = this.addProperty(
        methodStr(MyControl, parmText), Types::String);
    }

    public void applyBuild()
    {
        BuildMyControl build;

        super();

        build = this.build();
        if(build)
        {
            this.parmText(build.Text());
        }
    }

    [FormPropertyAttribute(FormPropertyKind::Value, "Text", true)
    private str parmText(str _value = textProperty.parmValue())
    {
        if(!prmIsDefault(_value))
        {
            textProperty.setValueOrBinding(_value);
        }
        return textProperty.parmValue();
    }
}
```

Runtime: new method

The **new** method on a control's X++ runtime class is called as a part of instantiating the control on a form. For the details on when the **new** method is called in the form lifecycle, please see the Control Lifecycle Diagrams. This method is used for instantiation of a control's FormProperties and setting the control's Template ID and Resource Bundle Path. See typical use of the **new** method in the example for FormProperty.

Runtime: applyBuild method

The **applyBuild** method on a control's X++ runtime class is called as a part of instantiating the control on a form. For the details on when the **applyBuild** method is called in the form lifecycle, please see the Control Lifecycle Diagrams. This method is used for initialization of a control's FormProperties to their default values, or to the values specified by the form developer who placed the control on the form. See typical use of the **applyBuild **method in the example for FormProperty.

Runtime: FormBindingUtil::initbinding method

The **FormBindingUtil** is an API provided by the control framework. It is used to bind FormProperties to data fields and data methods on a data source. The following example binds the data field with name "Value" on the data source with name "DataSource1" to the textProperty FormProperty of the runtime class.

```
[FormControlAttribute("MyControl", "/resources/html/MyControl", classStr(BuildMyControl))]
class MyControl extends FormTemplateControl
{
    FormProperty textProperty;

    public void new(FormBuildControl _build, FormRun _formRun)
    {
        super(_build, _formRun);
        this.setTemplateId("MyControl");
        this.setResourceBundleName("/resources/html/MyControl");
        textProperty = this.addProperty(
        methodStr(MyControl, parmText), Types::String);
    }

    public void applyBuild()
    {
        BuildMyControl build;

        super();

        build = this.build();
        if(build)
        {
            this.parmText(FormBindingUtil::initBinding(
            "DataSource1", "Value", this.formRun()));
        }
    }

    [FormPropertyAttribute(FormPropertyKind::Value, "Text", true)
    private str parmText(str _value = textProperty.parmValue())
    {
        if(!prmIsDefault(_value))
        {
            textProperty.setValueOrBinding(_value);
        }
        return textProperty.parmValue();
    }
}
```

Design time: The X++ build class

The build class defines the design time behavior of your control. This class determines which properties appear in the property sheet, and how the control behaves when it is modeled in the Form designer. The job of the design time class is to capture design time information for the runtime class to access later on.

Design time: Class declaration & FormDesignControlAttribute

The FormDesignControlAttribute is necessary for the control to appear in the Visual Studio Form designer when right-clicking on the design node of the form. If the FromDesignControlAttribute is missing, then the control can only be added to a form via imperative X++ code (i.e. via the addControlEx method on the form).

```
[FormDesignControlAttribute("MyControl")]
class MyControlBuild extends FormBuildControl
{

}
```

Design time: FormDesignPropertyAttribute

Placing this attribute on a method in the design time class will result in a new property with the name specified

by the first argument (and in the section specified by the second argument) appearing the property sheet for this control, with the corresponding X++ method operating as the getter/setter for the property.

```
[FormDesignControlAttribute("MyControl")]
class MyControlBuild extends FormBuildControl
{
    str text;

    [FormDesignPropertyAttribute("Text", "Data")]
    public str Text(str _value = text)
    {
        if(!prmIsDefault(_value))
        {
            text = _value;
        }
        return text;
    }
}
```

Design time: FormDesignProperty** **Attribute

There are a number of FormDesignProperty attributes which may be applied alongside the standard FormDesignPropertyAttribute for specialized behavior in the property sheet. The specialized behavior includes enabling the property as a combobox which allows selecting from a list of a values. The different types of lists that used are enumerated below. Whenever the user selects an item from the combobox, the string name of that item is passed into the X++ method getter/setter with the attribute.

- [FormDesignPropertyDataSourceAttribute] - Shows a list of the data sources on the form.
- [FormDesignPropertyDataFieldAttribute(<data source name>)] - Shows a list of the data fields on the specified data source.
- [FormDesignPropertyDataMethodAttribute(<data source name>)] - Shows a list of the data methods on the specified data source.
- [FormDesignPropertyFieldGroupAttribute(<data source name>)] - Shows a list of the field groups on the specified data source.
- [FormDesignPropertyExtendsClassAttribute(<class name>)] - Shows a list of the classes which extend the specified class.
- [FormDesignPropertyImplementsAttribute(<interface name>)] - Shows a list of the classes which implement the specified interface.
- [FormDesignPropertyReferenceAttribute(<FormDesignPropertyReferenceType::<type>)] - Shows a list of the specified AOT artifacts with the given type. The supported types are:
  - Table
  - View
  - Map
  - EDT
  - BaseEnum
  - Query
  - Class
  - Form
  - MenuItemDisplay
  - MenuItemOuput
  - MenuItemAction
  - Tile
  - KPI

The following example shows standard properties used to allow a Form developer to specify the Data Source and Data Field for the design time class.

```
[FormDesignControlAttribute("MyControl")]
class MyControlBuild extends FormBuildControl
{
    str dataSource;
    str dataField;
    str dataMethod;

    [FormDesignPropertyAttribute("Data source", "Data"),
     FormDesignPropertyDataSourceAttribute]
    public str DataSource(str _value = dataSource)
    {
        if(!prmIsDefault(_value))
        {
            dataSource = _value;
        }
        return dataSource;
    }

    [FormDesignPropertyAttribute("Data Field", "Data"),
     FormDesignPropertyDataFieldAttribute(methodStr(MyControlBuild, DataSource))]
    public str DataField(str _value = dataField)
    {
        if(!prmIsDefault(dataField))
        {
            dataField = _value;
        }
        return dataField;
    }

    [FormDesignPropertyAttribute("Data Method", "Data"),
     FormDesignPropertyDataMethodAttribute(methodStr(MyControlBuild, DataSource))]
    public str DataMethod(str _value = dataMethod)
    {
        if(!prmIsDefault(dataMethod))
        {
            dataMethod = _value;
        }
        return dataMethod;
    }
}
```

A control with a design time class like the one above can then bind to the specified data source and data field inside of the applyBuild method, as show below.

```
public void applyBuild()
{
    BuildMyControl build;

    super();

    build = this.build();
    if(build)
    {
        this.parmText(FormBindingUtil::initBinding(
        build.DataSource(), build.DataField(), this.formRun(), build.DataMethod()));
    }
}
```

If you supply both a data field and data method to FormBindingUtil::initBinding, the data field binding will override the data method binding.

HTML

HTML: Framework attributes

The following section documents the HTML attributes that are used in the control framework for control development.

data-dyn-bind

data-dyn-bind, the data binding attribute, standardizes many common DOM manipulations - such as modifying an element's attributes, properties and CSS, or handling DOM events - through a declarative HTML-based API. The data binding attribute allows for these behaviors without requiring complex JavaScript. Using the data binding attribute rather than writing complex JavaScript can save the control developer valuable time by making things such as designing, debugging and maintaining the control much easier. However, complex JavaScript is still available when scenarios require its use. The data binding attribute binds HTML element behaviors to values supplied by the control developer. The values supplied can be simple JavaScript variables, JavaScript comparison or arithmetic expressions, JavaScript functions and JSON objects. The values supplied can also be observable variables, created using the APIs described in this document. The way in which the supplied value is bound to the HTML element is determined by the binding handler that is used with the data binding attribute. A list of all supported binding handlers is provided in this document. The data binding attribute requires the following syntax when used with any binding handler. The syntax for **data-dyn-bind** is:

```
data-dyn-bind="[first binding handler]: [value to bind to]"
```

The data binding attribute accepts a comma-separated list of binding handler-value pairs, so you can supply more than one binding handler to the binding attribute at a time. The following example binds the **visible** property of the div element to true, and binds the **textContent** property of the div element to "Hi".

```
data-dyn-bind="text: 'Hi', visible: true"
```

The data binding attribute is a custom HTML attribute understood by the control framework. The data binding attribute can be applied to any HTML element. Some HTML elements may not have the behavior which the binding handler modifies. For example, using the text binding handler on an **<svg>** element will not show the text since the **<svg>** element does not have a textContent property. The control framework reads and executes the data bindings specified in the control's template at runtime. The lifecycle for the control in the browser can be summarized as follows:

1. The control's HTM file is loaded by the browser.
2. Any script or resource files referenced in the HTM file are also loaded by the browser. Steps 1 and 2 are executed only once during a user's session, even if there are multiple instances of the control.
3. The JavaScript class's constructor for the control is call and passed with the X++ properties for the control instance.
4. The control's template is copied from the HTM file and into the browser's memory.
5. HTML elements in the control's template are processed for data binding in hierarchical order (depth first), and data bindings on each element are executed in order from left-to-right
6. The final HTML, including the original data binding attributes as well as any other markup added by the binding handlers or by the framework, is added to the browser's DOM and rendered for the user to see.
7. Later, when the value of an observable changes, any binding handlers subscribed to the observable are re-executed and the live DOM is updated in real-time.

HTML: Binding handlers

attr

The **attr** binding handler applies the supplied HTML attribute and value to the element. For a list of HTML attributes see W3 Schools – HTML Attributes. The arguments are passed in as an object array. Each argument is dual-valued. The first value is the name of the attribute, and the second value is the value of the attribute.

- **Name**: a string that specifies the desired name of the attribute to create.
- **Value or Expression:** a string that specifies the value to set on the attribute. If an expression is supplied, the value returned by evaluating the expression will be used.

The following example creates the title and name attributes and sets their value.

```
<!-- the markup in the HTML template -->
<div data-dyn-bind="attr: {title: 'Hello', name: 'Greeting'}"></div>

<!-- the markup in the browser after the binding handler is applied -->
<div title="Hello" data-dyn-bind="attr: {title: 'Hello', name: 'Greeting'}" name="Greeting"></div>
```

The following example uses expressions and functions. However, using JavaScript functions as in-line HTML like the example below is not recommended.

```
<!-- the markup in the HTML template -->
<div data-dyn-bind="attr: {title: false? 'Hello':'World', name: function(){return 'Greetings';}()}"></div>

<!-- the markup in the browser after the binding handler is applied -->
<div title="World" data-dyn-bind="attr: {title: false? 'Hello':'World', name: function(){return
'Greetings';}()}" name="Greetings"></div>
```

click

Behavior

Subscribes the supplied function to the click event on the element. For more information on subscribing to the click event see jQuery – click().

Arguments

EventHandler (function)

The function to call when the event is raised.

Example 1

The following example shows an alert message “Hello” when the element is clicked.

```
// In your control’s code-behind JS file
<script>
... // boilerplate code
self.ElementClicked = function (event) {
    /* handle the click event */
    alert('Hello');
};
...
</script>

<!-- In your control’s template HTM file -->
<div data-dyn-bind="click: $control.ElementClicked"></div>
```

The following example prevents the click event on child elements from bubbling up to parent elements. The example below will show only one alert with message “Hello” when the child element is clicked.

```
// In your control’s code-behind JS file
<script>
... // boilerplate code
self.ParentElementClicked = function (event) {
    /* handle the click event */
    alert('Hi');
};

self.ElementClicked = function (event) {
    /* prevents the event form bubbling up to parent elements*/
    event.stopPropagation();

    /* handle the click event */
    alert('Hello');
};

...
</script>

<!-- In your control’s template HTM file -->
<div data-dyn-bind="click: $control.ParentElementClicked">
<div data-dyn-bind="click: $control.ElementClicked"></div>
</div>
```

The following example prevents the browser default behavior from executing. For anchor tags, the default hyperlink behavior is prevented, so the browser will not navigate to the link when the element is clicked.

```
// In your control’s code-behind JS file
<script>
... // boilerplate code
self.LinkClicked = function (event) {
        /* handle the click event */
        alert($dyn.format('Navigation to ' + {0} + ' was prevented', $(event.target).attr("href")));

        /* prevents the default event behavior */
        event.preventDefault();
};
</script>

<!-- In your control’s template HTM file -->
<a href="https://www.microsoft.com" data-dyn-bind="click: $control.LinkClicked">Click here</a>
```

css

Behavior

Adds or removes the specified CSS class name(s) to the element, based on the specified condition(s). Note that expressions supplied to the binding handler are only executed once.

Arguments

The arguments are passed in as an object array. Each argument is dual-valued. The first value is the Class name, and the second value is the Condition.

Class name (string)

The CSS class name to add to the element.

Condition (expression)

The condition on which to add the CSS class name. If the condition evaluates to true, the CSS class name is added. If the condition evaluates to false, the CSS class name is removed. If a supplied condition takes a dependency on an observable (via $dyn.value), then the condition will be re-evaluated whenever the observable value changes, and the associated CSS class name will be added/removed based on the new condition. The following example adds the CSS class names "red", "green", and "yellow".

```
// In your control's code-behind JS file
<script>
... // boilerplate code
self.red = function () { return true; };
self.yellow = $dyn.observable(true);
...
</script>

<!-- the markup in the HTML template -->
<div data-dyn-bind="css: {green: true, red: $control.red, yellow: $dyn.value($control.yellow)}"></div>

<!-- the markup in the browser after the binding handler is applied -->
<div class="green red yellow" data-dyn-bind="css: {green: true, red: $data.red, yellow:
$dyn.value($control.yellow) }"></div>
```

event

Behavior

Subscribes the supplied event handler to the specified DOM event. For a list of supported DOM events, see jQuery - Event.

Arguments

For details on the arguments to the event binding handler, see jQuery - .bind(). The following example subscribes to the mouseover event and shows an alert when the element is hovered.

```
// In your control's code-behind JS file
<script>
... // boilerplate code
self.elementHovered = function (event) { alert($dyn.format('{0}',$(event.target).text()))};
...
</script>

<!-- the markup in the HTML template -->
<div data-dyn-bind="event: {mouseover: $data.elementHovered}">Greetings!</div>
```

foreach

Behavior

Repeats the content of the child element, updating the binding context of each child based on the supplied data. Supply ***only one*** child element inside of the element with the **foreach** binding. This one element is the element that will be cloned and repeated. Any other additional elements or content will be removed when the binding is applied. Binding handlers are executed in the order in which they appear on the element. Since the **foreach** binding changes the binding context, it is a best practice to always place the **foreach** binding after all other bindings on the element. This will ensure that preceding bindings are not affected by the binding context created by the **foreach** binding. To avoid performance issues, be careful to not create unintentional dependencies on observables inside of your **foreach.** Do not access an observable in the array using **$dyn.value** from within the child elements of the **foreach,** as the **foreach** binding has already subscribed to the observables in the array. Instead, use **$dyn.peek** to access an observable's value once without creating a subscription.

Arguments

Data (array list or JSON object)

The list of items to bind the child element to. If an array list is supplied, the binding context is an item in the array. If a JSON object array is supplied, the binding context is one of the object's properties.

Scope variables

When inside the scope of the **foreach,** the following scope variables are useful and can be used on the repeatable child element: $data, index, control, your own scope variables. The following example uses **foreach** to render a span element for each color in the array.

```
// In your control’s code-behind JS file
<script>
... // boilerplate code
self.colors = ['Red','Blue','Green'];
...
</script>

<!-- the markup in the HTML template -->
<div data-dyn-bind="foreach: $control.Colors">
<span data-dyn-bind="text: $data"></span>
</div>

<!-- the markup in the browser after the binding handler is applied -->
<div data-dyn-bind="foreach: $control.colors">
<span data-dyn-bind="text: $data">Red</span>
<span data-dyn-bind="text: $data">Blue</span>
<span data-dyn-bind="text: $data">Green</span>
</div>
```

The following examples shows a nested **foreach** binding. This example showcases how to use the index framework scope variable and custom scope variables to access the binding context from the parent element.

```
    // In your control’s code-behind JS file
<script>
... // boilerplate code
self.colors = [
    {
        Name: 'Red',
        Variants: ['Maroon','Burgundy','Sunrise']
    },
    {
        Name: 'Green',
        Variants: ['Sage','Forest','Lime']
    },
    {
        Name: 'Blue',
        Variants: ['Navy','Sky','Ice']
    }
];
...
</script>
<!-- the markup in the HTML template -->
<div data-dyn-bind="foreach: $control.colors">
<div data-dyn-bind="vars: {$BaseIndex: $index, $BaseColor: $data.Name}">
<div data-dyn-bind="foreach: $data.Variants">
<div data-dyn-bind="text: $BaseIndex+'.'+$index+' '+$data+' '+$BaseColor"></div>
</div>
</div>
</div>
<!-- the markup in the browser after the binding handler is applied -->
<div data-dyn-bind="foreach...">
<div data-dyn-bind="vars...">
<div data-dyn-bind="foreach...">
<div data-dyn-bind="vars...foreach...">
<div data-dyn-bind="text...">1.1 (0.2552) X++ Language</div>
<div data-dyn-bind="text...">1.2 (0.7) Applications</div>
</div>
</div>
<div data-dyn-bind="vars...">
<div data-dyn-bind="text...">2. (600) Technology</div>
<div data-dyn-bind="vars... foreach...">
<div data-dyn-bind="text...">2.1 (600.343) Microsoft Corporation</div>
<div data-dyn-bind="text...">2.2 (600.117) Enterprise Resource Planning</div>
</div>
</div>
</div>
```

if

Behavior

Conditionally renders and binds the child elements of the element with this binding. This binding handler only operates on the child elements. It will not show/hide the element with the binding, nor will this binding show/hide the text content of the element with the binding. Bindings on the child elements will only be executed if the condition evaluates to true. Once the bindings on child elements have been evaluated once they will remain data bound even if the condition changes to false. This means that any calculations caused by bindings on child elements will continue to operate even after the child elements are hidden. Consider this when evaluating the performance of your control.

Arguments

Condition (expression)

Determines whether to render the children elements. The following example conditionally binds the **show** and **text** elements.

```
    // In your control’s code-behind JS file
<script>
... // boilerplate code
self.show = $dyn.observable(false);
self.text = "Hello";
...
</script>
<!-- the markup in the HTML template -->
<div data-dyn-bind="if: $control.show">
<div data-dyn-bind="text: $control.text"></div>
</div>
<!-- the markup in the browser after the binding handler is applied -->
<div data-dyn-bind="if: $control.show">
</div>
// Later on, the value of the “show” observable changes to true
<script>
...
self.show(true);
...
</script>
<!-- the markup in the browser after the binding handler is re-applied due to the observable value changing
-->
<div data-dyn-bind="if: $control.show">
<div data-dyn-bind="text: $control.text">Hello</div>
</div>
```

sizing

Behavior

Specifies the height and width of the control. The sizing binding handler should always be applied to the root element of the template (the element that has the id attribute), and supplied the height and width values from the X++ instance of the control by using the $dyn.layout.sizing helper function. See Example 1.

Arguments

The arguments are passed an object containing height and width properties.

Height (int)

Determines the height in pixels of the element on which the binding handler is applied.

Width (int)

Determines the width in pixels of the element on which the binding handler is applied. The following example specifies the size of **MyControl**.

```
<!-- the markup in the HTML template -->
<!-- this boilerplate binding ensures that the control’s container is sized based on the height and width
properties -->
<div id="MyControl" data-dyn-bind="sizing: $dyn.layout.sizing($control)"></div>
<!-- the markup in browser after the binding handler is applied will vary based on the height and width
properties defined in $control -->
```

The following example makes the control large or small depending on the value of the **bigbox** variable.

```
// Later on, the value of the “show” observable changes to true
<script>
...
self.bigBox = true;
...
</script>
<!-- the markup in the HTML template -->
<div data-dyn-bind="sizing: {height: $control.bigBox?50:10, width: $control.bigBox?50:10}"></div>
<!-- the markup in the browser after the binding handler is applied -->
<div style="width: 50px; height: 50px;" data-dyn-bind="sizing: {height: $control.bigBox?50:10, width:
$control.bigBox?50:10}"></div>
```

text

Behavior

Binds to the textContent property of the element. The text binding handler is meant to be used with UI text. It is not meant to bind non-string values (such as numbers, dates or Booleans) to the element. Convert all values into strings before supplying them to the binding handler, by using the dyn.format function. The text binding handler will replace all of the content inside of the element with the binding, whether or not the existing content is HTML or simple text.

Arguments

Text (string)

The text to bind to. The following example binds the textContext property of the div element to the text property on the control.

```
// In your control’s code-behind JS file
<script>
... // boilerplate code
self.text = "Hello";
...
</script>
<!-- the markup in the HTML template -->
<div data-dyn-bind="text: $dyn.format('{0}',$control.text)"></div>
<!-- the markup in browser after the binding handler is applied -->
<div data-dyn-bind="text: $dyn.format('{0}',$control.text)">Hello</div>
```

vars

Behavior

Creates an HTML scope variable with the supplied name and value. The created scope variable is accessible only from bindings in the template. In addition, the scope variable is inherited by child elements. Binding handlers are executed in the order in which they appear on the element. Since the vars binding adds variables to the binding context, it is a best practice to always place the vars binding before all other bindings on the element. This will ensure that the subsequent bindings can access scope variables added by the vars binding. Do not create scope variables with any of the following names, as these names are reserved for framework scope variables: $control, $data, $index, and $value.

Arguments

Scope variables (object array)

The object array whose keys are the scope variable names and whose values are the initial values for the scope variables. The following example creates scope variables named "Hello" and "World" and displays their values.

```
<!-- the markup in the HTML template -->
<div data-dyn-bind="vars: {$myVar: 'Hello', $myObs: $dyn.observable('World')}">
<span data-dyn-bind="text: $dyn.format('{0} {1}!', $myVar, $myObs)">
</span>
</div>
<!-- the markup in browser after the binding handler is applied -->
<div data-dyn-bind="vars: {$myVar: 'Hello', $myObs: $dyn.observable('World')}">
<span data-dyn-bind="text: $dyn.format('{0} {1}!', $myVar, $myObs)">
Hello World!
</span>
</div>
```

Example 2

For an example, see the foreach binding handler examples.

visible

Behavior

Sets the visibility of the element. Always supply the visible binding handler on the root element of the template, and bind to the *Visible* property from the X++ control. This will ensure that the control respects the *Visible* property when it is set by a form developer or when it is set by the framework. If a control is initialized with its *Visible* X++ property set to false, then the control will not appear on the form, and it the control’s template will not be loaded in the browser. If the control’s *Visible* X++ property is set to true at a later time, then the control’s

template will be loaded and instantiated in the browser at that time. A control's *Visible* X++ property can be inherited from its parent controls on the form. An element's visibility may be controlled by its parent elements, controls and containers, regardless of whether the visible binding handler is applied. The cascading nature of visibility is a standard HTML behavior and is not specific to the control framework.

Arguments

Visible (boolean)

Determines whether the element is visible or not. The following example sets the visibility of the control's outermost div element.

```
// In your control's code-behind JS file
<script>
... // boilerplate code
self.Visible(false); // set the X++ observable property to false
...
</script>
<!-- the markup on the root element of the HTML template -->
<div id="MyControl" data-dyn-bind="visible: $control.Visible">Hello World!</div>
<!-- the markup in browser after the binding handler is applied -->
<div id="MyControl" style="display: none;" data-dyn-bind="visible: $control.Visible">Hello World!</div>
```

HTML: Scope variables

Scope variables can be used when binding values to binding handlers. Scope variables are only accessible from within the control's HTML template, and can only be used with the data binding attribute. Scope variables are neither accessible from other HTML attributes nor from the control's JavaScript class, but scope variables can be used in inline JavaScript expressions, functions and JSON objects that are passed to binding handlers.

$control

The *$control* scope variable provides the bindings in the HTML template with access to the properties and functions on the control's JavaScript instance. The following example binds visibility of the div element to the of Visible property of the control.

```
<div id="MyControl" data-dyn-bind="visible: $control.Visible"></div>
```

$data

The *$data* scope variable provides elements with access to their current binding context. Only variables defined in $data (the binding context) or scope variables, can be used inside of HTML bindings. Variables that do not exist in the current binding context and do not exist as current scope variable cannot be accessed from an HTML binding. In most cases the binding context will be the control's JavaScript instance, so *$data* and *$control* will be equivalent. However, in some cases the binding context can change. For example, for elements inside of a **foreach** binding, *$data* provides the elements with access to the current array item. In cases involving multiple nested **foreach** bindings, elements in a nested binding may need access to the array item in a parent **foreach** binding. To access items in the parent **foreach** binding, you may create a scope variable which will be accessible to elements in the nested **foreach** biding. For an example, see the foreach binding handler examples.

$index

The $index scope variable provides a 0-based index of the array item when in a **foreach** binding. For an example, see the foreach binding handler examples.

JavaScript: Inherited properties

Visible

The **Visible** property is inherited from the base JavaScript class (via **$dyn.ui.Controls.apply**). There is also a **Visible** property in X++ that the runtime class in inherits from the base **FormControl** X++ class. Simply bind this property to the visible binding handler and place it on the root element of the HTML template for your

control. The framework takes care of the rest. The following example shows how to use the Visible property.

```
<!-- the markup in the HTML template -->
<div id="MyControl" data-dyn-bind="visible: $control.Visible"></div>
```

Observable framework

$dyn.observe

Usage

Subscribes a function to changes of an observable. We recommend that you use dispose.

```
$dyn.observe(observable, observer, [context], [disposableObserver])
```

Arguments

Observable (observable)

Instance of an observable. Or a function, which will become a $dyn.computed.

Observer (function)

Function is invoked upon registration and also later when the observable is updated. Function is invoked with one argument, the value of the observable. If Observer returns false, then we un-subscribed automatically.

Context (options, optional)

Context to pass to the Observer. The Context becomes the ***this*** variable inside of the observer.

DisposableObserver (options, optional)

Unsubscribes the supplied DisposableObserver

Returns

Subscription (object)

Observable, ID, Dispose function (public) used to unsubscribe The following example subscribes to the myObs observable, and executes the supplied function whenever the myObs observable value changes.

```
$dyn.observe(myObs, function (value) { console.log(value);});
```

The following example shows how a function can automatically subscribe to observables simply by accessing the observable using $dyn.value. The first function is treated like an observable whose value is dependent upon the value of two other observables (FirstName and LastName). Every time one of the observables (FirstName or LastName) changes its value, then the first function has also changed its value. When this happens, the second function (the callback function) will log the concatenation of the observable values to the console.

```
self.FirstName = $dyn.observable("Joanne");
self.LastName = $dyn.observable("Gordon");
$dyn.observe(
    function () {
        // Joann + " " + Gordon
        return $dyn.value(self.FirstName) + " " + $dyn.value(self.LastName);
    },
    function (value) {
        // "Joanne Gordon"
        console.log(value);
    }
);
```

The following example performs similarly to the previous example. However, this example uses a computed observable, named myComp, to handle the concatenation.

```
self.FirstName = $dyn.observable("Joanne");
self.LastName = $dyn.observable("Gordon");
self.myComp = $dyn.computed(function () {
    // Joanne + " " + Gordon
    return $dyn.value(self.FirstName) + " " + $dyn.value(self.LastName);
});
$dyn.observe(
    self.myComp,
    function (value) {
        // "Joanne Gordon"
        console.log(value)
    );
},
{FirstNameLabel: label1, LastNameLabel: label2}
);
```

$dyn.observable

Usage

Creates an observable variable.

```
$dyn.observable([initial value])
```

Arguments

Initial value (optional)

The value to initialize the observable to.

Returns

Observable (function)

The newly created observable The following example creates and observable variable named "Hello".

```
var greeting = $dyn.observable("Hello");
```

$dyn.value

Usage

Accesses the value of an observable variable. When **$dyn.value** is called from inside of an observer function (such as an observer passed to **$dyn.observe** or **$dyn.computed**, as well as the binding expression passed to a binding handler) a dependency on the observable is created. This will cause the binding handler or callback to re-execute whenever the value of the observable changes. Because this dependency is created automatically when using **$dyn.value**, it is important to only use **$dyn.value** when you intentionally wish to create such a dependency. If you wish to access the value of an observable without creating a dependency, you should use $dyn.peek.

```
$dyn.value(observable)
```

Arguments

Observable

The observable property whose value to access.

Returns

Value

The current value in the observable property The following example returns the value of variable named observable and prints it to the console.

```
console.log($dyn.value(observable));
```

$dyn.peek

Usage

Accesses the value of an observable variable, without creating a dependency. For more information about dependency, see the $dyn.value function.

```
$dyn.peek(observable)
```

Arguments

Observable

The observable whose value to access.

Returns

Value

The current value in the observable The following example returns the value of variable named observable and prints it to the console.

```
console.log($dyn.peek(observable));
```

$dyn.computed

Usage

Wraps a function with an observability scope. If observables are accessed from inside of the function by using the **$dyn.value** function, then the function will re-execute whenever the values of those observables change. Observables that are accessed by using **$dyn.peek** will not cause the function to re-execute when their values change.

```
$dyn.computed(observer, [context], [disposableObserver])
```

Arguments

Observer (function)

Function is invoked upon registration and can also be invoked later due to an observable value change. The Observer automatically observes any observables that are accessed using $dyn.value from within the scope of the function.

Context (options, optional)

Context to pass to the Observer. The Context becomes the *this* variable inside of the observer.

DisposableObserver (options, optional)

Unsubscribes the supplied DisposableObserver

Returns

Anything (optional)

If the Observer returns a value, then that value will also be returned by the call to **$dyn.computed** on the first time **$dyn.computed** is called (upon registration) as well every time the observer is invoked.

Framework functions

$dyn.callFunction

Usage

Calls the apply method on specified function. Is cannot be used during an interaction.

Arguments

Function (function or observable)

The function to call. If an observable is supplied, the current value of the observable will be retrieved and used as the function.

This (object, optional)

The object to assign to *this* within the scope of the function.

Arguments (array, optional)

The arguments to pass to the supplied function.

Callback (function, optional)

The callback function to call when the supplied Function has returned. The callback will be passed any values that are returned by the function that is called. The following example calls the **apply** function on the **printName** function.

```
self.Name = "Joanne M Gordon";
var printName = function () {
    console.log(this.Name);
};
$dyn.callFunction(printName, self);
```

The following example calls the **getWholeName** function.

```
var getWholeName = function (first, middle, last) {
    var wholeName = first + " " + middle + " " + last;
    return wholeName;
};
var printName = function (wholeName) {
    console.log("Your name is: " + wholeName);
};
var firstName = "Joanne";
var middleName = "M";
var lastName = "Gordon";
$dyn.callFunction(getWholeName, null, [firstName , middleName, lastName], printName);
```

$dyn.format

Usage

Builds a string using the supplied values according to the supplied format.

Arguments

Format (string)

The format in which to build the string. Use bracket notation for placeholders.

Values (optional)

The comma separated values to use in the format

Returns

FormattedString (string)

The string after formatting has been applied The following example builds a string with the first, middle initial, and the last name.

Example 1

```
var first = "Joanne";
var middle = "M";
var last = "Gordon";
var $dyn.format("Your name is : {0} {1} {2}", first, middle, last);
```

$dyn.label

Usage

Provides access to any labels stored via the Globalization API.

Arguments

Identifier (string)

The label ID, as specified to the Globalization API.

Returns

Value (string)

The label string in the current culture, if the Identifier is found. Otherwise, returns the supplied Identifier as a string. The following example returns and prints the label named "greeting".

```
Globalize.addCultureInfo("en", {
    messages: {
        "greeting": "Hello!"
    },
});
console.log($dyn.label("greeting"));
```

CSS

Add namespaces to all CSS class names by prepending the class name with the control's template ID. This will prevent your control and its styles from conflicting with other controls in the client.

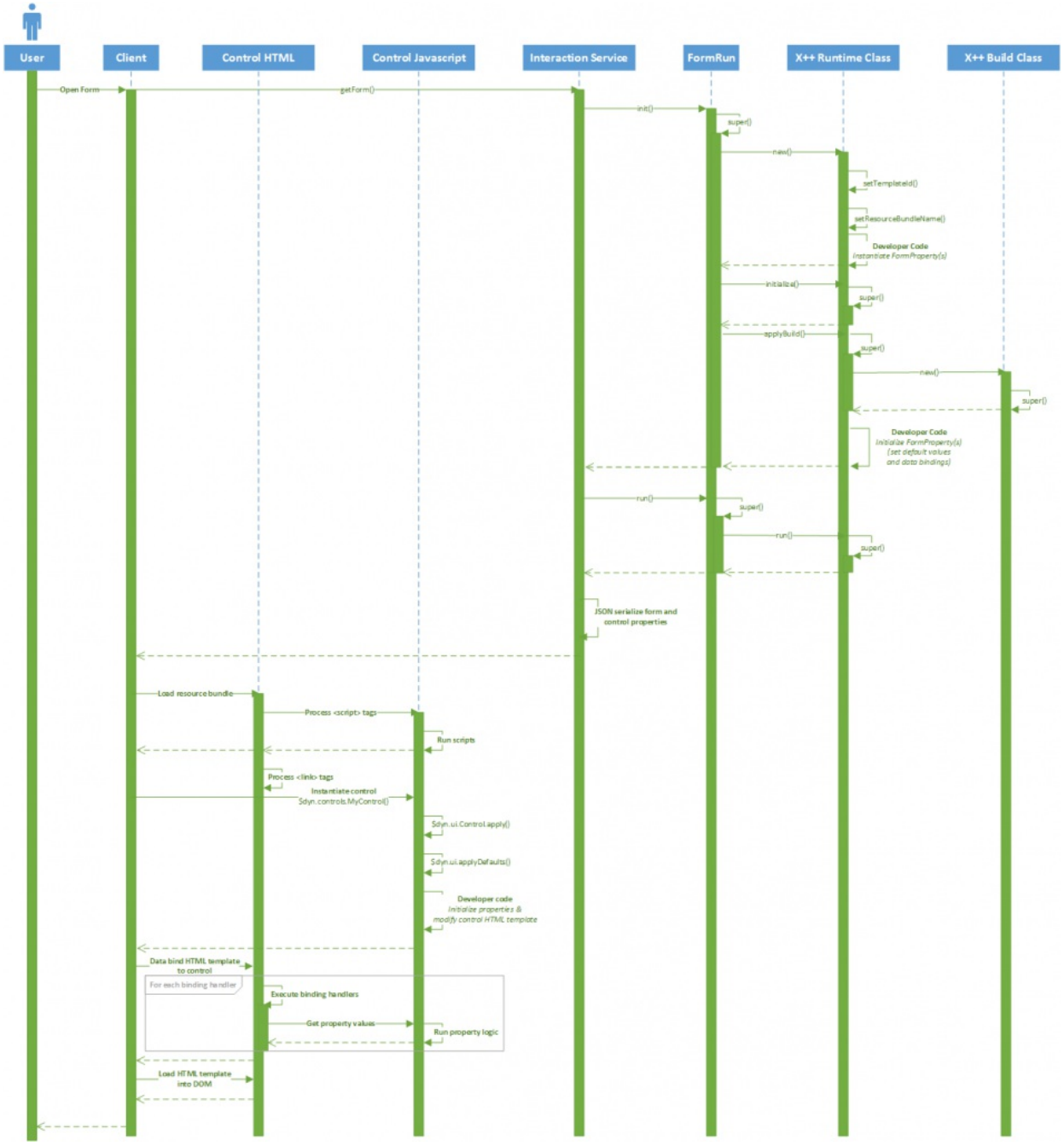
Flexbox

For advanced layout scenarios we encourage using Flexbox. Flexbox is compatible with the Extensible Control framework. Using CSS flexible boxes (Mozilla Developer Network) Please see the public Flexbox documentation for explanations and examples of the following topics:

- Responsive layouts
- Building columns and rows
- Arranging elements horizontally or vertically
- Arranging nesting elements
- Auto-sizing elements to stretch and shrink
- Locking/Freezing elements
- Building scrollable elements

Control Lifecycle Diagrams

Control Instantiation



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Control extensibility

2/18/2021 • 2 minutes to read • Edit Online

This article describes the architecture that lets developers extend the user interface and also define new user interface patterns.

You can extend the existing application user interface (UI) and can also define entirely new UI patterns to create compelling new user experiences. By using modern tools such as HTML5, CSS3, and jQuery, developers can define customized visualizations of business data and drastically enhance the program's interaction patterns.

Server-side architecture

The Control Extensibility Framework takes advantage of the existing and familiar X++ language for developing server-side data access and business logic. There are no artificial restrictions on the code that developers write to build extensible controls. Instead, developers can declaratively define the modeling experience and the run-time behavior through a set of X++ class and method attributes. A developer defines one class for the design-time behavior (the X++ Build Class) and one class for the run-time behavior (the X++ RunTime class).

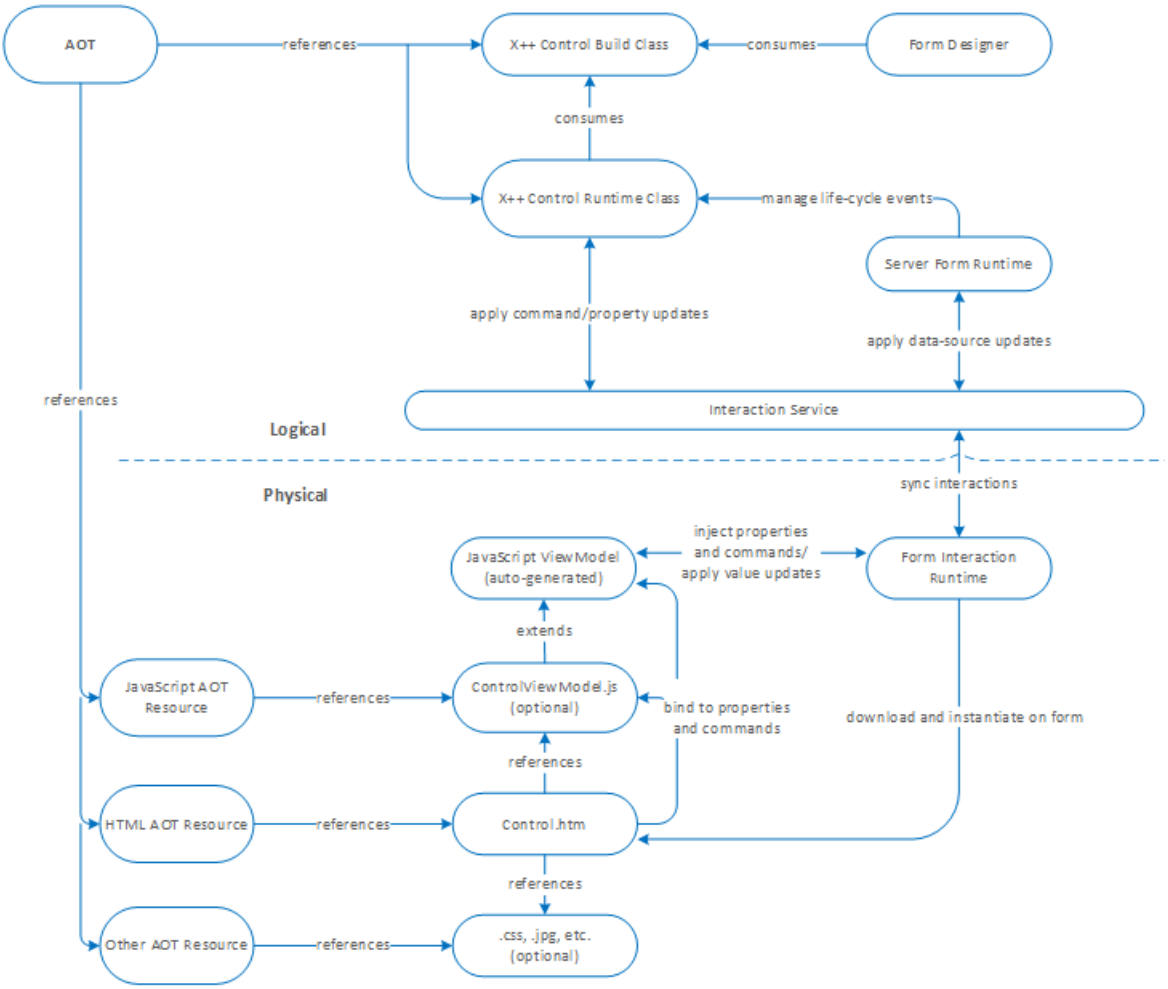
- In the X++ Build Class, attributes enable the definition of design-time behaviors such as custom properties in the property sheet, the addition of child controls, and extra modeling components.
- In the X++ Runtime Class, attributes are used to define the run-time properties and commands that the extensible control will access from the client. The X++ Runtime Class consumes the X++ Build Class to initialize the run-time properties, based on the values and data bindings that are specified in the property sheet.

Client-side architecture

The client-side behavior for the control is defined by using HTML and JavaScript. In the context of a Model-View-ViewModel architecture, the HTML for the control is the View, and the JavaScript is the viewmodel. The Control Extensibility Framework provides an HTML-based binding syntax that enables elements in the HTML View to be bound to data fields and properties in the JavaScript viewmodel. In addition, the framework enables visualization behavior to be defined based on conditional expressions or logical evaluations that can react to changes in viewmodel properties or business data. The JavaScript viewmodel is automatically generated at run time, based on the properties and commands that are defined in the X++ runtime for the control. This automatically generated viewmodel lets a developer define an HTML View that consumes the properties and commands that are defined in X++. If a developer wants additional client-side properties and commands, or wants to implement visualization behavior that can't be declaratively defined in the HTML View, the developer can extend the automatically generated viewmodel. The developer can take advantage of the JavaScript framework in conjunction with the powerful jQuery library.

Control extensibility architecture overview

The following diagram shows the artifacts that are involved and their relation to each other.



NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Create localizable labels

2/18/2021 • 5 minutes to read • Edit Online

This article explains how to create localizable labels for client components and HTML/JavaScript controls.

This article details the process for creating **localizable** labels for client components and HTML/JavaScript controls. This process uses the existing localization tools and process for labels to bring localization support to client components and HTML/JavaScript controls. The following process relies on the label resource controller that can serialize label files into their JavaScript equivalents so that the labels can be used by the client components and HTML/JavaScript controls. The label resource controller is deployed automatically. It is an MVC service that is located at the /Resources/Labels endpoint.

1. Create a label file

Use the developer tools to create a new label file for your control's area, or use an existing label file for your control's area. A control's area is determined by the owning team.

- For extensible controls, your goal should be to create one label file for each HTM resource file. If multiple HTM resources share the same set of labels, only one label file should be required for the set of HTM resource files.
- For client controls and components, in general, controls that share a lot of the same functionality (for example, the Input controls: StringEdit, ComboBox, CheckBox, and so on) should also share the same label file.

Don't use a label file that also contains labels that are only used in X++. The whole label file is serialized when it's loaded by the client, so be sure to keep the labels that aren't required by the client components/controls in a separate label file.

2. Add label strings to the label file

Use the developer tools to add label strings to the label file. **Example for extensible controls:**

- **Label file name:** ClockControl
- **Label ID:** Seconds
- **Label string:** seconds

3. Request the label file as a JavaScript file by using Resource manager

Use the script loading tag to load the JavaScript file. The loading tag should reference the label file from /Resources/Labels, because the label resource controller is located there. **Note:** For extensible controls, the controller automatically appends the label file name to the beginning of the label identifier in JavaScript.

```
<script src="/Resources/Labels/ClockControl.js"></script>
```

The JavaScript file that is returned will contain code that resembles the following example.

```
Globalize.addCultureInfo("en-us", {
    messages: {
        ClockControl_Seconds: "seconds"
    }
});
```

The culture information is injected automatically, based on the current user's culture. No action is required on the part of the control to set, modify, or read the user's culture. The preceding code adds each of your label strings to the jQuery Globalize label storage. You can then reference your labels throughout your HTML and JavaScript. The JavaScript code in the script file is run the moment that the file is loaded by the browser. Therefore, your labels are immediately ready for use. Be sure to add the label script loading tag **after** any other script loading tags in your HTML. The script loading tags are processed in order, from top to bottom. By loading the label script last, you help guarantee that no other scripts cause conflicts or override the labels that are set in the script label file.

4. Use localizable labels in HTML and JavaScript

The following framework application programming interface (API) can be used in HTML (inside **data-dyn-bind**) or in JavaScript to access the labels. **HTML**

```
<!-- Example of using a localizable label with a Label Control. Supply the label to the "Text" property on
the control -->
<div data-dyn-role="Label" data-dyn-bind="Text: $dyn.label('ClockControl_Seconds')"></div>
<!-- Example of using a localizable label with a basic HTML element. Supply the label to the "text" binding
handler for the element -->
<div data-dyn-bind="text: $dyn.label('ClockControl_Seconds')"></div>
```

JavaScript

```
/* Example of using a localizable label in JavaScript. */
var string mylabel = $dyn.label('ClockControl_Seconds');
```

You can also pass the label ID via a variable in HTML or JavaScript. **HTML**

```
<div data-dyn-bind="text: $dyn.label($data.SecondsLabel)"></div>
```

JavaScript

```
var string mylabel = $dyn.label(self.SecondsLabel);
```

The **$dyn.label** API will find the appropriately named label and return the string that can be used to display text to the user. This API will automatically select the label, based on the current user's culture.

Troubleshooting

If you have correctly created a label file, and the label has been deployed, you should be able to load the JavaScript version of the label file directly from the browser. You can access the JavaScript label file by navigating to the home page in the client and appending the following path to the URL: **/Resources/Labels/MyLabelFile.js**, where **MyLabelFile** is the name of the label file without the language suffix. For a deployed label file that is named MyLabelFile.en-us, follow these steps:

1. Navigate to the home page, and sign in. (On one-box deployments, the URL of the home page is `https://usncax1aos.cloud.onebox.dynamics.com/en/`.)

2. Make sure that the desired language has been set by going to **Options** > **Language and Region**. (You don't have to change the language if it's already set to the language that you want.) Now that the user's language has been set in the current session, the label resource controller will know what language to use when the label file is loaded.
3. To load the JavaScript version of the label file, navigate to the label file by adding **Resources/Labels/MyLabelFile.js** to the URL. (On one-box deployments, the whole URL is `https://usncax1aos.cloud.onebox.dynamics.com/en/Resources/Labels/MyLabelFile.js`.)
4. The corresponding label file will be JSON-serialized, and the browser will either show the text on the current tab or prompt you to download the .js file. If you download the file, you can then open it locally to inspect it.

If the browser doesn't find the file, you might have mistyped the name of the label, or you might not have deployed the label correctly. There is never a physical .js file for the label in the web folder /Resources/Labels. The .js file is dynamically generated by the label resource controller.

Microsoft Visual Studio form previewer

The form previewer isn't currently configured to load labels via the label resource controller. The form previewer will load only labels that are defined directly in the .js file for the code behind (located at /Resources/Scripts). Until the form previewer is updated so that it can load .js files from the label resource controller, make sure that you also define the labels in the .js file for the code behind of the control. The dependency on these labels will be removed in a future update.

Extensible control layout guidelines

2/18/2021 • 2 minutes to read • Edit Online

This article provides guidelines that you should follow when you specify the layout and sizing of extensible controls.

Dos and don'ts for achieving the desired layout

- Don't use the layout classes on your control directly. (For example, **layout-container** and **layout-horizontal** are classes that you might see on controls in the DOM.) Instead, use the layout binding handlers to apply these classes. Internet Explorer uses a different layout framework, and to add some inline styles to elements, this framework requires the extra binding information that the handlers provide. Therefore, make sure that the classes are **not** hard-coded into the controls.
- Don't use absolute positioning (**position: absolute** and **top/bottom/left/right** positions) for elements that are children of a container that uses the layout binding handler. Absolute positioning of these elements prevents the CSS classes that are applied from laying things out correctly.
- Be careful about using **width: 100%** and **height: 100%**. These settings might not always work when they are combined with our layout CSS classes. If you want filling behavior, it's a good idea to use the **$dyn.layout.Size.available** option of the sizing binding handler instead.

Layout binding handlers

Layout

Used to apply **ArrangeMethod** and **Columns** attributes to containers Options: **arrangeMethod**, **Columns**, **Children**

ArrangeMethod

- $dyn.layout.ArrangeMethod.vertical
- $dyn.layout.ArrangeMethod.horizontalLeft
- $dyn.layout.ArrangeMethod.horizontalWrap
- $dyn.layout.ArrangeMethod.horizontalRight
- $dyn.layout.ArrangeMethod.none (No layout CSS classes should be applied to the element.)

Columns

- **$dyn.layout.Columns.fill** – Use this constant. Depending on the control, either "fill" and "balanced" columns are used.
- **$dyn.layout.Columns.fixed** – Use a fixed value, such as **1**, **2**, or **3**.

Children

- Use **$data.Children** if the content handlers (append, replace) will be used to append children through this container. It should be used only if this control is a container.

Example usage:

```
<div data-dyn-bind="layout: { arrangeMethod: $dyn.layout.ArrangeMethod.vertical, columns: '1' }"> </div>
```

Sizing

Height/width

- **$dyn.layout.Size.available (SizeToAvailable)** – Fill the available space in that direction.

- **$dyn.layout.Size.content (SizeToContent)** – Size to the contents in that direction.
- **$dyn.layout.Size.fixed (Manual)** – A fixed pixel value.

Example usage:

```
<div data-dyn-bind="sizing: { height: $dyn.layout.Size.available, width: $dyn.layout.Size.content }"> </div>
```

Things to note about the sizing binding handler:

- When you use the sizing binding handler in combination with the **SizeToAvailable** option (**$dyn.layout.Size.available**), the layout binding handler must be used on the parent ** **/DOM element. This is how we know which axis to fill along (horizontal or vertical).

Should you use layout/sizing handlers or set the properties directly?

- If your extensible control inherits from a client control (such as Group or PivotItem), you won't use the binding handlers directly on the ** ** that is associated with that control, because the binding is already there. However, you might have to set the properties as you want them directly in the **data-dyn-bind** attribute.

Example:

```
<div data-dyn-role="Group" data-dyn-bind="ArrangeMethod: $dyn.layout.ArrangeMethod.vertical, Columns:
$dyn.layout.Columns.fill, Height: $dyn.layout.Size.available"></div>
```

FAQ

Is my control being laid out as expected?

A good way to tell is to inspect the element and look for the desired classes.

- Layout classes to look for, depending on the options applied:
  - layout-container
  - layout-vertical
  - layout-horizontal
- Sizing classes to look for:
  - For **$dyn.layout.Size.available**, you should see either **fill-width** or **fill-height**.
  - For manual size, you should see either **fixed-height** or **fixed-width**.
  - For **$dyn.layout.Size.content**, there should be no extra classes, and the manual height/width should be specified inline on the element.

If these classes don't appear as you expected, examine the usage of your binding handlers, and make sure that you've read through the list of dos and don'ts on this page.

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).

Control the text that Task Recorder generates for a control

2/18/2021 • 4 minutes to read • Edit Online

This article describes how Task recorder determines what instruction label to generate for controls. It then explains how you can make sure that these labels are meaningful for the user.

Every control must have useful and meaningful instruction labels, so that the task guide, Microsoft Word document, and Help content meet Content Publishing standards for readability. We must first define two terms:

- **Control label** – The value that comes from the label property on the control.
- **Instruction label** – The label that a control instructs Task recorder to use when it's describing how to use that control (for example, "Click OK" or "In the First name field, enter 'John'").

When a control logs an event to Task recorder, three methods can be used to determine the instruction label that is shown to the user:

- As part of logging the Task recorder event, the control might specify an exact instruction label ID to use. As a best practice, here is how label IDs should be named: [client control type name]_[property or command name] For manual specification of an instruction label ID, see the code example later in this article (**OptionalInstructionLabelIDOverride**).
- If the control doesn't explicitly specify an instruction label ID, Task recorder looks in the SysTaskRecorderLabel file to try to find an existing instruction label ID that fits the following naming syntax: [client control type name]_[property or command name] If an instruction label ID of this type is found, Task recorder uses it.
- If a label can't be determined by using the preceding methods, Task recorder falls back to a more general-purpose instruction label. The general purpose instruction labels are in the SysTaskRecorder label file. There is one general-purpose instruction label for commands and another for properties.
  - Here is the general purpose instruction label for commands:
    - **Label ID:** CommandUserAction
    - **Label string:** %2 the %1
  - Here is the general purpose instruction label for properties:
    - **Label ID:** PropertySetValue
    - **Label string:** In the %1 field, enter %2

When Task recorder has determined which instruction label to use (via one of the preceding three methods), it strFormats the label by using the following arguments:

- **%1** – This argument is the control label. Task recorder gets the control label by inspecting the label on the intractable. However, a control can override this label and provide its own label. See the code example later in this article (**OptionalControlLabelOverride**).
- **%2** – This argument is either the value (for properties) or the command name (for commands). This value will be the value that the control sent to Task recorder as part of logging the event. However, the raw data value can be ugly or meaningless to an end user. Therefore, a control can also provide a more user-friendly version of the value that Task recorder can display instead of the raw data value. See the code example later in this article (**OptionalValueLabelOverride**).
- **%3–%5** – These are command arguments and are used rarely. However, grids use them to record the row number, for example.

Case study

Let's use the Checkbox control as a case study for improvement. Currently, an instruction label isn't specified for the Checkbox via either method 1 or method 2 (see the previous section of this article). Therefore, the general-purpose property instruction label is used instead. If someone records selecting the Checkbox for a field that is named **Show infolog on failure**, the Task recorder output currently looks like this:

> In the Show infolog on failure field, enter True.

However, typical end users might not know what it means to set a Checkbox to **True**. Therefore, a suggested improvement is for the Checkbox to produce a label that looks like this:

> Check Show infolog on failure.

To make this improvement, someone must add a new label ID. That user must then use the label ID when the event is logged to Task recorder by using method 1:

- **Label ID:** Checkbox_Value
- **Label string:** "%2 %1."

This change produces output that looks like this:

> True Show infolog on failure.

This isn't quite what we want to see. Therefore, in addition, when the Checkbox logs the property change event to Task recorder, it should pass in a specific *value label* that says either "Check" or "Uncheck." If the control explicitly specifies a value label, Task recorder will use that value label instead of the raw data value that was recorded (**True** or **False**, in this case). See the code example later in this article (**OptionalValueLabelOverride**). After the user creates the new label and specifies the value label when the event is logged to Task recorder, the control will have suitable text output:

> Check Show infolog on failure.

Finally, the Checkbox should have a second instruction label for "example value" usage of the Checkbox. "Example value" represents a special way that Task recorder displays an instruction label to the user. The author of the task recording can specify whether the task guide should instruct users to enter the same values that were entered when the guide was recorded, or whether the task guide should instruct users to enter their own values that are specific to their business requirements. Example value instruction labels have the following label ID syntax: [client control type name]_[property name]_Example For the Checkbox example, the value label would look like this:

- **Label ID:** Checkbox_Value_Example
- **Label string:** "Check or uncheck the %1 field."

The following code example shows how property change events are logged to Task recorder by an X++ control. Similar application programming interfaces (APIs) exist for C++ kernel controls. Command events have a similar API.

```
[FormPropertyAttribute(FormPropertyKind::Value, #MyPropertyName)]
    public str value(str_value = valueProperty.parmValue())
    {
        if(!prmIsDefault(_value))
        {
            using (var scope = SysTaskRecorder::addPropertyUserAction(#MyPropertyName, this, _value,
[OptionalInstructionLabelIDOverride], [OptionalValueLabelOverride], [OptionalControlLabelOverride]))
            {
                // Property set logic goes here
                valueProperty.setValueOrBinding(_value);
            }
        }
    }
```

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Build operational workspaces

2/18/2021 • 6 minutes to read • Edit Online

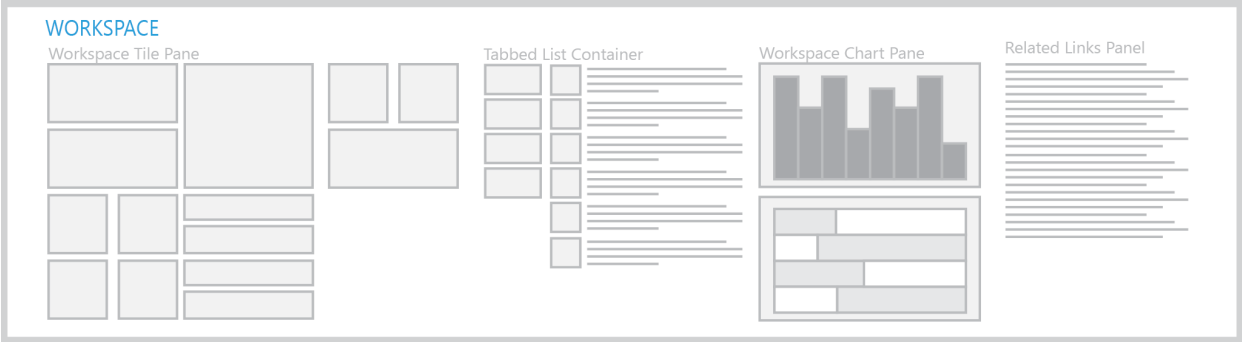
This topic provides detailed information about workspaces and the patterns and subpatterns that are used to build operational workspaces.

A workspace is defined as...

- Part of the primary navigation mechanism.
- A form that supports a business activity (a logical group of tasks that make up the work of a target persona).
- A way to provide an initial overview and to increase productivity in the activity by allowing simple tasks to be completed directly in the workspace.

Workspaces have the following goals...

- Enable the user to understand the current state of the activity to support informed decisions.
- Let users navigate to deeper pages by selecting data, which avoids round-trips to pages with no information.
- Let users perform light tasks in the workspaces to avoid round-trips to deeper pages.
- Complete an activity without leaving the workspace.
- Reduce the need for navigation.
- Provide visual impact.
- Be constructed using prescriptive patterns and best practices that lead to minimal COGS and fast response times.



To accomplish these goals, the operation workspace pattern was developed.

Examples

An example of a workspace is the **Reservation management** workspace in **Fleet management**. You can get to it by accessing the menu item **FMClerkWorkspace**. The workspace, shown above, has the following items:

- **Summary** - Contains tiles and a chart.
- **Rentals** - Contains a vertical tab control having three pages - the first is selected, and you can see the corresponding content on the rightmost side.
- **Statistics** - Contains stacked charts.
- **Related links** - Contains a series of grouped menu item links to forms of relevance to this user and activity.

The overall form, as well as each of the sections within, are defined using UX patterns and subpatterns. The corresponding patterns are described in detail in the following sections.

Patterns and subpatterns

When building an operational workspace, you must use the patterns and subpatterns defined for that purpose. These patterns and subpatterns are described below. In general, when a control is cited within a pattern's structure, it will be described as: Common name (ControlType) [cardinality] If cardinality isn't specified, then the item is required exactly once. Simple patterns and subpatterns have the structural tree omitted.

Patterns

There are the top-level patterns for use with operational workspaces.

Workspace Operational

This pattern is the primary operational workspace pattern and should be applied to the **Design** node of the operational workspace form. It will prescribe the following structure:

- Design
  - ActionPane (ActionPane) [0..*]
  - Workspace Page Filter Group (Group) [0..1]
    - Subpattern: Workspace Page Filter Group
  - Panorama (Tab)
    - Section Summary Tiles (TabPage)
      - Subpattern: Section tiles
    - Section Tabbed List (TabPage)
      - Subpattern: Section Tabbed List
    - Section Charts (TabPage) [0..1]
      - Subpattern: Section Stacked Chart
    - Section-Related Links (TabPage)
      - Subpattern: Section-Related Links

Form Part Section List

This pattern is used for **Form Part** forms containing a list. These lists are referenced within the Section Tabbed List TabPage in the Workspace Operational pattern.

- Design
  - Header Group (Group) [0..1]
    - Subpattern: Filters and toolbar - Inline OR Subpattern: Filters and Toolbar - Stacked
  - Content Grid (Grid)
  - Default Action Button ($Button - any type of button) [0..1]
  - See All Menu Item (MenuFunctionButton) [0..1]

Hub Part Chart

This pattern is used for **Form Part** forms containing a chart control. These chart forms are referenced within two subpatterns: Section Tiles and Section Stacked Chart. It requires exactly one Chart control.

Subpatterns

Workspace Page Filter Group

This subpattern is referenced in the pattern Workspace Operational, in Workspace Page Filter Group. It allows for a single input control, which can be used to filter the workspace as a whole.

Section Tiles

This subpattern is referenced in the pattern Workspace Operational, in Section Summary tiles. It allows for both tiles and charts. Any number of tiles and charts can be defined, in any order. Tiles are defined with TileButton controls, and charts are defined with Form Part controls. A chart Form Part must have dimensions that match those of a normal tile, to ensure the chart flows correctly with the tiles displayed.

Section Tabbed List

This subpattern is referenced in the pattern Workspace Operational, in Section Tabbed List. It allows for multiple list containers to be modeled, each of which ultimately references a Form Part that points to a form containing the desired list. It requires the following structure:

- Tabbed List (Tab)
  - Tabbed List Page (TabPage) [0..*]
    - Form Part Section List (FormPartControl). **Note**: The referenced form must use the form pattern, Form Part Section List.

Section Stacked Chart

This subpattern is referenced in the pattern Workspace Operational, in Section Stacked Chart. It allows for up to two charts, which will be rendered in a vertically stacked orientation. It requires the following structure:

- Tab page (TabPage)
  - First Chart FormPart (FormPartControl) [0..1]
  - Second Chart FormPart (FormPartControl) [0..1]

Section-Related Links

This subpattern is referenced in the pattern Workspace Operational, in Section-Related Links. It allows for a series of links, with one level of nesting permitted. It requires the following structure:

- Tab page (TabPage)
  - Menu Function Button (MenuFunctionButton) [0..*]
  - Links Group (Group) [0..*]
    - Menu Function Button (MenuFunctionButton) [1..*]

Filters and Toolbar – Inline

This subpattern is referenced in the pattern Form Part Section List, in Header Group. It allows for some filters and a toolbar, all on the same line. It requires the following structure:

- Group (Group)
  - Filter Group (Group) [0..1]
    - Quick Filter (QuickFilterControl) [0..1]
    - Custom Filter Fields ($Field – any type of field) [0..*]
  - Toolbar (ActionPane) [0..1]

Filters and Toolbar - Stacked

This subpattern is referenced in the pattern Form Part Section List, in the Header Group. It allows for some filter fields on one line, and a toolbar on a line below those filters. It requires the following structure:

- Group (Group)
  - Filter Group (Group) [0..1]
    - Quick Filter (QuickFilterControl) [0..1]
    - Filter Field 1 ($Field - any type of field) [0..1]
    - Filter Field 2 ($Field - any type of field) [0..1]
  - Toolbar (ActionPane) [0..1]

Future best practices check

There are a few best practice (BP) checks that will eventually be built for workspaces. These checks are intended to provide guidance to the user about items that are recommended for performance or design reasons.

Filters are covered by indexes

This check is intended to ensure that any field that is modeled for use as a field on a workspace-wide filter is covered by an index. This check will help ensure that performance remains superior when users are taking

advantage of these filters.

Chart parts only contain OLAP charts

When a workspace contains a chart, that chart is modeled as a separate form, and that form is referenced on the workspace via a Form Part control. The intent of this check is to ensure that any such charts ultimately referenced from a workspace are only using OLAP data.

Count tiles have queries defined

A tile caching system has been implemented to improve performance of workspaces, as these forms generally contain several count tiles. For these count tiles to work correctly with the caching system, each tile must have a query defined. That query may be defined on the tile or on the menu item referenced by the tile. The intent of this BP check is to ensure a query is defined in one of these two locations for all count tiles.

Tile and list caching for workspaces

2/18/2021 • 10 minutes to read • Edit Online

It's important that workspaces perform well, and that they be responsive (that is, the data that appears in a workspace is refreshed as expected and kept up to date). This topic discusses framework support for caching data that is used for tiles and lists.

Introduction

Workspaces are intended to be the hub of activity for most users. They display a wealth of information that is collected from various sources. Therefore, you must make sure that workspaces perform well and are responsive (that is, the data shown in the workspace is refreshed as expected and/or kept up to date). Caching can help guarantee great workspace performance when data comes from poorly performing queries, and is especially useful when multiple users require access to the same set of data at the same time. For example, if a grid on a form uses a complex (that is, low-performing) query, you can cache the results so that they are available for subsequent loads of the form. This article discusses framework support for caching data that is used for tiles and lists. In terms of a responsive workspace, when users navigate to a workspace (or return to it), they expect that the data in the workspace will be relatively up to date. For example, if a user takes an action that changes the data for a list or tile in a workspace, the workspace should reflect that change immediately after the action is performed (if the action was taken directly from the workspace) or when the user returns to the workspace form (if the action was taken on a different form). This article describes techniques for making sure that this type of data refresh occurs (both metadata and code) for both tiles and lists.

Count tiles

Automatic data caching

The framework automatically sets up data caching behind any defined count tile. Therefore, no extra code is required in this case. A **Refresh Frequency** metadata property on Tiles determines how often the count on the tile is automatically updated. The following table describes the options and guidance for this property.

VALUE	WHEN TO USE
As Fast As Permissible (5 seconds)	Query execution time is 25 ms or less, and there is demand to see the updated value all the time.
10 Minutes	Query execution time is less than 250 ms, and updated data must be seen periodically
24 Hours	Query execution time is less than 2000 ms, and/or the count isn't expected to change often or up-to-date counts are vital.

Refresh management

Count tiles that show values that represent pending work should be fairly responsive. Ideally, the fastest possible refresh frequency should be defined for these tiles, so that the count that appears on a workspace tile is always within five seconds of being up to date. Because this quick refresh rate should be set only on performant queries (queries that have an execution time of less than 25 ms), the first recommendation is that you work on query performance for count tiles, to try to get query execution under the 25-ms threshold. In general, achieving this execution speed requires two things:

- A selective WHERE condition on the query Preferably, the conditions on the query should reduce the result

set to fewer than ~500 records, and ideally to fewer than ~100 rows.
- An index structure that can act on the selective WHERE condition

See the "Common mistakes and tips for query optimization" section for details about how to evaluate and improve query performance to achieve these execution speeds. For tiles that have backing queries and can't meet the 25-ms execution speed threshold, the refresh frequency on the tile should be set to one of the lower values (for example, 10 minutes or 24 hours). If the values must be updated more frequently for tiles that have less-efficient queries (which should be rare), you can add the following code to manually refresh the cache when an action is taken that will affect the cached set.

```
TileDataService::forceRefresh(tilestr(<tileName>), formRun)
```

An example of a data set that might not change often is products that have no configuration. A tile that shows this count might have a refresh frequency of 10 minutes. However, the tile count might still appear responsive if the products form is instrumented to force-refresh the data cache when a configuration is defined for a product that previously had no configuration.

Workspace lists

Although the framework automatically sets up data caches for count tiles, manual setup is required for lists that must use data caching. Here are the high-level steps for introducing cached data for a list:

1. Create a query that references all the columns that you want in the cached data set.
2. Create a table that contains all the fields that you want to cache.
3. Create a class that defines a mapping between the cache query and cache table.
4. Add/reference the cached table as a data source on your form.

Each of these steps is described in detail in the following sections and is paired with an example from the Fleet workspace (Reservation Management).

Cache query

First, you must create a query that will be used to populate the cache table. This query should have the following characteristics:

- It should be against the tables that you want to get your cache data from.
- It should limit the results to the results that you're actually interested in.
- It should select only the fields that you want to cache. **Note:** Each data source should have **DynamicFields**=**No** to avoid extraneous fields.

Cache table

Next, you must define a table that contains a set of fields that match the fields from the cache query. In addition to those fields, you must also add a field that is named **SysDataCacheContextId** (Int64). This field is used to map the cache row to the base cache tables. You'll define a mapping on the table, between the SysDataSetCacheTableMap table's **Id** and **SysDataCacheContextId** fields and the cache table's **RecId** and **SysDataCacheContextId** fields, respectively. You can also define relations between this table and others, in addition to data methods that use the cached fields.

Cache class

The third step is to create a class that defines the relationship between the cache query and the cache table. This class requires that a few attributes be defined, and it must also extend and implement the appropriate framework data caching classes. The following code shows the corresponding class from the Reservation Management workspace.

```
[SysDataSetExtension(classStr(FMPickupAndReturn)), // The name of this class
SysDataSetCacheTableExtension(tableStr(FMPickupAndReturnCache))] // The name of the cache table
class FMPickupAndReturn extends SysDataSetQuery implements SysIDataSet
{
    public SysDataCacheRefreshFrequency parmRefreshFrequency()
    {
        return 600; // Cache refresh frequency, in seconds.
    }
    public SysQueryableIdentifier parmQueryableIdentifier()
    {
        return queryStr(FMPickupAndReturnQuery); // The name of the query.
    }
    public SysDataCacheTypeId parmCacheTypeId()
    {
        return tableNum(FMPickupAndReturnCache); // The name of the table.
    }
    public static FMPickupAndReturn construct()
    {
        return new FMPickupAndReturn();
    }
}
```

In some circumstances, you might also have to implement the **parmQueryableToCacheMapping()** method. This method is required when at least one column name in your cache table doesn't match the name of the corresponding column in the backing table (for example, if you must add two fields that have the same name but are from different tables). In this case, you can implement this method to define the column mapping between the cache table and the backing tables. The syntax is the same as the syntax for the **Query::Insert_RecordSet()** method (https://msdn.microsoft.com/library/query.insert_recordset.aspx).

```
public Map parmQueryableToCacheMapping()
{
    Map sourceToTargetMap = super();
    return sourceToTargetMap;
}
```

Form implementation

After you've built your cache query, table, and class, you're ready to use the cache on your form. Add the cache table to your form as a data source, and reference it just as you would reference any other data source. In order for the form to correctly take advantage of caching, some code is required.

1. In a **registerDatasourceOnQueryingEvent()** method, add an event handler to the data source's **OnQueryExecuting** event that calls **prepareDataSet**,. This requires that the form class implement **SysIDataSetConsumerForm**.
2. If you want the form to use filtering, as provided by a workspace-wide filter, in a **registerDatasourceOnQueryingEvent()** method, register **applyFilter** on the **OnQueryExecuting** event. This requires that the form class implement **SysIFilterConsumerForm**.
3. If the form must react when a parent form's filters change (for example, a workspace-wide filter), implement **SysIFilterEventHandler** on the form class, and include an **onFilterChanged()** method that calls **executeQuery()** on the cache data source.

The following code is an example from the **FMPickingUpTodayPart** form, which is one of the tabbed lists on the Reservation Management workspace.

```
[Form]public class FMPickingUpTodayPart extends FormRun
implements SysIFilterConsumerForm, SysIDataSetConsumerForm, SysIFilterEventHandler
{
    public void registerDatasourceOnQueryingEvent()
    {
        FMPickupAndReturnCache_DS.OnQueryExecuting +=
eventhandler(this.parmDataSetFormQueryEventHandler().prepareDataSet);
        FMPickupAndReturnCache_DS.OnQueryExecuting
+=eventhandler(this.parmFilterFormQueryEventHandler().applyFilter);
    }
    public void onFilterChanged()
    {
        FMPickupAndReturnCache_DS.executeQuery();
    }
}
```

Additional examples

In addition to the previous examples, you can refer to **SysFoundationTestOpenHeaderDataset**, which is used on **SysFoundationTestOpenHeadersFormPart** (which is itself referenced in **SysFoundationTestWorkspace**). This form uses the same method that is described earlier, but also includes some caching of a query aggregate count.

Refresh management

Lists of data in workspaces should also be responsive to user actions, especially if those actions cause records to no longer meet the criteria for appearing in the list. For example, you have a list of car rentals that are scheduled to start today. Every time that an employee initiates a rental record from that list with a customer, that record should no longer appear in the workspace list. Two mechanisms are available for keeping this list up to date:

- If the user takes an action that removes the record from the list, code can be added to guarantee that the list is immediately updated by refreshing the data source. If the data source is a cache data source, and the refresh frequency is slow, you might want to delete the corresponding records from the cache data source before you call **refresh**, to avoid having to force-refresh the cache.
- If the first option doesn't meet your requirements, you can asynchronously refresh form parts at a time interval. This option can be set up by using the **AutoRefreshInterval** property on the FormPart control.
    - For workspaces that have user interaction, the cache refresh frequency should be set based on query performance. In this case, the current recommendation is to not set the **AutoRefreshInterval**. Instead, rely on the user to manually refresh the workspace to bring in more data. Alternatively, you consider using an infrequent auto-refresh. However, in this case, the user's current selection will not be retained if the list is updated while the user is interacting with it.
    - A small percentage of workspaces are intended for viewing purposes only (no user interaction). These workspaces should use set the **AutoRefreshInterval** property programmatically to match the current refresh frequency of the backing cache. (Use **SysIDataCacheConfiguration.parmRefreshFrequency()** to retrieve the current refresh frequency of a cache, because it can be modified by the system administrator at run time.)
    - FormParts that have Charts should set the **AutoRefreshInterval** property to periodically show updated chart data.

Common mistakes and tips for query optimization

Here a few general guidelines to consider when you optimize queries. This isn't an extensive guide to query optimization but provides just a few simple guiding principles.

- **Trace the statement by using SQL Profiler.** Attach the Microsoft SQL Server Profiler to the database, and capture the SQL statement that you want to optimize. You can use the tuning template that is provided in a default installation. Don't forget to disable tracing after you've obtained the statement that you're interested

in. For more information, see https://msdn.microsoft.com/library/ms175047.aspx.

- **Always look at the query plan.** In Microsoft SQL Server Management Studio, make sure that you've enabled **Include actual Execution Plan**. Look at the query plan, and watch out for any warnings. The thickness of the arrows indicate how many rows have been fetched and brought to the next step.
- **Compare CPU milliseconds and logical I/O.** One good way to determine whether a change to a given SQL statement improved the statement is to look at the logical I/O and CPU milliseconds. To obtain these numbers, use the following statements in the query editor:
  - set statistics time on
  - set statistics io on
- **Always clear caches when you measure a statement.** To make sure that Microsoft SQL Server isn't using a cached execution plan, it's advisable that you flush the cache when you rerun a statement. To flush the cache, run the following two commands:
  - dbcc dropCleanBuffers
  - dbcc freeProcCache
- **Here are some common patterns to watch out for:**
  - **Missing index:** Analyze your WHERE conditions, and make sure that a selective index exists.
  - **Processing the same table/row many times:** Especially when you do joins, try not to repeat yourself. Minimize the number of times that the same table/row must be processed. If you have tables that must be part of the result set but aren't used to narrow down the result set, move them as far out as possible (especially in union queries).
- **Test on volume data.** To identify issues in your query, always run it on volume data.

Task recorder resources

2/18/2021 • 33 minutes to read • Edit Online

This topic describes how to use Task recorder to record business processes.

Overview

Task recorder

Task recorder for Finance and Operations apps is a utility that lets users record business processes for several different use cases. Here are some examples:

- Step-by-step guided tours of a specific business process in the application itself
- Documentation of a business process as a Microsoft Word document that can optionally include screenshots
- Regression tests for a business process
- Automatic playback of a business process in the application

Task recorder for Finance and Operations apps boasts high responsiveness, a flexible extensibility application programming interface (API), and seamless integration with consumers of business process recordings. Task recorder is also integrated with the Business process modeler (BPM) tool in Microsoft Dynamics Lifecycle Services (LCS), so that users can continue to organize their recordings. However, users can no longer produce business process diagrams from recordings.

Task recorder can automatically generate application regression tests from business process recordings and play back previously recorded processes. These features also include test-specific gestures that let users take full advantage of Task recorder.

Architecture

Task recorder can record user actions in the client with exact fidelity, because every control is instrumented to notify Task recorder about the execution of user actions. The control notifies Task recorder that an event has occurred and passes all the relevant information about the corresponding user action in real time. From this information, Task recorder can capture the type of user action (for example, a button click, value entry, or navigation) and any data that is related to the user action (for example, the input data value and type, form context, or record context). Task recorder persists the information with enough detail to ensure that a playback of the recording can run the recorded actions exactly as they were performed by the user.

Basic configuration

Task recorder is included with every Finance and Operations app, and lets users begin to record business processes immediately after they open the client for the first time.

> **IMPORTANT**
>
> The **Task guides** tab is currently not available in Commerce or Human Resources. We are currently working to enable this functionality in a future release. Task guides in the Getting Started experience in Human Resources remain available to cover basic functionality. Procedural help is also available on the docs.microsoft.com site (https://docs.microsoft.com/dynamics365/) for both Commerce and Human Resources.

Start a new recording

The following steps show how to use Task recorder to start a new recording.

1. Open the product, and sign in. It's a good practice to refresh the browser before each new recording. A refresh creates a new user session and restarts Task recorder. Therefore, it provides the most stable recording experience.

2. Select the company that you want to use while recording. If this is your first time using Task recorder, you can follow along as this tutorial creates a sample recording based on a Fleet Management business process. You will need to load the Fleet demo data to follow along:

   a. Go to **Dashboard > Fleet Management > Fleet setup**.
   b. Click **Load demo data**.
   c. When the data is finished loading, click **Close**.
   d. Go back to the **Dashboard** by clicking the product name in the navigation bar.

3. Go to **Settings > Task recorder**.

4. The **Task recorder** pane is opened. You can click the **Close** button (**X**) in the upper-right corner to close the **Task recorder** pane before you begin a new recording. You can reopen the pane by following the previous steps.

5. Click **Create recording**.

6. Enter a name for the recording and click **Start**. Recording begins the moment **Start** is clicked. For the Fleet example in this tutorial, we'll use the name "Create a new rental reservation."

   While you're recording, you can click the **Close** button (**X**) in the upper-right corner to hide the **Task recorder** pane without stopping the recording. You can reopen the pane by clicking the **Task recorder** button that appears at the top of the page. This button appears only while recording is in progress.

   > **NOTE**
   > If the Saved views feature is turned on, recordings should be created by using either published views or the standard view, to ensure that recordings work reliably for users.

7. Task recorder enters **recording mode**. The pane shows information and controls that are related to the process of recording.

Now you're ready to record a business process using Task recorder. If you're following this guide as a first-time user, you may complete the following Fleet Management scenario as an example. Otherwise, you can record your own application scenario.

Record a Fleet Management scenario

1. In the **Task recorder** pane, click **Start sub-task**.
2. Set **Name** to "Create a new rental customer". Leave the **Comment** field blank.
3. Click **OK**. The task is added to the step list.
4. Go to **Dashboard > Fleet Management > Reservation Management**.
5. Go to **All customers** under the **Summary** tab.
6. In the Action Pane, click **New**.
7. Enter a first and last name for the customer.
8. Click **Save**.
9. In the **Task recorder** pane, click **End sub-task**.
10. Return to the **Reservation Management** workspace by clicking the browser back button twice.
11. In the **Task recorder** pane, click **Start sub-task**. Name the task "Rent a vehicle to the new customer". Click **OK**.
12. Click (**+**) **Rental** under **Summary**.

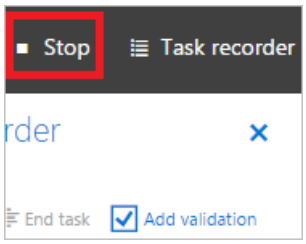
13. Under **Information**, select a "1975 Litware McKinley" as the vehicle.
14. Under **Information**, set the customer to the one just created.
15. Expand the **Discounts** section.
16. Click **Add** under **Discounts** and add the Frequent Customer discount. Click **OK**.
17. In the Action Pane, click **Start Rental**.
18. Set the return date to some date in the future.
19. Click **OK**.
20. In the **Task recorder** pane, click **End sub-task**.
21. Click **Stop** at the very top of the page.

Recording a business process

After you've started your recording, you can perform your business process just as you would typically perform it by using the web client. As you interact with the product, new steps are added to the step list in the **Task recorder** pane. In this section, you will learn about other actions that you can perform while you're recording a business process, to take full advantage of Task recorder's capabilities.

Stop

Stop is used to end the recording session. Before you click **Stop**, you should make sure that the recording is completed, because this action isn't reversible. When you click **Stop**, you're taken to the download options screen.



Start/End sub-task

Start/End sub-task lets a user specify the beginning and end of a set of grouped steps in a recording. Click the **Start sub-task** button to add a "Sub-task" step to the end of the current list of recorded steps. The sub-task will include all steps that you perform from this point until you click the **End sub-task** button. When you click the **End sub-task** button, an "End sub-task" step is also added to the list of recorded steps.

> **NOTE**
> You must start a sub-task before you perform/record the steps that you want to include in the task. Then, after you've performed/recorded all the steps that you want to include in the task, you must end the sub-task.

Sub-tasks are purely an organization tool, and consumers of business process recordings can interpret the task groupings in useful ways. Because tasks can be nested inside other tasks, they provide the flexibility to organize very long and complex business processes.

Delete/Restore step

Delete/Restore step enables a user to remove steps from the recording, or undo the removal of a step from the recording. You must first select the step in the Steps list that you want to delete/restore, and then click the **Delete/Restore step** button.

> **NOTE**
>
> The behavior of the **Delete** button changes when you play back a recording. In playback mode, a deleted step can't be restored after playback has passed the point where it would have run the deleted step. For example, you load a recording that contains three steps, and then you delete step 2 before you start playback. You can restore step 2 only as long as playback hasn't run step 3. After you start playback, and playback has "skipped" step 2 (because you deleted it) and run step 3, you won't be able to restore step 2. Because step 2 wasn't run and therefore wasn't recorded, it can't be retroactively added back into the recording at its previous position.

Add developer placeholder

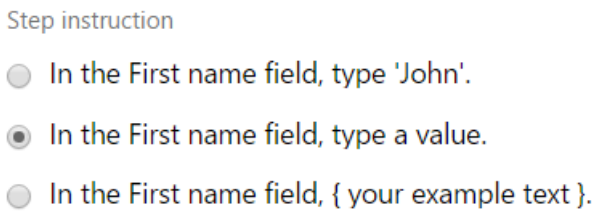
Add developer placeholder lets the user add a placeholder step to the list of recorded steps. This placeholder step doesn't appear when the task guide is viewed, and it isn't run during maintenance of a recording. It's used only by the Regression suite automation tool (RSAT) or the X++ code generator that enables an X++ test to be created from a task recording. When the code generator creates an X++ test, it automatically adds a method stub to the generated code. The developer can then add X++ code into this method stub. The automated code will call the validation when the generated test is run at the point in the recording where this placeholder was added.

Enriching steps in a recording

There are various options for enriching a step in a recording. For example, you can adjust the text that is associated with a step and add information about a specific step. This section describes the step enrichment capabilities that are available. To access these options, click the **Edit step** button on a specific step of a recording.

Step instruction

The **Step instruction** is the primary text that is displayed for this step in the task guide. There are usually 2-3 alternative options for step instructions, and they appear in the following order when editing the annotation.



This image shows the annotation options for changing a step.

- **Preferred value instruction** This type of instruction will direct the user to enter the same data that was used when the step was recorded. *Example*: In the First name field, enter 'John'.
- **Example value label** This type of instruction will direct the user to enter their own data, indicating that the data that was used when the step was recorded was only *Example* data. *Example*: In the First name field, enter a value.

  If users click the **See more** button on this step when they play the recording as a task guide, they will be able to see the data that was used when the step was recorded. This recorded data value will be labeled as an *Example* data value.

  > **NOTE**
  >
  > Steps that are not related to fields, such as clicking buttons, opening forms, or selecting records from a lookup, do not set *Example value label* as an option when annotating.

- **User-supplied value label** This step instruction contains placeholder text, which the author can fill in with their own text. For steps which have an **Example value label** option, the placeholder allows substituting the text which normally specifies the data to enter. This is useful for scenarios where neither the **Preferred value label** nor the **Example value label** sufficiently express the data that should be used for this step.

  - *Example label*: In the First name field, enter *{your example text}*.
  - *Example label after supplying the placeholder text*: In the First name field, enter the customer's name.

  For steps which do not have an **Example value label** option, the placeholder allows substituting all of the label text. Steps associated with buttons, for example, do not have **Example value labels**, so you may replace the entire label text with your own text.

  - *Example label before replacement*: Click Post.
  - *Example label after replacement*: To post the order, click Post.

Titles and notes

Titles and notes provide places for user-specified text to be associated with a step in a task guide.

- **Title** – The title lets you specify the text that appears above the step instruction for this step in the task guide. The title a good place to put text that you want users to read before they complete the action that is indicated by the step instruction.
- **Note** – You can use a note to specify text that appears in the expandable section of the pop-up for this step in the task guide. A note is a good place to put optional reading material or other information that might be useful to users, but that they aren't required to read to complete the action that is indicated by the step instruction.

Change recorded values

Starting in version 10.0.12, you can adjust the values that are recorded in basic input controls (for example, simple text, numeric, date, and picklist fields), without having to re-record those steps. Note that lookup controls and reference groups aren't currently supported.

Hide from task guide

The **Hide this step** option lets the author prevent specific steps from appearing in the task guide. This option is useful for hiding steps that are required for the task recording to run in playback mode, but that should not be seen by users. Examples of these steps include copy steps, system-generated steps, and data clean-up steps. If you hide a sub-task, all the steps that are recorded inside that sub-task will also be hidden.

Using control gestures

The basic recording capability lets a user record an end-to-end business process by using Task recorder, but without adding overhead to the process. In some circumstances, more advanced recording features can be used to create even richer business process recordings. Each of the following gestures is found under the **Task recorder** option on the shortcut menu (also known as a right-click menu or context menu) for a control and causes a step to be added to the recording. If the gesture isn't supported for a control, it won't appear on the shortcut menu for that control.

Copy

The **Copy** gesture lets you copy the value for the current control to the Task recorder "clipboard." That value can then be used later as part of a **Paste** or **Validate** gesture. Because values from multiple controls might have to be pasted, the Task recorder clipboard maintains a list of all control values that have been copied in the recording.

Paste

The **Paste** gesture lets you paste a value from a previous **Copy** gesture in the same recording. The Task recorder

paste function works like the standard paste function that users might be familiar with, but it has an additional benefit when it's used during recordings. Because Task recorder will replay the recorded **Copy** and **Paste** commands during playback, if the copied control has a different value than it had during recording, Task recorder will paste the current value instead of the value that the copied control had during recording. This feature is useful in scenarios where the copied control has a value that can change between environments (for example, recID values or number sequences).

There is an additional benefit from using the **Copy** and **Paste** gestures when test code is generated. For any control where the value is set via the **Paste** command, Task recorder doesn't have to create a parameterized input variable for that control's value, because it's set based on another control's value. This feature can be very useful in scenarios where an entity such as a customer is created, and an identifier for that entity is frequently entered during the recording. Instead of manually re-entering the customer name or ID throughout the scenario, and causing Task recorder to generate a parameterized input variable for each entry, the user can copy the customer name or ID one time, and then repeatedly paste it. In this case, Task recorder will generate a single parameterized input variable to represent the customer name or ID. This feature can make it much easier to change the input data for a generated test.

Validate

The **Validate** gesture lets you insert a step that validates the value of the targeted control. This gesture always uses equality to validate the control value. *Validations aren't currently run during recording playback.* Instead, they are run only when the generated test code is run. Two kinds of validation are available:

- **Current value validation** will capture the targeted control's value at the time of recording and use it to generate an assertion in the test code. In the list of validation options on the shortcut menu, **Current value** is always first.
- **Reference value validation** will use the value of a previously copied control when generating an assertion in the test code. This allows creating assertions that are resilient to changes in the data, since the value is not hardcoded into the test code. In the list of validation options on the shortcut menu, **Reference value validation** follows the format [AOT name of copied control: current copied value].

Additional options are available in version 10.0.13 and later. Here are some examples:

- **Enabled/Disabled** validates that the targeted control's state is enabled (or disabled), and then uses that validation step to generate an assertion in the test code.
- **Read-only/Editable** validates that the targeted control's state is read-only (or editable), and then uses that validation step to generate an assertion in the test code.

Add info step

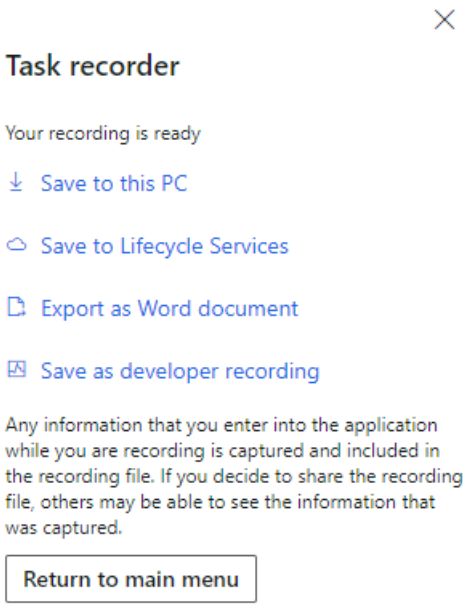
The **Add info step** gesture lets you insert a step and supply your own text for it. This feature is useful primarily for creating task guides. An **informational step** (or **info step** for short) is a task guide step where the instruction text for the step is user-specified. Info steps are useful for describing actions that are a part of the scenario but must occur outside the client. For example, a scenario might require the user to search for item inventory or check an email for information.

You can specify where an info step should appear in the task guide. The info step can point to a control on the page, if the step is associated with the control. Alternatively, the info step can appear in the upper right of the page, if the step is external to the client, or if it's an explanation that applies to the whole page.

> **NOTE**
>
> Because info steps are manually specified steps and are not automatically recorded by Task recorder when the user takes an action on a control, the info step does not have the capability to automatically progress when a user completes the step in the task guide. Because the info step is not associated with taking an action in the client, there is no action for a task guide to detect that the user has completed in order to automatically progress to the next step.

Options after a recording is completed

After you click **Stop** to end your recording session, several options are shown so that you can save the files that are related to the completed recording. Select **Save to this PC**, and save the task recording package to your desktop. You will use this file later.



Save to this PC

One option after you finish your recording is to download the task recording package (an .axtr file) to your computer. This file can be loaded later via the **Task recorder** pane, so that the recording can be played as a task guide or edited.

Save to Lifecycle Services (LCS)

When you save your recording to an LCS library, it's published on the specified business process in a BPM library. If the selected LCS library is set as a Help library, you will be able to find the task guide for the recording by searching the **Help** menu.

> **NOTE**
> To be able to save a recording to an LCS library, the user must be in the Azure Active Directory (Azure AD) tenant that the environment was deployed from.

Export as Word document

The Microsoft Word document for your recording contains the recorded steps as well as any screenshots that were captured.

Save as developer recording

The raw recording file (developer recording) is useful for developer scenarios, such as test code generation and scenarios where RSAT is used.

Playing back a recording

The **playback** functionality of Task recorder can automatically run the steps of an existing recording by using the pages and values that were originally recorded. Playback mode can be used to update an existing recording if changes were made to the underlying application, and those changes altered the business process steps that are required for the scenario. It's important to remember that, in this mode, Task recorder simultaneously re-

records the steps and plays them back. When the playback is completed, a new recording is produced that reflects both the steps that were run from the existing recording and any new steps that the user manually performed. Any steps that aren't run either by the user or automatically by Task recorder aren't included in this new recording.

To play back an existing recording, follow these steps.

1. Refresh the browser tab.

   > **NOTE**
   > It's a good practice to refresh the browser before each new recording.

2. Open the **Task recorder** pane.
3. Click **Playback recording**.
4. Click **Open from this PC** to load a recording from a previously downloaded Task recorder package (.axtr file).
   - If you're reading this guide for the first time and following along, choose the "Create a new rental reservation" file that you downloaded previously.
5. Click **Start**.

When you play back a recording, additional actions are available in the **Task recorder** pane.

Play next pending step

Play next pending step runs the next step in the recording. This action is useful because it gives you more control over the playback speed when you want to analyze the effects of a single step. This action has a side-effect that it's important to be aware of. When you click **Play next pending step**, any open lookups, drop-down dialog boxes, or Action Pane tabs might be dismissed, because this action removes focus from those elements. For these situations, we recommend that you use **Play all pending steps** instead.

Play all pending steps

Play all pending steps begins sequential execution of the remaining steps in the recording, and continues until either playback is paused or all steps have been run. During playback, the **Play all pending steps** button is replaced by a **Pause** button that can be used to pause playback. If playback can't successfully run a step for any reason (for example, because it can't find a button that has been renamed), Task recorder will skip that step, and playback will automatically be paused. In this way, the user has an opportunity to replace the obsolete step by completing the new steps in the client. Task recorder will record the new steps and ignore the step that was skipped. The user can then click **Play all pending steps** to continue playback for the remaining steps. After the recording is completed, the user can download the updated recording. This recording will contain all the steps of the original recording, but will exclude any skipped steps and include any new steps.

Play to selected step

Play to selected step behaves like **Play all pending steps**, but it lets you play only a subset of the steps instead of all the steps. In the list, select the step that you want playback to stop at, and then click **Play to selected step**. Task recorder will begin to run the steps in the list and will stop when it has run the step that you selected.

Editing a recording

Although you can edit a recording through the playback functionality, there is also a mode that lets you make simple edits to a recording without having to replay the whole recording. To access this feature, click **Edit recording** after you open the **Task recorder** pane. You can use this feature to make the following edits:

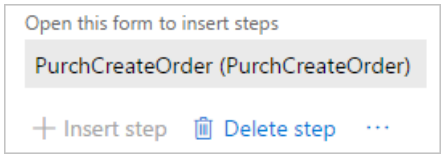
- Insert steps into a recording without re-recording the whole file.
- Move steps under a sub-task without re-recording the whole file.
- Adjust the name and description of the recording.

Insert steps without re-recording the entire file

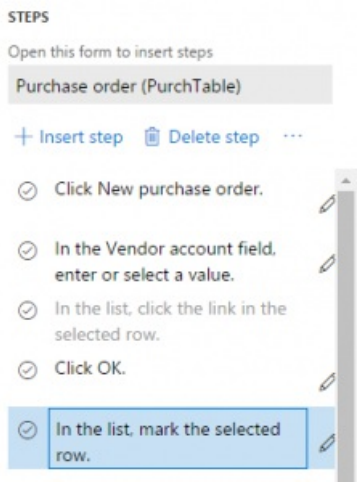
You can add a step anywhere in a task recording without playing back or re-recording the whole file.

1. Select the step after which you want the new step to be inserted. Make sure the step is highlighted.

   In order for task recorder to insert a step, you must have the correct page open. The correct page is the page on which the new step occurs. Task recorder has a mechanism that determines what the active page is, and will disable the functionality if the correct page isn't open.

   When you are on the correct page, **Insert step** becomes available.

2. Click **Insert step**.

   When you click **Insert step**, Task recorder switches to recording mode. Any action that is performed in the user interface (UI) will now be recorded and inserted into the recording as steps.

3. Click **Stop**.

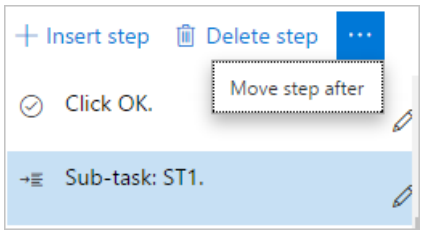
   Recording mode is stopped, and you can now continue to edit the recording. For example, you can repeat this process to insert steps in other places in the recording, or you can move sub-tasks as described in the next section.

4. When you've finished editing the task recording, click **Done editing**, and then select one of the options to save or publish the recording.

Move steps under a sub-task without re-recording the entire file

You can move steps under a sub-task without playing back or re-recording the entire file. You can also move the sub-task step or the end sub-task step if you want to group an existing block of steps.

1. Select the step or sub-task step that you want to move. Make sure that the step is highlighted.
2. Click **Move step after**. To access this command, you might have to select the ellipsis (...) button.



3. Select the step or sub-task step that you want to move the step or sub-task step after. Task recorder will move the step.

4. To move the end sub-task step, select it, click **Move step after**, and then select the step that you want the end sub-task step to be after.

   If you want the first step in the task guide to be within a sub-task, create a sub-task step as the second step, and then move the first step into it. You can add or move as many steps or sub-tasks as needed.

5. When you've finished editing the task recording, click **Done editing**, and then select one of the options to save or publish the recording.

Adjust the recording name and description

You can adjust values of the **Recording name** and **Recording description** fields. If you want to see more steps in the Task recorder editing pane, you can also collapse the section that shows the recording name and description.

Playing a task guide

A **task guide** is a user-focused experience that lets the user follow a guided step-by-step set of instructions to complete a business scenario by using a task recording. The user is instructed to complete each step through an animated pop-up prompt that will move across the page and point to the UI element that the user should interact with. The prompt will also tell the user how to interact with the element. For example, it might state, "Click here" or "In this field, enter data." Each step that the user is instructed to complete is based on a step that was originally recorded in the task recording. Because the task recording file contains the data that describes the step that was originally recorded, the task guide can automatically determine when the user has completed the step as expected. It then automatically moves on to the next step.

> **NOTE**
>
> One way that the task guide determines that a user has completed a step is by detecting when the value in a field has changed. Although the task guide doesn't require that a specific value be set, it does require that the field value be changed in order to determine that the step was completed. The user must change the field value, and then press the **Tab** key or click in an area outside the UI element. Only at that point does the client detect that the field value has changed, and it can then proceed to run any required application validation or business logic. Therefore, before the task guide can determine that the step was completed by the user, it relies on the client to detect that the field value has changed.

What can a task guide allow a user to do?

When a user is completing a task guide, the client behaves in the same manner, with the same data, security, and validation rules as it does when the user is not completing a task guide. There is no difference of behavior in the client that would allow a user to take an action that they cannot otherwise take when they are not completing a task guide. When a user is completing a task guide:

- Any data the user enters is subject to the same data validation rules as when not playing the task guide.
- Any data the user enters may be saved, and the user may modify data according to the same restrictions and rules as when not playing the task guide.
- Any security mechanisms the user encounters behave the same as when the user is not playing the task guide.
- Any forms or controls the user accesses are subject to the same security and access mechanisms as when the user is not playing the task guide.

The "On-rails" feature of task guides

By default, when a user begins a task guide, they are placed "on-rails". These "rails" prevent the user from *clicking* on elements other than the element the task guide is pointing to. When the user tries to click on something outside of the UI element that the task guide is pointing to, the task guide pop-up will animate to let the user know that they cannot progress until they complete the current step of the task guide.

While a user is prohibited from *clicking* on other elements, the user is not prevented from tabbing through the other controls on the form, and the user is not prevented from using keyboard shortcuts. This is by design, as the "on-rails" feature is designed for and targeted at first-time users, who are expected to primarily use the mouse as they become familiar with the application.

More advanced or experienced users can turn off the "on-rails" feature when they complete a task guide. At any point during the task guide, these users can turn off the rails by clicking the **Unlock** button that appears on the Task recorder toolbar at the top of the page. This button can also be used to restore the rails at any point during the task guide. In some situations, the task guide might automatically turn off the "on-rails" feature. When the rails are turned off, the user can click UI elements just as they do when the task guide isn't running. The "on-rails" feature might be automatically turned off in the following situations:

- The user is being directed to go to a page by using the navigation pane or navigation search.
  - Because the user can use either entry point, the task guide doesn't point to a specific entry point, and it doesn't prevent the user from using either entry point.
- The task guide enters an error state (see the next section for a list of error states).
- The task guide is showing an info step.

Error detection

An *error state* occurs when the task guide is not able to point to the UI element that is associated with the current step because the UI element is not visible on the screen. When the task guide detects that the current step requires the user to interact with a UI element that is not visible, then the task guide pop-up will move to the upper-right side of the screen. These causes of an error state can be simplified into two categories.

The control is not visible on the form

This error state usually occurs when the user has opened or closed the incorrect tab, FastTab, collapsible section, FactBox, or pop-out menu.

Because the UI element that is needed for the current step is somewhere on the current form, but it is not visible on the screen, the task guide pop-up will simply move to the upper-right side of the screen while displaying the same instruction that informs the user of the action they need to take.

Because the task guide can't find the UI element on the screen, the user must manually determine what is causing the UI element to be hidden and then make the element visible on the screen. The task guide pop-up will automatically detect that the UI element is visible and will reposition itself so that it's pointing at the now-

visible element.

The control is not on the form

This error state usually occurs when the user has gone to the wrong form, either by navigating to the wrong form or by leaving the correct form.

Because the UI element is not visible on the screen, the task guide pop-up will move to the upper-right side of the screen. In addition, when the task guide detects the user is on the wrong form, the task guide pop-up text will change to inform the user of the form they should navigate to.

In some cases, the task guide pop-up will not mention the form by name. This is because the user may need to navigate to a dynamic form. A dynamic form is a form that is not modeled, frequently known as a runtime-generated form. These sorts of forms do not have a proper name. Some examples of runtime-generated forms include simple and custom lookups. The way for a user to navigate to a lookup form is to re-open the lookup.

Next step and Previous step

The **Next step** and **Previous step** buttons appear in the task guide pop-up and let a user manually control the flow of the task guide. When these buttons are clicked, the task guide will go to the next or previous step. The task guide doesn't verify that the user has completed a step before it goes to the next or previous step.

The task guide **never** automatically completes any step for the user, even when the **Next step** and **Previous step** buttons are used. Use of these buttons can cause an error state if the previous or next step refers to a UI element that isn't on the current page. When the user is completing an info step, the only way to proceed is to use the **Next step** button. This action is required because an info step doesn't represent an action that was recorded on any UI element. Because no action was recorded in the task recording, the task guide doesn't have the necessary information to determine what action it should expect the user to complete.

The See more button

When the **See more** button is clicked, the task guide pop-up expands to show additional information that is related to the step. The additional information is often optional reading material that isn't required for the user to successfully complete the step. The following information might be included:

- An **Example** value
  - The Example value is the value that was originally used when the task recording was created.
  - Example values appear only for steps that use non-lookup fields. These fields include text fields, number fields, date fields, combo boxes, and check boxes.
- A **Note**
  - A Note may contain scenario-specific information that will help provide context to the user about the current step of the task guide.

Taking screenshots in Task recorder

By using a **pre-release** Chromium browser extension that works for both the new (Chromium-based) Microsoft Edge browser and Google Chrome, Task recorder can take screenshots of the browser as a user records a business process. After the user completes the recording, Task recorder can use these screenshots to generate Microsoft Word documents. To turn on this functionality, follow these steps to install the pre-release Chromium extension that enables Task recorder to take screenshots during recording.

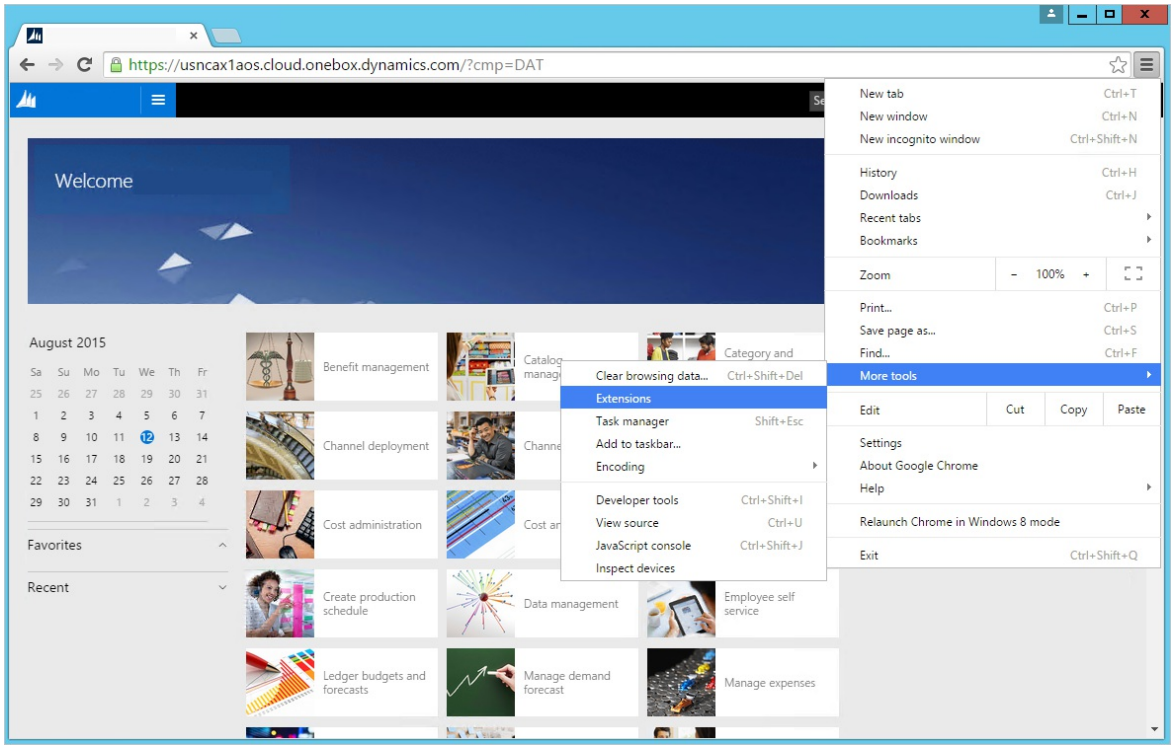
1. Download the **FMLabTaskRecorderScreenshot** folder that contains the extension from GitHub, at https://github.com/Microsoft/FMLab.
2. **On-premises deployments only:** Adjust the manifest for the extension so that it matches the following code. Replace <hostname> with the base URL for your environment.

```
...
"content_scripts": [
    {
        "matches": ["https://*.dynamics.com/*", "<hostname>"],
        "js": ["screenshot.js"]
    }
    ...
```

3. **21Vianet deployments only:** Adjust the manifest for the extension so that it matches the following code. Replace **.com** with **.cn**

```
...
"content_scripts": [
    {
        "matches": ["https://*.dynamics.cn/*"],
        "js": ["screenshot.js"]
    }
    ...
```

4. Open the latest Microsoft Edge browser or Google Chrome.
5. Select **Settings and more > Extensions** in Microsoft Edge (or **Customize and control Google Chrome > More tools > Extensions** in Google Chrome).



6. Select **Developer mode**.
7. Click **Load unpacked extension**.
8. Browse to the folder that contains the Task recorder extension by using the path **FMLab-master > FMLab > TaskRecorderScreenshot**, and then select **Select Folder**.
9. Make sure that **Enabled** is selected so that extension is turned on.
10. Restart the browser.

Task recorder will now take screenshots of the tab where the client is running. These screenshots are available for one week after the recording has been played. (If you're running a platform version that is earlier than

Platform update 16, the screenshots are available for only 15 minutes.) If the screenshots have expired, you can regenerate them by playing the task recording again.

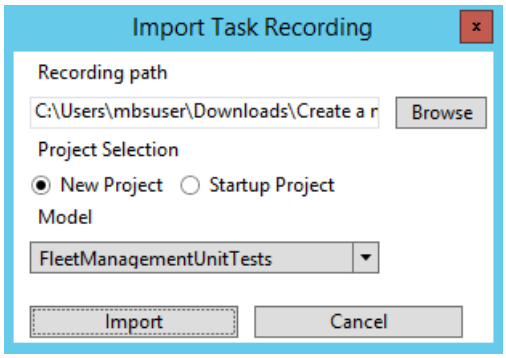
Note that Task recorder **does not** capture screenshots from other tabs or of the user's desktop.

Generating tests from a recording

After a business process recording has been completed by using Task recorder, a developer can import the raw developer recording file (.xml file) into Visual Studio to create an X++ test. The import tool generates a human-readable X++ test from the recording, and translates any control gestures, validations, or tasks into the appropriate test code.

Import a recorded test

1. Open Visual Studio by using the Finance and Operations development tools.
2. Go to **Dynamics 365 > Addins > Import task recording**.
3. In the **Import task recording** menu, use the **Browse** button to locate a previously downloaded recording file.
4. Optionally, choose to have the generated test code be added to the startup project. This requires that a solution containing a project is set as the startup project. This will place the generated X++ test into the same model as the project.
5. If you're creating a new project, select the model for the project. The generated X++ test will be put in this model. For the generated test to be successfully built, the model must have references to the **TestEssentials** model.
6. Click **Import**.



7. In the **New Project** dialog box, provide a name for the project.
8. After the project is created, the user can open and inspect the generated code.
9. To run the test, build the project.
10. Go to **Test > Windows >Test Explorer**.

Appendix

Controls that are known to have incomplete support for Task recorder

- Table
- Filter pane, which is the filter that pops out from the left side
  - When adding filters to the filter pane, the steps are delayed. The steps do not get recorded until the user clicks "Apply" on the Filter pane.
- Enhanced previews
  - No planned support for recording gestures inside of enhanced previews. While recording, enhanced

previews will be disabled.

- No extensible controls are supported out of the box, except Segmented Entry.
  - Extensible control owners need to individually build support for Task recorder.

Controls that can be recorded, but have limited support for the Copy/Paste/Validate gestures

- Date/Time
  - Doesn't support copy/pasting "Never" as a value.
- Image
  - No ability to copy/paste/validate an image value.
- Filter pane
  - Copy/Paste works, but the UI will not show the pasted data. You can proceed as if it pasted correctly.
- Message box
  - You cannot validate the text in the message box.

Controls that are known to have incomplete support for being used in a task guide

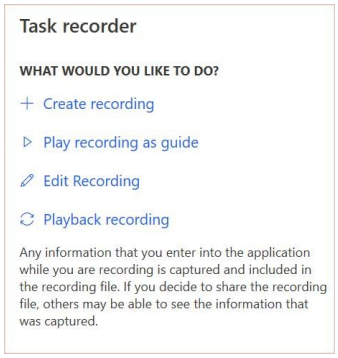
- Quick Filter, which is the filter control that appears above lists
  - Does not support displaying a "generic value" during the task guide. Currently displays the value that was used during recording.
- Filter pane, which is the filter that pops out from the left side
  - The task guide does not point to the individual elements within the Filter pane that need to be clicked on.

Task Recorder quick reference

2/18/2021 • 5 minutes to read • Edit Online

This article provides a quick reference sheet that explains what each button in the Task recorder menus does.

Main menu



Create recording

Choose this option to begin creating a new recording.

Play recording as guide

Choose this option to see what your recording looks like when viewed as a Help topic or played as a Task guide.

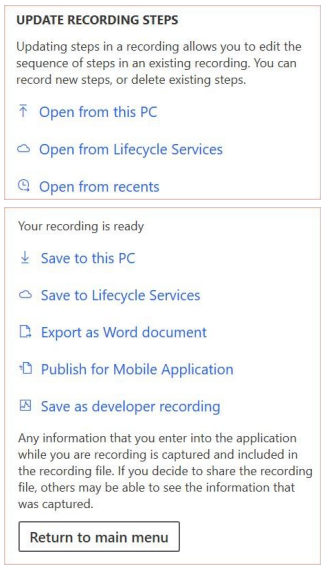
Edit recording

Choose this option if you need to change the recording's name, description, or the text that is displayed in the steps.

Playback recording

Choose this option if you need to add or remove steps. You can also use this mode to automatically play a recording.

Open and save options



Open/Save from/to this PC

These options allow you to open a recording that is saved on your computer, or save a recording to your computer.

Open/Save from/to Lifecycle Services

This option allows you to open a recording that has been saved to a Lifecycle Services library, or save a recording to a Lifecycle Services library.

Open from recents

This option allows you to pick from a list of Task recordings that you have recently created.

Export as Word document

This option allows you to download a Word document that contains the list of steps in the recording.

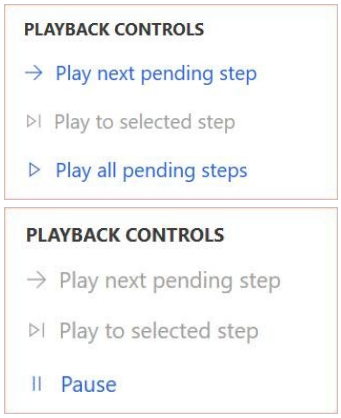
Publish for Mobile Application

This option allows you to publish the task recording for a mobile application.

Save as developer recording

This option allows you to save the task recording as a developer recording.

Playback controls



Play next pending step

This option will cause Task recorder to execute and record the next pending step, which is indicated by the arrow in the Steps list.

Play to selected step

This option will cause Task recorder to begin playing and recording pending steps, beginning at the next pending step and pausing after playing the step that was selected in the list when this option was clicked.

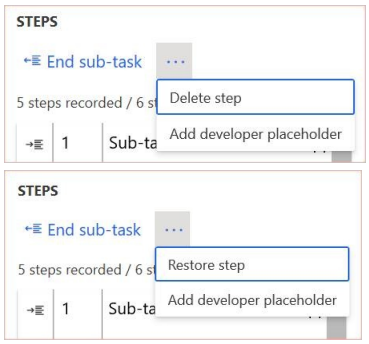
Play all pending steps

This option will cause Task recorder to play and record all remaining pending steps, until there are no remaining pending steps.

Pause

This option only appears while playback is in progress. This option allows you to pause playback manually.

Step actions



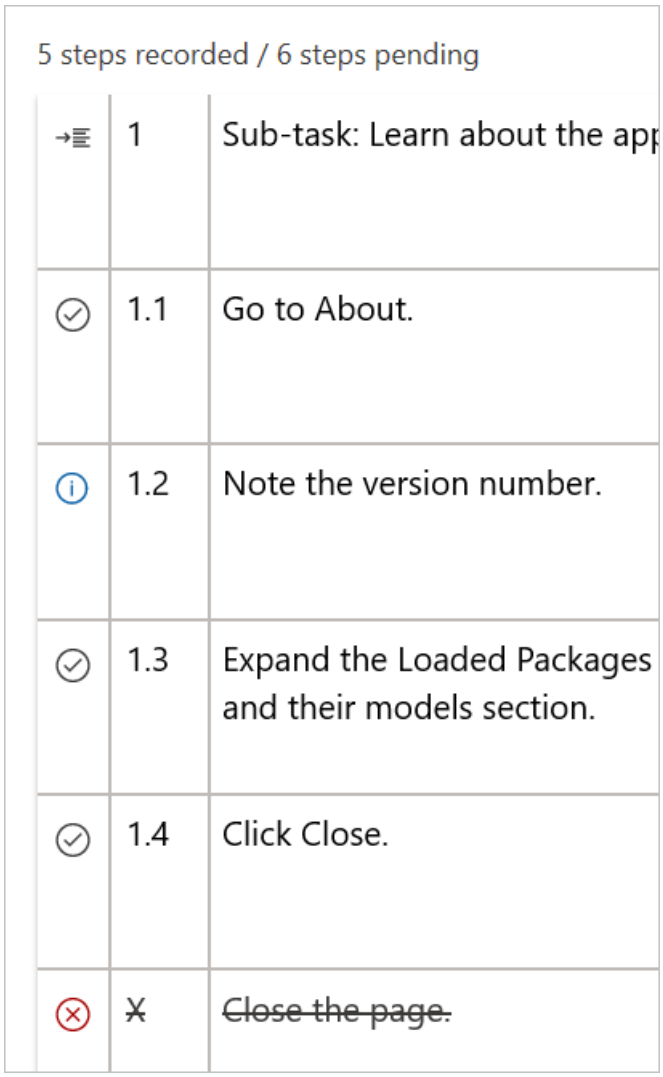
Start sub-task/End sub-task

These options allow you to add special steps to the recording. These special steps are task steps, and you can use them to indicate when a sub-task begins and when it ends. These options are disabled while playback is in progress.

Delete step/Restore step

These options allow you to remove steps from the recording. If you delete a pending step, it will be skipped during playback, and it will not be recorded. If you delete a recorded step, then it will be flagged for removal and it will not be included in the recording when you save the recording. You can only restore steps that have been successfully recorded. You cannot delete a task while it is in progress.

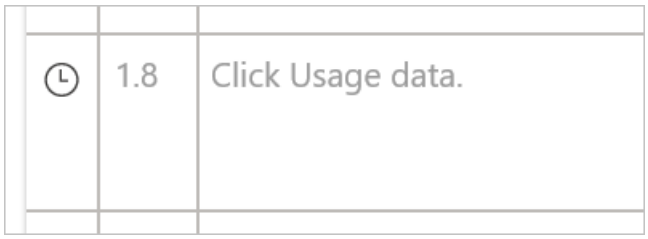
Steps list



Step counter

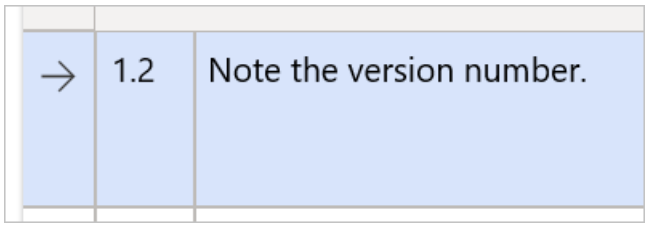
This keeps track of how many steps have been recorded. This includes steps played by using the Playback controls, as well as steps recorded by actions that you take in the client.

Pending step



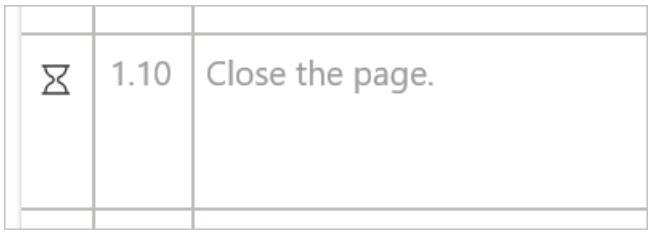
This symbol represents a step that is pending and has not been recorded yet. Pending steps can be played using the Playback controls. When a pending step is played successfully, it is recorded and the symbol will update appropriately. *Pending steps are not included in the recording when you save the recording.* You must first play the pending steps so that they are recorded. If the steps are played and recorded successfully, then they will be included when you save the recording.

Next pending step



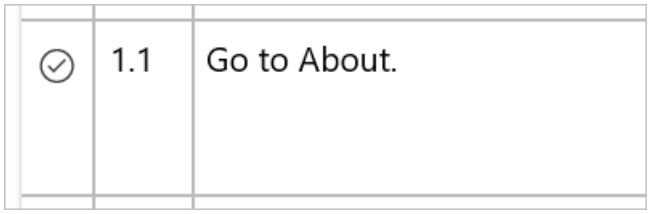
This symbol represents the next pending step. If you start playback, this is the first step that will be played.

Queued pending step



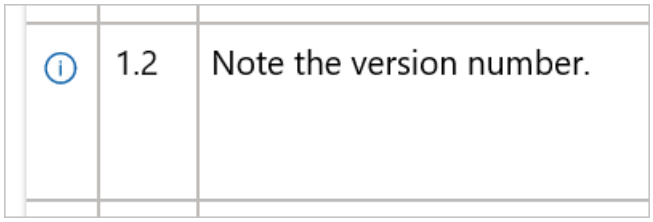
This symbol represents pending steps that are queued for playback. This symbol is updated either when playback pauses, or when the queued pending step is played.

Recorded action step



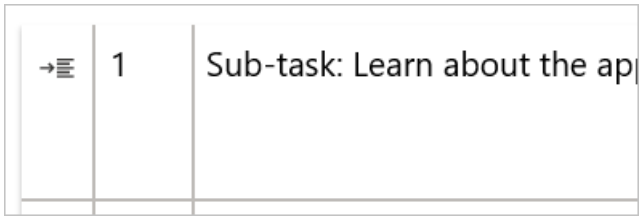
This symbol represents steps that were recorded successfully, either from being played back, or manually recorded by you.

Recorded info step



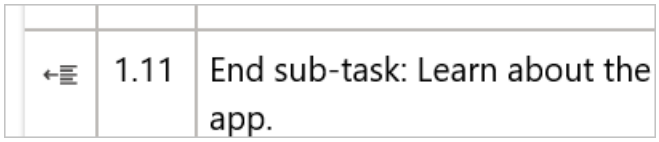
This symbol represents an info step that was played and recorded. Info steps do not result in any action being executed on the application.

Recorded begin sub-task step



This symbol indicates the beginning of a sub-task. Sub-task steps do not result in any action being executed on the application.

Recorded end sub-task step

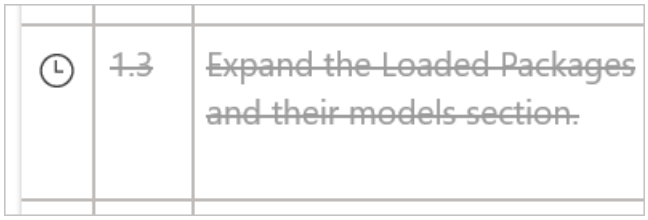


This symbol indicates the end of a sub-task. Sub-task steps do not result in any action being executed on the application.

Deleted recorded step

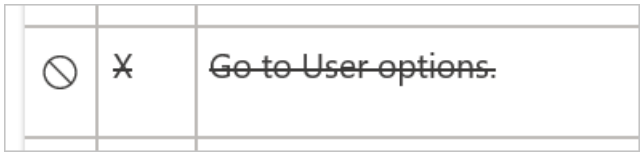
This symbol represents a successfully recorded step that you have marked for deletion. Recorded steps that are marked for deletion will not be included when you save the recording. If a step has been successfully recorded when you decide to delete it, then you have the option to restore the deleted step before you save the recording.

Deleted pending step



If you delete a pending step, it will retain its pending symbol until it is played. When it is played, it will be skipped. You can restore a pending step as long as it has not been played and skipped.

Skipped step



This symbol represents a step that was deleted while it was pending, and was skipped during playback. Skipped steps are not played and are not recorded. Because skipped steps are not recorded, they are not included when you save the recording. You cannot restore a skipped step.

Error step

This symbol represents a step that was attempted by the playback system, but was not successful in being played. Error steps are not recorded, and are not included when you save the recording. You cannot restore an error step. Playback will automatically pause when an Error step is encountered. This gives you the opportunity to record replacement steps before continuing playback. An error step may occur for the following reasons:

- The step could not play because the form or lookup needed by the step was not open.
- The step could not play because the button or field needed by the step was disabled, not visible, or not present on the form.
- The step could not play because the name of the form or name of the control has changed.
- The step could not play because of a framework change to the control.

Create documentation or training with Task Recorder

2/18/2021 • 8 minutes to read • Edit Online

This topic explains what Task recorder and task guides are, how to create task recordings, and how to customize Microsoft task guides and include them in your Help.

> **IMPORTANT**
> You can record your own task guides for Dynamics 365 Human Resources, but you won't be able to save them to a Business Process Modeler (BPM) library or open them from the Help pane at this time. You can save them locally or to a network location, and then open and replay them using Task recorder.

Learn about Task recorder

Task recorder is a tool that you can use to record actions that you take in the product user interface (UI). When you use Task recorder, all of the events that you perform in the UI that are executed against the server—including adding values, changing settings, removing data—are captured. The steps that you record are collectively called a *task recording*. Task recordings can be used in many ways:

- **Task recordings can be played as task guides.** Task guides are an integral piece of the Help experience. A task guide is a controlled, guided, interactive experience through the steps of a business process. The user is instructed to complete each step by way of a pop-up prompt (or "bubble"), which will animate across the UI and point to the UI element that the user should interact with. The "bubble" also provides information about how to interact with the element, such as "Click here" or "In this field, enter a value." A task guide runs against the user's current data set and the data that is entered is saved in the user's environment.
- **Task recordings can be saved as Word documents.** This allows you to easily produce printable training guides.

You can create your own task recordings, play task recordings provided by Microsoft, or modify Microsoft-provided task recordings to reflect your configuration. For more information about Task recorder, see Task recorder.

Plan your task recording

Whether you're creating a new task recording or basing your recording on a Microsoft task recording, keep the following information in mind.

- Plan your recording like you would a video. Make all your decisions ahead of time.
- Walk through the business process once or twice without recording it to understand the steps.
- When you walk through the process before you record, notice where you use shortcut keys or the **Enter** key, so that you can avoid using them during the actual recording.
- Identify the following:
  - Do you want to group steps together into sub-tasks? Sub-tasks visually set apart sections of a process. For example, if you are creating a recording for "Creating and releasing a product," you may want to group together the steps that are required to create a product, and then group together the steps that are required to release the product. Sub-tasks also make longer processes easier to read.

- Do you want to add annotations, and if so, where? See "Understand the different types of annotations" below for more information.
- What values will you add in the various fields as you complete the steps of the business process? It is a good idea to know what you'll select or enter as you proceed so that you don't backtrack or fix mistakes as you're recording.

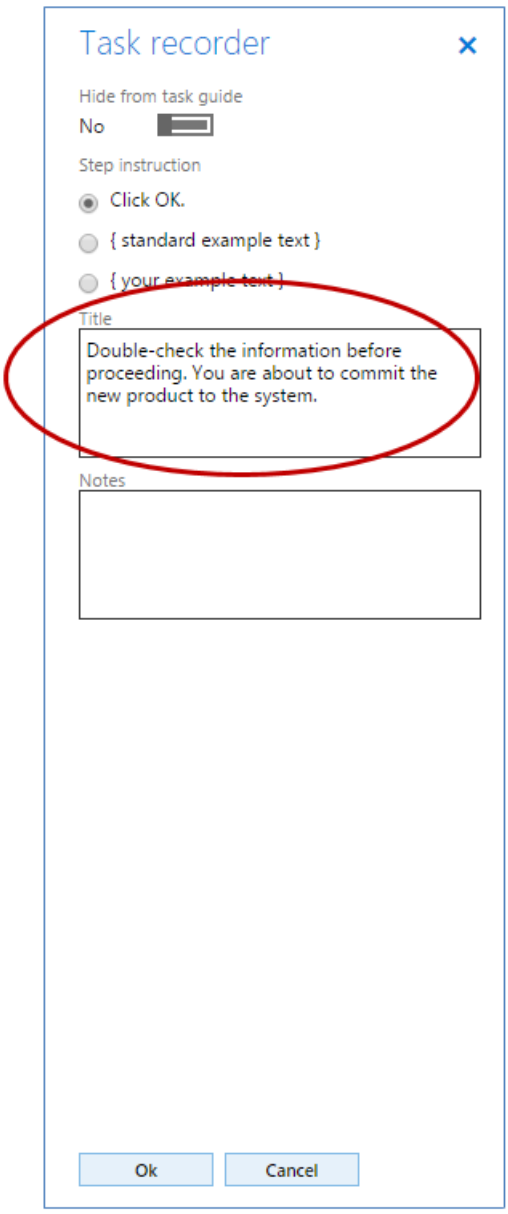
Write your description and annotations ahead of time

- At the beginning of each task recording, there's a description field that allows you to enter an introduction to the recording. It is a good idea to write and save the description ahead of time in a separate document so you can copy and paste it into the task recording when you are recording. That way, you can spend time refining the text when you aren't in the process of recording. Cutting and pasting the text makes the recording process go more quickly and smoothly.
- For each step in a task recording, you can create annotations. During playback of a task guide, annotations appear in the "bubble" as notes above or below the text for the step. When viewed as text in the Help pane, annotations appear as text inline in the step. As with the description, it is a good idea to write and save your annotations in a separate document. When you're recording the task recording, cut and paste the annotations in from that document.

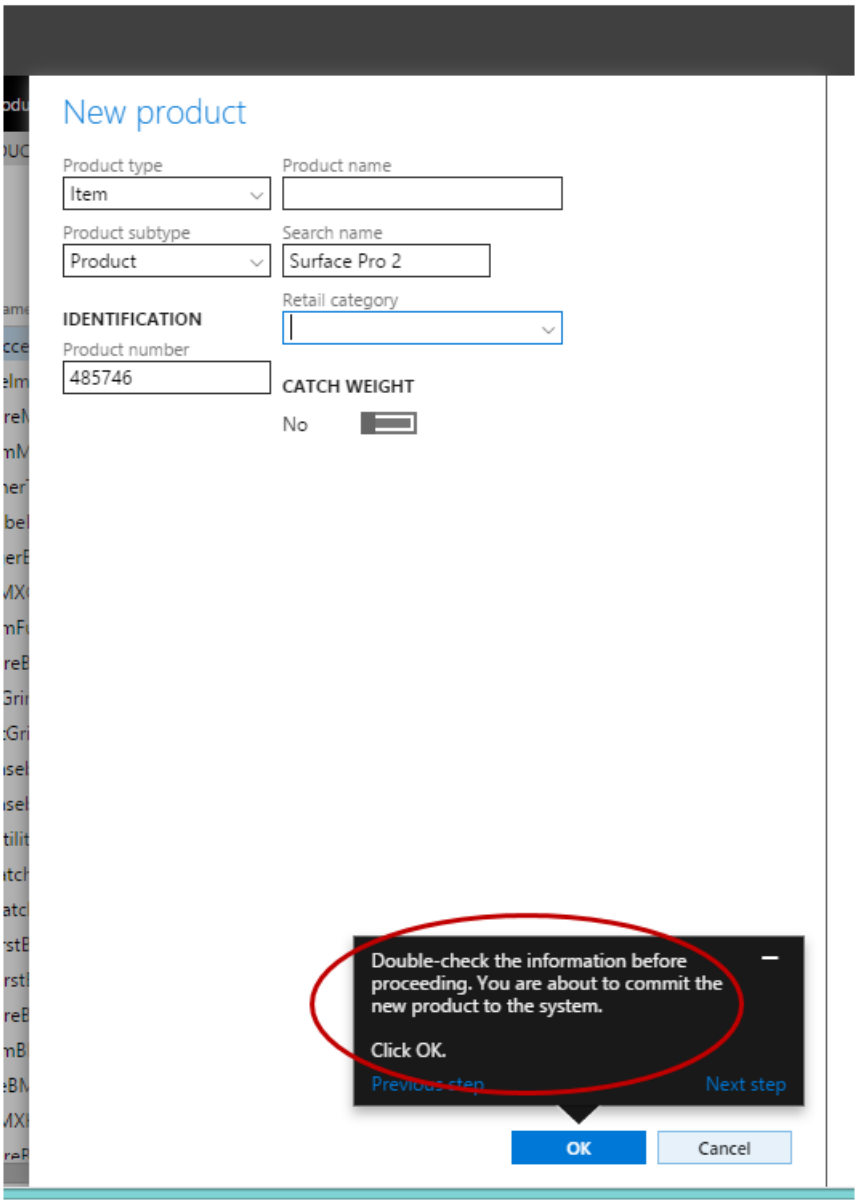
Understand the different types of annotations All annotations are optional. Only add them when they'll provide helpful information to the user.

- **Title**: A title annotation will appear before the step text that task recorder automatically generates. In the task guide, the title annotation appears above the automatically generated text. Use this type of annotation to explain why the user is doing the step or to give additional context.

This is the editing pane that you see when you add an annotation as you create your recording. Enter a title annotation in the **Title** box.

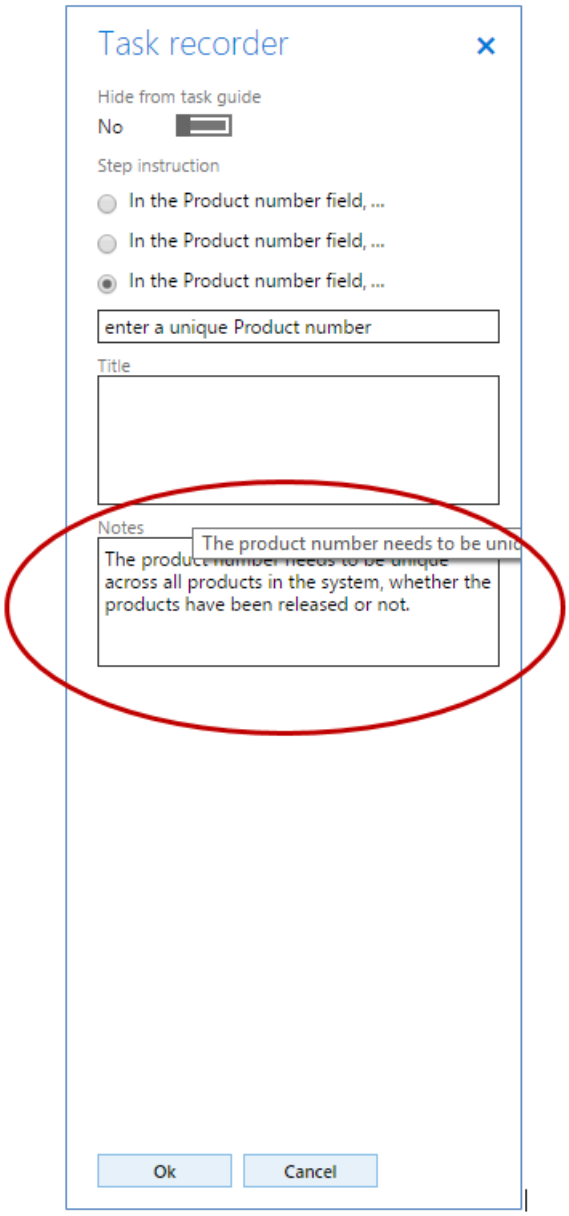


This is what the title annotation looks like in the "bubble" in the task guide.

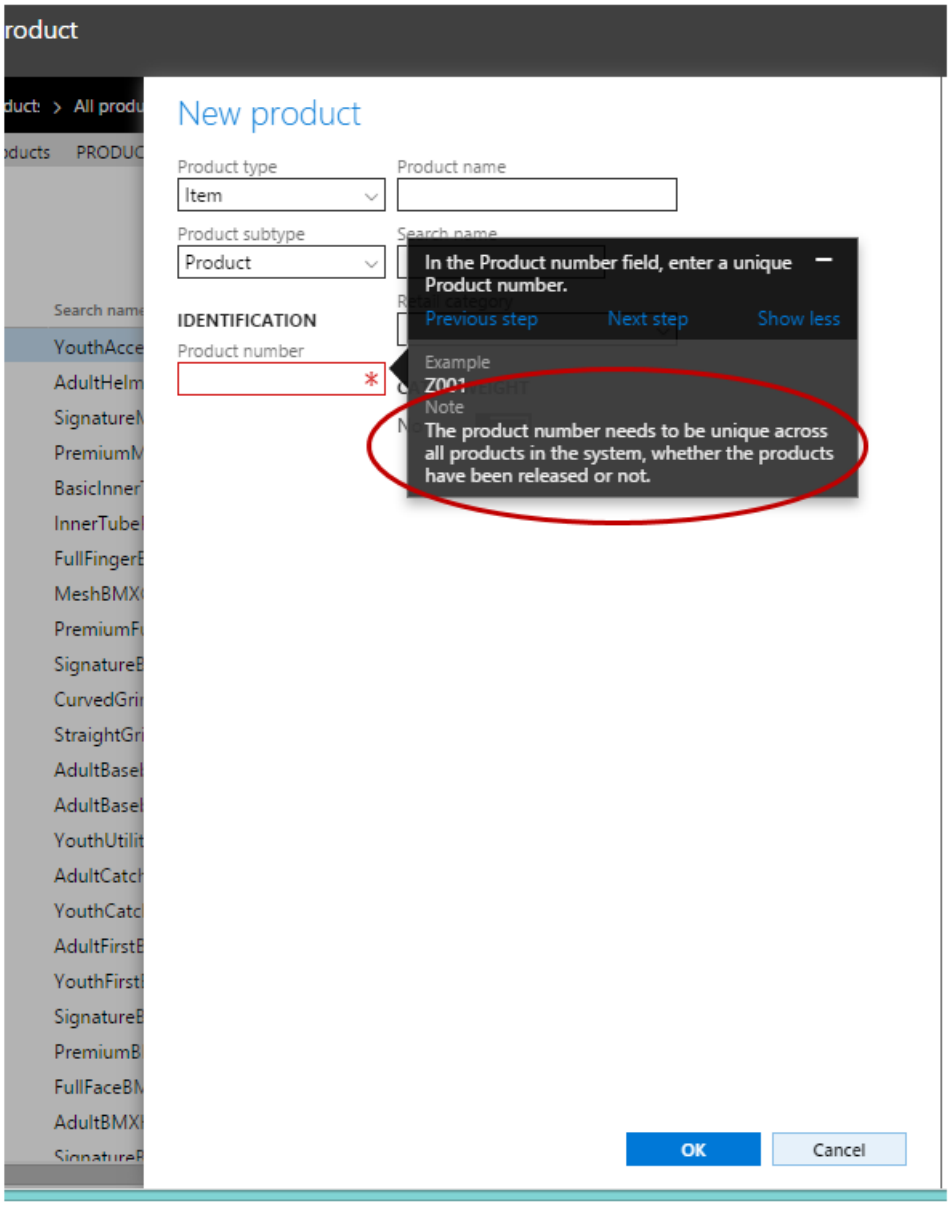


- **Notes:** A notes annotation will appear after the step text that task recorder automatically generates. In the task guide it will only be visible if the user clicks the **Show more** link in the task guide bubble. Use this type of annotation to describe anything that a user needs to know to complete the step.

This is the editing pane that you see when you add an annotation as you create your recording. Enter a notes annotation in the **Notes** box.



This is what the notes annotation looks like in the "bubble" in the task guide.



- **Info step**: These annotations are created by right clicking on a control or anywhere on a form < **Task recorder** < **Add info step.** Info steps appear as a numbered step at whatever point you insert it, even though no action was recorded in the UI. You can add a form-level info step or an info step associated with a control. When an info step is associated with a form, the task guide "bubble" will appear someplace on the form, with no pointer, when the task guide is played. When an info step is associated with a control, the task guide "bubble" will point to the control when the task guide is played. In the Help pane, an info step annotation will appear as a numbered step with whatever text you entered. Use info steps to prepare the user for the next steps, to describe steps that need to be done outside of the application, or to refer to other recordings (although you cannot create hyperlinks in annotations).

Determine how long to make your recording

- The user will generally either read or play the recording from start to finish, so don't combine steps or tasks that are better done separately.
- Try not to record a long scenario that spans multiple sub-processes. For example, "Operate in-store customer service desk" is too broad; break it up into shorter tasks such as "Accept returns" and "Add to gift card."
- If a task can be carried out as part of several different business processes, create a separate recording for it, and you can refer to it in the other recordings.
- If the process involves multiple tasks that the person likely does all at once, you can keep the tasks in one

recording, for example, “Set up and assign functionality profiles.”

- If it is something someone does once (such as configuration) and then another task that they can do immediately afterward but may do repeatedly, and on its own, break them up into two task recordings.

Decide where, in the UI, to start a recording The page that you are on when you start recording a task recording affects which page the task guide is displayed for. For example, if you want your task recording to be listed in the Help pane when a user clicks Help on the General ledger parameters page, you must start your recording on the General ledger parameters page. **Save recordings as .axtr files** When you are done creating or editing a task recording, you are presented with several options for how you want to download, or save the recording. You can download the file as a task recording package (.axtr), download it as a raw recording file (.xml), download it as a Word document, or save the file to an LCS library. It is a good idea to always save your task recording as a task recording package file (.axtr). This will help make maintenance of the file easier if procedures or annotations need to change later. If you want to download the file as a Word document, also save it as a task recording package file.

Create your task recording

For detailed walk-through steps, see Task recorder resources.

Copy and customize Microsoft's task recordings

You can download and edit Microsoft's task recordings to use them for your own Help documentation or training materials. To download a Microsoft task recording, follow these steps:

1. Open Task recorder. Task recorder is located in the **Settings** menu.
2. In the Task recorder pane, click **Maintain a recording.**
3. Under **Where is the recording**, click **It is in an LCS library**.
4. Click **Select the LCS library**.
5. Select the Microsoft global library.
6. In the tree, select the business process library node that the task recording is associated with.
7. Click **OK**.
8. Click **Start**.
9. At this point, step through the recording, changing any steps as you go to re-record it. **Note**: If you only need to change the text of a recording, you can open the recording in **Edit a recording's annotations** mode, and then save it.
10. After the recording has played to the end, click **Stop** in the task recorder bar at the top of the screen.
11. Choose how you want to save the task recording.

Additional resources

Help system

Connect the Help system

Task Recorder

Create Rich Help Topics with Task Recorder (external link)

Supply Chain Management home page

2/18/2021 • 2 minutes to read • Edit Online

This topic provides a list of the help topics and other resources in Dynamics 365 Supply Chain Management.

What's new and in development

Go to the Dynamics 365 Roadmap to see what new features are released and what new features are in development.

Core concepts and tasks

Select a feature area to learn more about it.

- Asset management
- Cost accounting
- Cost management
- Inventory management
- IoT Intelligence
- Master planning
- Procurement and sourcing
- Product information management
- Production control
- Sales and marketing
- Service management
- Transportation management
- Warehouse management

Dynamics 365 Finance

For information on Dynamics 365 Finance, go to the Finance home page.

Videos

This short video summarize the new supply chain management features added to Microsoft Dynamics 365 for Finance and Operations version 8.0 (April 2018).

- Synchronize a work order between Field Service and Finance and Operations

These short videos summarize the new supply chain management features added to Microsoft Dynamics 365 for Finance and Operations, Enterprise edition 7.3 (December 2017).

- Prospect to cash integration
- Optimization advisor
- Use warehouse template to copy configuration

These short videos summarize the new supply chain management features added to Microsoft Dynamics 365 for Finance and Operations, Enterprise edition (July 2017).

- Get started with Cost accounting

- Cost control mobile workspace
- Use Excel for cost analysis
- Approve purchase orders on a mobile device
- Visual scheduling with Gantt chart for production and batch orders

The following tech conference recordings discuss supply chain management functionality from previous versions of Finance and Operations. This functionality is now part of Dynamics 365 Supply Chain Management; the same concepts still apply, and the procedures are similar in the current version.

- **Cost management:**
  - Overview of Cost management
- **Master planning:**
  - Extend the demand forecasting functionality
  - Master planning - tips and tricks for troubleshooting performance
  - Help! MRP is slow!
- **Product information management:**
  - Product configurator in Microsoft Dynamics AX
- **Warehouse management:**
  - Get the best out of your warehouse management system
  - Dynamics AX 2012 R3: Advanced warehouse management - A day in the life of process manufacturing
- **Production control videos:**
  - Subcontracting operations and activities in manufacturing
- **Transportation management videos:**
  - Transportation management (TMS) in the new Microsoft Dynamics AX

Blogs

There are many topics about manufacturing and supply chain management on the Dynamics AX Manufacturing R&D Team Blog and Supply Chain Management in Dynamics AX R&D Team Blog. Most of these were written for the previous version, but the same concepts still apply, and the procedures are similar in the current version.

White papers

- Lean manufacturing: Capable to promise and kanban job scheduling
- BOM calculation by using a costing sheet

eLearning courses

For online courses and training, check out Dynamics 365 Supply Chain Management on Microsoft Learn.

Finance home page

2/18/2021 • 2 minutes to read • Edit Online

This topic provides a list of the help topics and other resources for the financial management features in Microsoft Dynamics 365 Finance.

Select a feature area to learn more about it.

- Accounts payable
- Accounts receivable
- Budgeting
- Cash and bank management
- Cost accounting
- Expense management
- Financial reporting
- Fixed assets
- General ledger and Financial reporting
- Project management and accounting
- Public sector

Additional resources

Blogs

- Microsoft Dynamics 365 blog
- Financials blog
- Microsoft Dynamics Operations Partner Community Blog

Task guides

Additional help is available as task guides inside Finance and Operations. To access task guides, click the Help button on any page.

Videos

Check out the how-to videos that are now available on the Microsoft Dynamics 365 YouTube Channel.

Country/region functionality

Country/region regulations affect tax setup and other areas of financial management. Refer to the Localization and regulatory features section of our help content to learn about country/region-specific functionality.

Additional content

Supply chain management functionality covers parts of the procure-to-pay process that includes requisitioning, ordering, receiving, invoicing and paying for the goods and services your organization purchases. Refer to the Supply Chain Management home page for information about the capabilities for managing purchases, inventory, and manufacturing.

NOTE

Can you tell us about your documentation language preferences? Take a short survey.

The survey will take about seven minutes. No personal data is collected (privacy statement).

Commerce home page

2/18/2021 • 2 minutes to read • Edit Online

Dynamics 365 Commerce—built on the proven Dynamics 365 Retail capabilities—delivers a comprehensive omnichannel solution that unifies back-office, in-store, call center, and digital experiences. Dynamics 365 Commerce enables you to build brand loyalty through personalized customer engagements, increase revenue with improved employee productivity, optimize operations to reduce costs and drive supply chain efficiencies, ultimately delivering better business outcomes.

This release enables the creation of digital experiences using built-in web authoring and development tools to produce engaging and intelligent digital storefronts. A connected marketing and headless commerce platform further enable the seamless management of content, assets, promotions, inventory, and pricing across all channels.

- **Everything to build and run digital commerce:** Streamline your business and end-to-end commerce solution that scales to your needs across traditional and emerging channels. Built-in web authoring and development tools enable you to create engaging intelligent digital storefronts, while a connected marketing and headless commerce platform enables seamless management of content, assets, promotions, inventory, and pricing across channels.
- **Build loyalty and exceed customer expectations:** Use clienteling tools to gain a comprehensive view of your customer and respond to their needs at every level of engagement, based on customer profile, history, and preferences that flow across physical and digital channels. Empower your employees to foster lasting relationships through AI-driven recommendations, customer insights, and loyalty programs that elevate brand appeal.
- **Flexible and intelligent omnichannel experience:** Unify physical and digital commerce by providing consistent experiences to customers across cloud search and discovery, product reviews, wish lists, inventory, gift cards, and loyalty. Allow customers to purchase when, how, and where they want, on any device—while providing choice around modern payment methods and product collection or delivery.
- **Streamline operations using AI in the cloud:** Drive omnichannel commerce experiences and integrated, optimized back-office operations through ingrained, pervasive, and context-aware cloud intelligence. Use advanced merchandising, inventory management, distributed order management, and pricing and promotion to innovate and stay ahead of competition.Derive insights by visualizing and analyzing comprehensive and consistent data across all aspects of your business. Use AI-driven technologies to provide accessible websites, protect your business against payment fraud, and efficiently moderate user-generated content like ratings and reviews.

Core concepts and tasks

Select a feature area to learn more about it.

- Configure a Commerce preview environment
- Commerce architecture
- Set up your channels
- Merchandising your products and services
- Manage your orders
- Manage your customers
- Manage your financials

- Manage your e-Commerce site
- Fraud protection
- Commerce development and extensibility

> **NOTE**
>
> Can you tell us about your documentation language preferences? Take a short survey.
>
> The survey will take about seven minutes. No personal data is collected (privacy statement).